EPFL

Thèse n° 12 067

Robust machine learning for neuroscientific inference

Présentée le 17 janvier 2024

Faculté des sciences de la vie
Chaire Fondation Bertarelli de Neuroscience Intégrative
Programme doctoral en neurosciences

pour l'obtention du grade de Docteur ès Sciences

par

Steffen SCHNEIDER

Acceptée sur proposition du jury

Prof. C. Petersen, président du jury
Prof. M. Mathis, Prof. M. Bethge, directeurs de thèse
Prof. A. Hyvärinen, rapporteur
Prof. A. Pouget, rapporteur
Prof. V. Cevher, rapporteur

École
polytechnique
fédérale
de Lausanne

2024

L'art est fait pour troubler,

la science rassure.

– Georges Braque

Für meine Großmutter Thea Rinscheid,

und in Erinnerung an meine Großeltern

Herbert Rinscheid, Marianne & Norbert Schneider

Acknowledgements

Doing a PhD was quite a journey, and I was fortunate to meet a lot of amazing people along the way. I want to firstly thank my two PhD advisors, Matthias Bethge and Mackenzie Mathis.

When I first met Mackenzie during a short visit to her lab at the Rowland in 2019, it was inspiring (and still is) to see her vision on studying adaptive behavior and the energy that goes into pursuing her research program. I was the first PhD student to join the lab, and hence experienced much of the process of building a research team from the start. Over the years, I was fortunate to work with her on ambitious and long-term projects, and I am grateful for the numerous hours of time she invested into mentoring me. As someone good at programming but without any experience shipping software to the research community, it was inspiring to see the amount of work it takes to make scientific software (like DeepLabCut) successful, and the effort Mackenzie puts into open software herself. Mackenzie profoundly influenced my own approach to science, especially through to the rigor she applies when discussing research, and her great eye for detail (I would have never imagined my PhD advisor helping to align elements in my paper figures, among other examples).

I met Matthias a bit before Mackenzie when applying for PhD positions in 2018, eventually writing my Master's thesis in his lab – my first contact to domain adaptation research. Back then, he was leading one of the few NeuroAI labs around the world. It was interesting to see the organization structure of a bigger lab with multiple existing collaborations tackling various aspects of neuroscience research merged with machine learning. When I mentioned that I was interested in sensorimotor research (to study adaptation from the perspective of ML and neuroscience), Matthias put me in touch with Mackenzie, which was the start of the best PhD setup I could have possibly hoped for. I thank him for his openness for collaboration, which was also implemented within the lab culture and helped with getting started. I thank him for many inspirational ideas, and also flexibility to explore and pursue my own. With the Tübingen AI Center and ELLIS, Matthias also created a unique research environment in Tübingen and beyond, which added a lot to my experience, and made this form of joint PhD possible.

Another very influential person that made my PhD journey enjoyable is my partner, Luisa Eck. I am grateful for having her on my side, navigating academia is much more manageable and enjoyable as a team. We also started collaborating at some point

(potentially due to being each other's only in-person contact during the COVID-19 pandemic) which became a lot of fun, and is still ongoing. Luisa taught me a lot about doing science without a computer, and it was great to ponder about ML theory together.

It also meant a lot to get appreciation and a good scientific discussion after presenting my thesis work. I want to thank the members of my thesis committee, Carl Petersen, Aapo Hyvärinen, Volkan Cevher and Alex Pouget for their time and effort assessing and discussing my thesis. I want to thank Aapo, for all his inspiring work on contrastive learning and identifiability that got me interested in the field in the first place.

I also had a lot of luck in meeting the right supervisors, managers and mentors before and during my PhD during multiple internships and research visits. I want to thank Wieland Brendel, for mentoring me on a couple of projects in the first half of my PhD, and introducing me to identifiability problems. I want to thank Alex Mathis for a few fun collaborations, chats about neuroscience, and great BBQ. I want to thank Michael Auli, for his mentorship during my AI residency at FAIR and his profound influence on my approach to rigorous machine learning research. To Alexei Baevski, for teaching me how to scale models and setting up ML experiments at scale. To Ronan Collobert, for many great discussions and expert advice on speech recognition, the first sequence modeling task I worked on. To Bernhard Schölkopf, for his advice and many insightful discussions during my Amazon internship. To Peter Gehler, for his advice on becoming a better software engineer. Thanks to Laurens van der Maaten and Ishan Misra for introducing me to vision-language models during my internship project at Meta in New York.

Many of my papers were written in collaborations. It is great to have worked in labs with cultures supporting collaboration and an open mindset about it, and I thank Matthias and Mackenzie for creating these environments. I want to thank all my colleagues in the Bethge and Mathis labs: To Evgenia Rusak, for two fun projects on robustness and adaptation, and for starting to collaborate with me right after I joined as a PhD student. To Roland Zimmermann, for interesting discussions on ML theory. To Matthias Kümmerer, for great discussions about math, eye movements and the best cocktail bars around the world. And to Karan Desai and Julius Berner for the fun trips to NYC, London and NOLA for testing these recommendations.

Thanks to all other collaborators over the past years: Jessy Lauer, Shaokai Ye, Yash Sharma, Tom Biasi, Mert Yüksekgönül, George Pachitariu, Matthias Tangemann, Julius von Kügelgen, Francesco Locatello, Thomas Brox, Alberto Chiappa, Maxime Vidal and Ori Press and to everyone else in the Mathis and Bethge labs not explicitly listed for being great colleagues and making me feel welcome while hopping between two labs.

On that note, I want to thank Heike König, Laure Blanc-Miéville and Alice Emery-Goodman for helping me jump over the administrative hurdles during my PhD. Thanks to Tuebingen AI, EPFL and the Rowland Institute at Harvard for hosting me, and thanks to the Federal Ministry of Education and Research (through Tuebingen AI), the Swiss

National Science Foundation, and Google for funding my PhD research.

A lot of work was done in collaboration with great undergraduate and Master's students. Thanks to Rodrigo González Laiz, Jin H Lee, Célia Benquet, Shubham Krishna, Anastasiia Filippova, Mert Yüksekgönül, Khushdeep S. Mann, Xingying Chen and Jan Hansen-Palmus for working with me. I learned a lot during this time, and the positive and rewarding experience to work with students considerably influenced my decision to continue with a career in academia.

Thanks to everyone in the open source community for writing and maintaining the various packages and languages I used for my work: PyTorch, LaTeX, datajoint, pgfplots/tikz, numpy, scipy, pandas, matplotlib, SLURM, Docker, Python and many other occasionally used ones. Thanks to Léo Belzile, Gael Lederrey, Nicolas Tappy, Mathias Payer and the tufte-latex contributors for writing the thesis templates I used as the basis for this document.

Doing science would be half the fun without the chance to communicate it to the public. I am very fortunate for everyone working with me on scaling KI macht Schule, a small student project into a Germany-wide non-profit organization. I cannot possibly name everyone of the 100+ people investing their spare time in science communication, but want to especially thank Nicolas Berberich, Auguste Schulz, Leon Hetzel, Christian Hölzer, Sarah Schönbrodt, Lothar Sebastian Krapp, Laura Helleckes, Jasper Albers, Paul Pommer and Elena Natterer for their countless hours of pondering how we can bring AI topics to schools. I also want to thank many other friends and peers not explicitly listed here.

Becoming a scientist was certainly not a standard job choice I had on the radar when I grew up. I want to thank my family and especially my parents. To my mum, Sieglinde Schneider, for investing a lot of time in my education. To my dad, Andreas Schneider, for getting me interested in engineering. To both of them, for always letting me pursue my interests, with all the travel and moving this implied.

Preface

Prof. Mackenzie Weygandt Mathis, Ph.D.
Assistant Professor
Bertarelli Foundation Chair of Integrative Neuroscience
École Polytechnique Fédérale de Lausanne (EPFL)

December 21st, 2023

Adaptive behavior is a hallmark of biological intelligence, yet we do not fully understand how the brain orchestrates this process. Moreover, the brain is impressively robust and can rapidly learn—making it an exemplar system to understand biological stability and flexibility, and ultimately imbue this into artificial systems. While reaching this goal will certainly take a number of projects, people, and Ph.D. dissertations, it is my great pleasure to write the preface for Mr. Steffen Schneider's dissertation, which marks a critical milestone in this quest.

In 2019, the year Steffen joined my group, I was a Rowland Fellow at Harvard University, having started a group in 2017. I was determined to record from as many neurons simultaneously as possible while mice carry out complex behaviors in order to understand the neural basis of adaptive motor control. New microscopy techniques were emerging that allow for hundreds to up to one thousand genetically-identified neurons in cortex to be imaged, and—just as Steffen joined—there were new electrophysiological probes (Neuropixels) that allow for hundreds of neurons to be simultaneously recorded at spiking resolution across the brain. Deep learning was also just starting to impact data analysis in neuroscience: we could begin to markerlessly measure movement with computer vision tools like our package DeepLabCut—DeepLabCut provides neural networks that can be tailored to keypoints of interest with relatively little annotation data thanks to transfer learning (Mathis & Mathis, 2020). In parallel, new neural pre-processing of imaging and neural data were emerging. But, there was still a significant effort needed to build data processing and analysis tools in order to begin to answer fundamental questions about the causal neural basis of adaptive behavior (not to mention a myriad of theoretical questions about how robust and efficient deep learning was and/or needed to become in order to be deployable at scale).

Therefore, it was my utmost pleasure to have Steffen join as my first Ph.D. student.

He was keen to work at the interface of experimental neuroscience and machine learning with myself and Prof. Matthias Bethge (Tübingen, DE), and proposed to work on understanding robustness in brains and machines. Steffen joined as an ELLIS PhD student. Over the next three years he spent time between our groups in Cambridge, Tübingen and Geneva (my lab moved to the EPFL in 2020).[1]

To this end, Steffen has worked with me on building methods to help answer the critical question of how neural circuits adapt to dynamically changing environments. The basis is rooted in pioneering work building computational models that can account for how humans can rapidly adapt to novel environmental perturbations (Todorov & Jordan, 2002)—such as a velocity-dependent force field that pushes your hand off course as you move a robotic handle in center-out reaches (Shadmehr & Mussa-Ivaldi, 1994). We aimed to study the biological mechanisms of this adaptation, thus I had translated this to a mouse model (Mathis et al., 2017), and in my lab we study how cortical circuits in mice change under these perturbations in a model-guided manner, while others have pushed this in non-human primates (as Steffen and I reviewed in a Dispatch (Mathis & Schneider, 2021)). At the time, multi-area recordings were rare, and recording from animals during learning within a session meant you had few-to-no repeated trials. This limited the type of analysis one could do and required considering continuous time-series data more rigorously. Therefore, early on it was clear that new methods for analyzing neural population dynamics would be increasingly critical.

Thus, the stage was primed for algorithmic innovation, and Steffen embarked on a masterful course of merging rigorous neuroscience, machine learning, and mathematical theories to culminate in what is a remarkable dissertation yielding advances in both machine learning robustness and neural analysis. He explored how robust methods can be built and tested to impact scientific inference. Importantly, he worked on both fundamental problems in scaling and creating robust machine learning algorithms, and on methods for neuroscience data that often needed domain knowledge.

Under the direction of Prof. Bethge he has worked on "Improving robustness against common corruptions by covariate shift adaptation" (Chapter 2, NeurIPS 2020), "If your data distribution shifts, use self-learning" (Chapter 3, TMLR 2022), "RDumb: A simple approach that questions our progress in continual test-time adaptation (Chapter 4, NeurIPS 2023), and "Contrastive Learning Inverts the Data Generating Process" (Chapter 6, ICML 2021). All of these works were fundamental machine learning contributions.

In my group, as part of our collective quest to understand the circuit mechanisms of adaptation, he made several key intellectual contributions. Together, we published "Pretraining boosts out-of-domain robustness for pose estimation" (Chapter 5, WACV 2021), "Learnable latent embeddings for joint behavioral and neural analysis" (Chapter 7, Nature 2023), and "Identifiable attribution maps using regularized contrastive learning" (Chapter 8, NeurIPS-W 2023). In particular, Chapters 7 and 8 represent phenomenal

[1] It also must be noted he completed his work also in the midst of a pandemic (COVID-19), which made in person interactions difficult for several years, nonetheless he not only persisted, he excelled.

work that he lead in my group (more below). Beyond the bounds of the dissertation he provided co-author contributions to animal pose estimation: "Multi-animal pose estimation, identification and tracking with DeepLabCut" (Nature Methods, 2022), "A primer on motion capture with deep learning: Principles, pitfalls and perspectives" (Neuron 2020), and in revision, "SuperAnimal pretrained pose estimation models for behavioral analysis" (arXiv, 2023).

Chapter 7 introduces CEBRA, a new self-supervised learning algorithm for obtaining interpretable, **C**onsistent **E**m**B**eddings of high-dimensional **R**ecordings using **A**uxiliary variables. Leveraging the framework of nonlinear ICA, it jointly uses behavioral and neural data in a (supervised) hypothesis- or (self-supervised) discovery-driven manner to produce consistent, high-performance latent spaces using a generalized contrastive learning approach with new sampling schemes for both discrete and continuous data. It allows for single and multi-session datasets to be leveraged, which would be important when trials as limited. Steffen, I, and a great master's student, Jin Lee, worked tirelessly to validate its accuracy and demonstrate its utility for both large-scale calcium and electrophysiology (Neuropixels) datasets, across sensory and motor tasks, and in simple or complex behaviors across species—key examples from diverse areas in neuroscience. Scientifically, the paper shows CEBRA can be used for the mapping of space, uncovering complex kinematic features in somatosensory cortex, and rapid, high-accuracy decoding of natural movies from visual cortex, which was not easily possible before. Since its publication in May 2023 has it been cited already 70 times with 12k downloads of the code-base. Thus, it stands to be a **landmark contribution to the field**, and will likely change our ability to quantify the neural basis of adaptive behaviors in large-scale recordings.

I would like to also take this opportunity to thank the people who enriched the work you will find within his dissertation and the funding agencies who supported it. His co-advisor, Prof. Matthias Bethge, as well as amazing students and postdocs such as Ms. Jin Lee, Mr. Rodrigo González Laiz, Ms. Anastasiia Filipova, Dr. Jessy Lauer, and Mr. Shaokai Ye, who also published papers with him from my group. I thank the Rowland Institute at Harvard, the Bertarelli Foundation, the Swiss National Science Foundation, and Google for funding aspects of his work. Additionally, I would like to thank his esteemed dissertation committee, Prof. Carl Petersen, Prof. Volkan Cehver, Prof. Aapo Hyvärinen and Prof. Alexandre Pouget.

Collectively, the presented work is a testament to his intellectual prowess, his unwavering commitment to excellence, and countless hours of deep thinking (and coding!). As another of his defense examiners exclaimed (and I agree), it should be required reading for anyone who works at the intersection of neuroscience and machine learning! As one of his advisors, I can firmly say it was a joy to work with him to ultimately produce nine papers together in the last four years. He also has exceptional standards for data and code production, which meant I learned as much from him as I humbly hoped he learned from me. In particular, working with him on the development of CEBRA will continue to bookmark truly special years of learning

and growth together, and stands as a highlight of my own career. For this I would also like to take the opportunity to thank Steffen for joining my lab, working so hard on such exciting problems with me, and being a wonderful person.

In closing, it is with great optimism and scholarly pride I get to introduce this Ph.D. dissertation to the academic community. Steffen has not only met the rigorous standards expected of a doctoral candidate but has exceeded them, leaving an indelible mark on the landscape of neuroscience and machine learning. I am honored to have played a role in his academic development and eagerly anticipate the impact he will continue to make in the future.

References

Mathis, M. W., Mathis, A., & Uchida, N. (2017). Somatosensory cortex plays an essential role in forelimb motor adaptation in mice. *Neuron*, *93*, 1493–1503.e6 (p. vi).

Mathis, M. W., & Schneider, S. (2021). Motor control: Neural correlates of optimal feedback control theory. *Current Biology*, *31*, R356–R358 (p. vi).

Mathis, M. W., & Mathis, A. (2020). Deep learning tools for the measurement of animal behavior in neuroscience. *Current Opinion in Neurobiology*, *60*, 1–11 (p. v).

Shadmehr, R., & Mussa-Ivaldi, F. A. (1994). Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience* (p. vi).

Todorov, E., & Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, *5*, 1226–1235 (p. vi).

Summary

Modern neuroscience research is generating increasingly large datasets, from recording thousands of neurons over long timescales to behavioral recordings of animals spanning weeks, months, or even years. Despite a great variety in recording setups and experiments, analysis goals are often shared. When studying biological systems, we want to probe and infer the "hidden causes" underlying a phenomenon and their dynamics, though such dynamics can have different underlying structures, and unroll on different time scales. Towards this goal, we need robust methods for processing and analyzing data, and interpreting our findings to inform subsequent experiments. In this thesis, I study the problem of supporting the scientific discovery process by applying machine learning and statistical tools for data processing (§2–§5), analysis (§6–§7), and informing subsequent experiments through interpretability (§8). For *processing*, in §2 I introduce new evaluation paradigms for testing the performance of a computer vision model under distribution shift at deployment time. In many realistic scenarios, a few unlabeled samples from the target distribution are available in such scenarios. I leverage this assumption to propose batch norm adaptation which considerably improves the error rates of current machine vision models on the ImageNet-C and ImageNet-R datasets. I then extend the methodology for test-time adaptation and empirically study the performance of self-learning techniques in §3. I show that self-learning methods are effective at adapting models of all kinds on a range of adaptation benchmarks. While more powerful than batch norm adaptation, self-learning techniques are prone to collapse during long adaptation spans. In §4, I study this problem in-depth, and show through a simple baseline that the only effective solution right now is to perform periodic resetting of the model. In §5, I study the robustness problem in the context of pose estimation, and assert that pre-training is crucial for out-of-distribution performance. For *analysis*, in §6, I study the effectiveness of current self-supervised learning approaches for representation learning, and show that through building of specialized loss functions we can use contrastive learning to solve non-linear independent component analysis for different assumptions on the latent distribution of a dataset. In §7 I design such a loss function, a generalized variant of the InfoNCE loss, and apply the algorithm to several open neuroscience datasets. The method, CEBRA, can perform scientific discovery and hypothesis testing within a single algorithmic framework to jointly model behavioral and neural data. Finally, I extend this model to allow interpretability and propose an identifiable approach to generating attribution maps in §8. This method is able to attribute latent and observable factors back into the original signal space. Such methods can close the loop to informing data collection for the next iteration of experiments by proposing worthwhile interventions. This work is a step towards more reliably using machine learning methods for science, where reproducibility and robustness is of even greater interest than in engineering applications.

Keywords: robustness, adaptation, computer vision, self-supervised learning, identifiability, neuroscientific inference, neural population analysis, interpretable machine learning

Zusammenfassung

Datensätze in den Neurowissenschaften sind in den letzten Jahren immer größer geworden und wachsen weiterhin. Wir können die Aktivität von tausenden Neuronen über lange Zeitskalen messen und das Verhalten von Versuchstieren über Wochen, Monate oder sogar Jahre aufzeichnen. Während das Format der Daten stark variieren kann, gibt es gemeinsame Ziele bei der Analyse: Den aufgezeichneten Signalen wird die Existenz latenter (nicht messbarer) Variablen zugrunde gelegt. Es gilt, Struktur und Dynamik dieser Variablen bestmöglich zu modellieren. In dieser Arbeit beschäftige ich mich mit verschiedenen Facetten ebendiesen Problems: Wie kann modernes maschinelles Lernen eingesetzt werden, um Datenverarbeitung (§2–§5), Datenanalyse (§6–§7), und die Generierung neuer Hypothesen zur Aufzeichnung kommender Datensätze durch interpretierbare Methoden (§8) zu ermöglichen und zu beschleunigen? Zur *Verarbeitung* entwickle ich neue Ansätze zur robusten Bildklassifizierung durch Adaption während des laufenden Betriebs (§2). Die dabei zugrundeliegende Annahme, dass einige nicht annotierte Beispiele aus dem Testdatensatz vorhanden sind, trifft auf viele Probleme in den Neurowissenschaften zu. "Batch Norm Adaptation" verbessert zahlreiche Modelle auf den ImageNet-C und ImageNet-R Datensätzen. In §3 erweitere ich diese Methodik um die Verwendung von "Self-learning" und zeige die Anwendbarkeit auf eine noch größere Klasse an Modellen. Dabei demonstriere ich in §4 auch aktuelle Grenzen von Self-learning: Über längere Zeiträume degradiert die Genauigkeit teils bis auf Zufallsniveau, oder ist nicht nennenswert höher als beim periodischen Zurücksetzen des Modells auf den Anfangszustand. In §5 betrachte ich die Rolle von Pre-Training und Adaption im Kontext der robusten Erkennung von Tierposen. Ich zeige, dass die Wahl des Pre-Trainings unabdingbar für eine hohe Genauigkeit außerhalb der Trainingsverteilung ist. Für die *Analyse* stelle ich zunächst in §6 eine neue Theorie zur Verbindung von Contrastive Learning und nichtlinearer Unabhängigkeitsanalyse vor – diese ermöglicht es, durch Wahl der Fehlerfunktion bestimmte Annahmen über die dem Datensatz zugrundeliegenden latenten Variablen zu berücksichtigen. In §7 konstruiere ich eine solche Fehlerfunktion, welche die InfoNCE Funktion generalisiert. Diese wende ich auf verschiedene öffentlich verfügbare neurowissenschaftliche Datensätze an. Die Methode "CEBRA" kann zur explorativen Datenanalyse oder zum Testen neurowissenschaftlicher Hypothesen gleichermaßen angewandt werden. Sie erlaubt die Modellierung multimodaler Datensätze, in den Neurowissenschaften oftmals die Kombination aus Neuronenaktivität und Verhalten. Zuletzt erweitere ich diese Methode in §8 um eine bessere Interpretierbarkeit: "Attribution maps" mit Identifizierbarkeitsgarantien erlauben die Erklärung der rekonstruierten Faktoren im ursprünglichen Signalraum. Diese Arbeit stellt einen Schritt zum vermehrten Einsatz moderner Methoden des Maschinellen Lernens in der (neuro-) wissenschaftlichen Anwendung dar. Verglichen mit anderen technischen Anwendungen von maschinellem Lernen ist es hier ungleich wichtiger, robuste, reproduzierbare und erklärbare Algorithmen einzusetzen.

Stichworte: Robustheit, Adaptation, Maschinelles Sehen, selbstüberwachtes Lernen, Identifizierbarkeit, Neurowissenschaftliche Datenanalyse, Interpretierbares Maschinelles Lernen

Contents

List of Figures

List of Tables

1

Introduction

"How does the brain generate adaptive, intelligent behavior?" might be one of the most interesting scientific questions besides the understanding of the universe in physics, and the origins of life in biology. The scientific endeavor towards solving this question is highly interdisciplinary, and investigates the brain at very different temporal and spatial scales: Single neuron activity informs us about the basic building blocks of neural circuits. Brain-level studies give us a birds-eye understanding of which brain areas are relevant in a particular task. The predominant paradigm in neuroscientific investigation for decades has been the study of brain function in controlled environments, with clear ways to causally test relationships. Yet, the brain is capable of incredibly complex tasks and behaviors – and in recent years, a second emerging way of doing neuroscience was the incorporation of increasingly large datasets. Advances in technology enabled this new paradigm of leveraging large scale data recordings in for scientific discovery in neuroscience.

Many fundamental, basic scientific questions about the brain remain unsolved to date: How does the brain parse a hierarchically organized visual scene, how are objects encoded along the visual processing hierarchy, and which brain areas are involved? Which algorithm is used for planning and navigation in a complex environment, and how is a complex motor plan formed and finally executed? What is the learning algorithm used to acquire new knowledge, build internal models of the world, and how does it relate to the objective functions, architectural implementations and algorithms we study in machine learning today? While many fundamental insights towards addressing such questions in neuroscience were generated by close examination of properties of single synapses, cells, and smaller populations of neurons, neuroscience has unprecedented methods for data collection, and further advances in how we process and analyze data promise to accelerate scientific discovery.

DATA ACQUISITION AND ANALYSIS NEEDS IN THE LIFE SCIENCES have grown exponentially in recent years due to new recording paradigms, better hardware systems and imaging techniques. Particularly in neuroscience, we are now able to record from thousands of neurons during increasingly complex tasks through electrode arrays with hundreds of recording sites (Jun et al., 2017; Maynard et al., 1997; Steinmetz et al., 2021) and powerful optical imaging techniques (Denk et al., 1990; Grewe et al., 2010; Sofroniew et al., 2016; Svoboda et al., 1997), some of which are applicable in freely moving animals (Ghosh et al., 2011; Lecoq et al., 2023). The high-dimensional neural recordings are paired with behavioral data acquired by modern microscopes or camera data processed by high-throughput computer vision systems (Mathis et al., 2018, 2020; Nath et al., 2019).

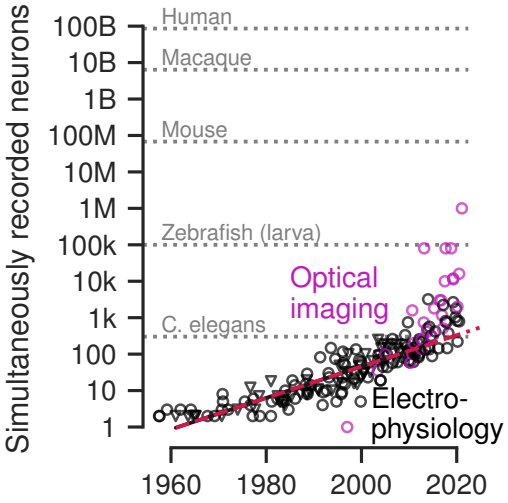


Figure 1.1: Growth of recording capabilities in neuroscience over the recent years. Figure adapted from Urai et al. (2022a) (CC-BY), extended from Stevenson and Kording (2011).

The number of simultaneously recordable neurons continues to grow exponentially (Fig. 1.1), which bears tremendous opportunities for scientific discovery in neuroscience (Stevenson & Kording, 2011). Larger scale recordings bear the opportunity to investigate coding of information in neuron populations, spatially closely related, as well as distributed distributed across multiple areas in the brain: As Urai et al. (2022a) review, insights about sparse and distributed representations, examples for neural computations that are only detectable at the population level, computation of low-dimensional neural manifolds for representing behavior, and the orchestration of activity cross large network architectures were only possible by larger scale recordings of activity. As data recording methods become increasingly capable and complex, our current bottleneck are robust, interpretable and reproducible algorithms that are able to help us model and understand the resulting datasets.

Moving towards large-scale and high-throughput recording and analysis pipelines is already common and established in other fields like high energy physics and astronomy: Research projects in these areas require the interplay of theoretical, experimental and computational approaches. Research collaborations like the Large Hadron Collider (LHC) at CERN, or the Event Horizon Telescope (EHT) aim to record large amounts of raw data, paired with computational approaches to filter, process and model these datasets for testing scientific hypotheses. Yet, a key difference between neuroscience and physics is our lack of equations and theories describing the functioning brain. Killcoyne and Boyle (2009) assert that "life sciences research is, by nature, borderline

chaotic".

In physics, scientists record large datasets for different reasons: in the LHC, the events of interest are rare, and long recording times along with filtering algorithms are necessary to detect these events. In contrast, the EHT's result in 2022 was to create an image of a black hole by fusing data[1] recorded over hours of recording times in the years 2009–2017, which are then combined using theory-grounded computational approaches (Akiyama et al., 2019). In both cases, robust processing algorithms are crucial for successful application of computational approaches. While in the filtering case, non-robust approaches could fail to detect a positive event, or produce false alarms, in the latter case a non-robust approach could yield wrong downstream conclusions in the further analysis of the phenomenon. In physics, fortunately, a lot of theory backs these computational approaches, and in many cases, exact equations are known for informing and testing the employed models.

This is less the case in the life sciences. We are also interested in observing complex dynamical systems like the brain and improving our understanding of them through observation and intervention. Computational approaches based on machine learning promise to vastly increase our data processing capabilities, and are urgently required in the light of our increasingly powerful recording techniques. Yet, compared to our precise physical understanding of a microscope, camera system or an electrode array which are used to acquire data, modern machine learning tools and their behavior on real world datasets are relatively poorly understood. ML models typically lack guarantees for producing accurate outputs, can collapse into failure modes under distribution shift (Geirhos et al., 2020), cannot be easily adapted to a particular experiment or setup and can yield hard-to-reproduce results. It is therefore worthwhile to both empirically and theoretically investigate the behavior of modern ML tools, and equip algorithms with model architectures, objective functions and optimization procedures that aid the interpretability and understanding of these algorithms. As Wilson et al. (2014) argue, "software is just another kind of experimental apparatus and should be built, checked, and used as carefully as any physical apparatus".

If we succeed at improving robustness and interpretability of our ML tools, they can enter the scientific discovery process at multiple stages. To aid the further discussion, we focus on different phases of scientific discovery in the life sciences (Figure 1.2): (1) data collection, (2) data processing and (3) data analysis. These three phases are often closely interwoven, or even interdependent. In all three phases, machine learning has enormous potential to accelerate and scale up the process of scientific discovery in an area of increasingly growing datasets.

In the following, I present progress on machine learning systems for the data processing and data analysis phase, and close the loop to inform the data collection process of successive experiments by utilizing explainable AI methods. I aim to address the key question how modern machine learning methods and new statistical

[1]The EHT produces multiple peta-bytes (PB) of data, https://eventhorizontelescope.org/faq/how-much-data-recorded-during-observation-and-how-it-transferred-central-processing

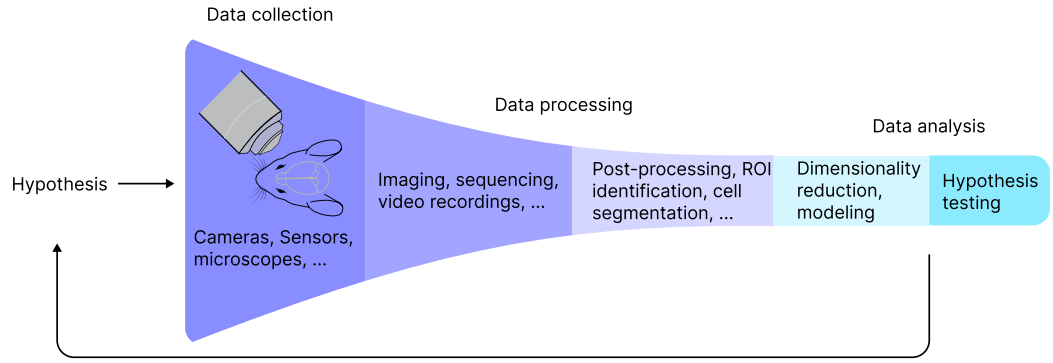


Figure 1.2: The process from a hypothesis to conclusion in scientific inference is a multi-step process comprising data collection, processing and analysis. Machine learning or statistical algorithms can assist at every step of this process if the right methods are available to ensure robustness of the models used.

techniques can accelerate our progress in understanding brain function at the system and population level: How can we more robustly preprocess data to arrive at large scale datasets for investigation? How can we use new statistical tools to study the interplay between neural activity and behavior during complex tasks, with many confounding factors? How can we inform next experimental interventions from these statistical insights?

For *data processing*, I propose methods for robust learning which are applicable to biological imaging and to many other real-world technical applications involving image and video acquisitions systems. I focus on building methods that separate their training and deployment phases, and are adaptable to changing world conditions and setups during deployment. For *data analysis*, I equip machine learning systems with identifiability guarantees, allowing data to be modeled and statistically analyzed in more reproducible ways. I specifically focus on techniques whose theoretical guarantees hold empirically in realistic evaluation scenarios and on real datasets. To close the loop to *data collection*, I build algorithms with theoretical guarantees that allow for post-hoc explainability to inform scientists on new experiments and/or influence the data collection process on the fly. As with our data analysis techniques, these theoretical guarantees should be tested on synthetic and real world datasets.

Approaches to understanding intelligent behavior

Modern neuroscience and artificial intelligence share a lot of history, and are of comparable age: "Modern neuroscience" as a separate scientific discipline emerged in the mid 20th century, the term "artificial intelligence" was first coined at Dartmouth college in 1956 (McCarthy et al., 2006)[2]. While there were some overlaps between

[2]The proposal started with, "The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine

the fields throughout their history, and some neuroscientific insights informed AI research at a rather high level (e.g., the NeoCognitron by Fukushima, 1980), both fields developed mostly independently and with separate focus. Today, machine learning became an important component in many signal processing systems, and in this section we examine how it can also be beneficial for data analysis in neuroscience, to generate new scientific insights. Today, both fields are in interesting stages:

Neuroscience grew massively in scale and availability of data, and the applicable experimental procedures for collecting data. Scientific collaboration projects like the International Brain Laboratory or the Allen Institute's OpenScope project released highly standardized large scale datasets comprised of 10,000-100,000s of neurons[3] (de Vries et al., 2020; International Brain Lab et al., 2023; Siegle et al., 2021a). We are already able to perform whole brain recordings of neural activity in smaller model animals like *C.elegans* (Nguyen et al., 2016; Schrödel et al., 2013), zebrafish larvae (Ahrens et al., 2013; Cong et al., 2017; Kim et al., 2017; Symvoulidis et al., 2017) and move towards whole brain recordings in drosophila (Aimon et al., 2019; Mann et al., 2017), cf. Urai et al. (2022a) for an extended review. While these advances in the scale of recording methods do not replace investigation of single neuron properties and characteristics at the cell level, they certainly open interesting avenues for analyses that were not possible before.

Machine learning experienced many breakthrough moments since the success of convolutional networks on the ImageNet benchmark (Deng et al., 2009a). After the individual success of deep learning with specialized architectures in speech, vision, and natural language processing, the transformer architecture and attention mechanism (Vaswani et al., 2017) demonstrated the first proof of concept of a single, replicated building block that could excel on all disciplines with little domain-specific changes. With the transition to foundation models (Bommasani et al., 2021) – models pre-trained on vasts amount of data, which excel then on a variety of downstream applications with only little additional training samples – we are now at a stage were intelligent agents can productively interact with humans during activities like writing, programming, and creative work.

Interestingly, despite this sharp increase in data and simulation capability, it remains an open problem of *how* and *why* biological and artificial systems exhibit complex and intelligent behaviors. Our understanding of intelligent systems is still very limited, and the tools to explore and study it at the present scale are lagging behind. Despite being able to fully observe whole model animals, or measure the activity of every single neuron while a foundation model generates a complex response, we are unable to pin down the computational principles underlying this behavior. *New computational tools for processing and analysis of experimental data are urgently needed.*

---

can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves."

[3]For a sense of scale, the IBL dataset is comprised of recordings from 116 mice, with 295,501 units (before sorting), two Neuropixels insertions per session International Brain Laboratory (2023).

How can neuroscience benefit from machine learning?

Neuroscientific data analysis traditionally operated across different spatial scales and timescales. When "modern neuroscience" formed in the 1960s, it was heavily influenced by the principles of reductionism, a well-known approach in physics. Abi-Rached and Rose (2010) characterize the field of modern neuroscience as a "hybrid of hybrids", merging lines of work from previously separated disciplines. In contrast to the more distinct fields of psychology, anatomy and neurology which existed well before, Kandel (1982) asserted that a feature of this "modern neuroscience" is its "common purpose" towards the understanding of complex sensory and behavioral phenomena from different angles of investigation.

Many works from this early era of modern neuroscience were concerned with studying the contribution of individual building blocks – neurons, synapses, axons, glia cells, etc. – and connecting them to principles of brain function. A multitude of works emerged that studied the properties of individual neurons in a variety of tasks. Firing properties of individual cells were modeled based on single-unit activity in the well-known experiments from Hodgkin and Huxley (1952) in the squid giant axon. Such models of single-cell physiology remain influential till today. In vision, many experiments strived towards identifying single-neuron properties in various visual areas, beginning with Hubel and Wiesel (1959) who found cortical neurons tuned to particular angles of grating stimuli in the cortex of cats, and later studied the spatial arrangement within cortical columns (Hubel & Wiesel, 1963).

From single neurons to population dynamics

Despite these success stories of investigating single neuron properties, the importance of examining the population level activity was acknowledged since the 1960s. As Moore et al. (1966) note, "observation of multiple units can reveal details of interconnections and functional interactions". Certain phenomena cannot be explained by single neuron activity alone, but an orchestration of activity among multiple neurons which makes up the neural code. A fundamental challenge in neural information processing are the various sources of noise in a biological system – a problem well-studied in information theory, which is concerned with transmitting information optimally across a noisy channel. Consequently, neural coding was hypothesized to follow such optimality principles (Barlow et al., 1961). Later works were successful in making predictions in how e.g. visual coding and filters merge through these principles: Olshausen and Field (1996a, 1996b) showed that response properties of simple-cells neurons were optimal in terms of information compressing under a noise model. This gives rise to a population code that collectively explains how visual stimuli are encoded across neuron populations using a sparse code. Likewise, while being based on single-neuron measurements, emergence of place-cells (O'Keefe & Dostrovsky, 1971) and grid-cells (Hafting et al., 2005a) can be explained through optimality principles on the population

level. This even allows theoretical predictions of firing properties of such cells in different experimental settings, e.g., in 3D environments (Mathis et al., 2015).

Other interesting examples of population codes can be studied in sensorimotor learning. Compared to the previous examples focused on perception and representation of the environment, sensorimotor learning is an interesting field of study for closed-loop experiments between the subject and the experimental environment (for a review, cf. Wolpert et al., 2011). Many phenomena in sensorimotor learning can be explained by principles of optimal control (Todorov, 2004), which raises the question how and where individual computational components are implemented. Strides towards this question require analysis of neuron populations across brain areas, e.g. through theory-informed inhibition experiments (Takei et al., 2021, see Mathis and Schneider, 2021 for a summary) to investigate how theory maps onto individual regions. A finer dissection requires the analysis of population dynamics: Analyzing activity with the assumption of an underlying lower-dimensional representation can aid the understanding of seemingly complex single-unit activity, as Churchland et al. (2012a) show for a reaching task.

Evolution of recording methods

Recording techniques in the neurosciences made substantial progress over the years, and influenced the kind of insights that could be generated (Stevenson & Kording, 2011). Electrodes for recording single neuron activity are used since the 1950s (Hubel, 1957).The tretrode (Recce, 1989) became popular as an extension to single electrodes, allowing to record from a few electrodes simultaneously. Especially for longer-term recordings in primates, larger electrode arrays like the Utah array were introduced (Maynard et al., 1997; Nordhausen et al., 1996), allowing to record from up to 100 recording sites. These recording techniques are still used up to date, despite known limitations, e.g. the impact on surrounding tissue due to the used materials (Patel et al., 2023). Modern electrodes increasingly build on flexible and more bio-compatible materials, allowing longer term recordings (Hong & Lieber, 2019; Tang et al., 2023). Electrode-based solutions continue to increase in scale and recording sites: It is now possible to record from a few hundred recording channels using electrode arrays like Neuropixels (Jun et al., 2017). Neuropixel probes can be inserted at various locations in the brain, including measurement with multiple simultaneously inserted probes.

Besides methods directly recording electrical signals, measuring proxies for information processing through optical methods was investigated since the 1990s (Svoboda et al., 1997) using two-photon microscopy (Denk & Svoboda, 1997; Denk et al., 1990). Such recording methods can be used for whole brain imaging of small organisms as discussed before, or multi-area recordings in e.g. rodents (Sofroniew et al., 2016). The calcium concentration within a neuron is used as a proxy for its activity, and indicators like GCaMP (Tian et al., 2009) are used to measure changes in the cells calcium concentration via fluorescence microscopy.(**calcium imaging!GCaMP**) With suitable microscopes, the indicators are excited using light and the fluorescence in case

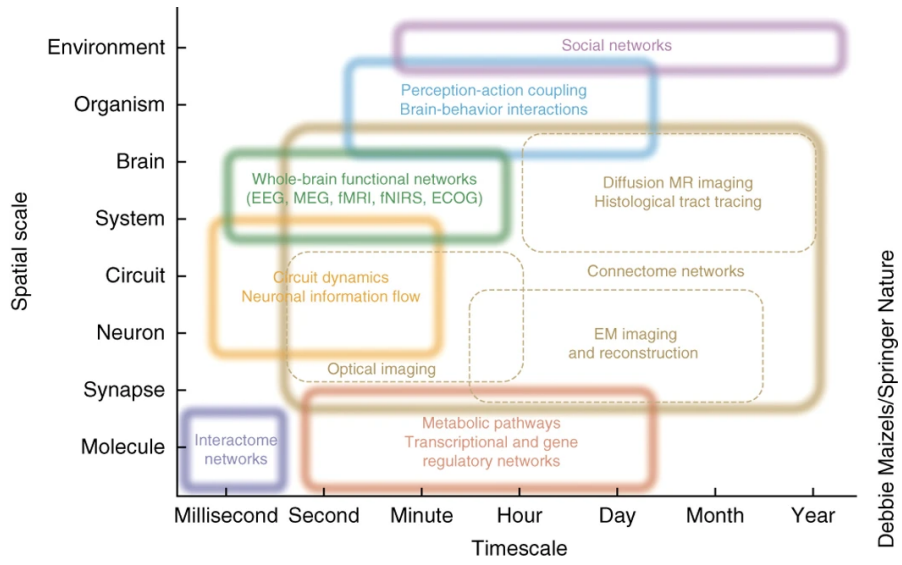


Figure 1.3: A. Overview of recording techniques to measure connectivity in the brain at different time scales and spatial scales. Reproduced from Bassett and Sporns (2017) with permission from Springer Nature.

of calcium concentration in the cell is recorded in form of a video or photon counts. Depending on the technique, the activation energy for the indicator is delivered via a single photons (short wavelength light), two photons as longer wavelength (Denk & Svoboda, 1997; Denk et al., 1990) or, more recently, three photons. The resulting video recording is first segmented into individual regions and then "deconvolved": Calcium indicators have decay times in the range of multiple seconds, effectively low-pass filtering the underlying activity. Such recording methods can also be headmounted on rodents, allowing to record freely moving behavior along side optical recording: Lecoq et al. (2023) review the current state of the field, including single photon (Ghosh et al., 2011), two photon (Sawinski et al., 2009) and three photon (Klioutchnikov et al., 2020; Zhao et al., 2023) variants of such "miniscopes" (Ghosh et al., 2011).

Besides electrophysiology and optical imaging as popular invasive recording techniques, a breadth of other recording techniques exists not covered here – Figure 1.3 visualizes recording methods for connectivity across time and spatial scales. Figure 1.4 depicts the range of functional recording methods, and means for interventions, across maximum recording length and the number of recordable units.

There are still many open computational challenges in neural recordings: Extracellular recordings need post-hoc processing which identify signals from individual cells. While a variety of spike-sorting algorithms exist, there is disagreement on the choice and application of these algorithms, and we need ground-truth datasets (e.g, paired with patched intracellular recordings, Neher and Sakmann (1976)) for the design and evaluation of robust algorithms (Harris et al., 2016). Likewise, optical signals often need to be post-processed with de-convolution algorithms, and for optimal design of such algorithms, paired ground-truth across different brain regions is needed (Rupprecht

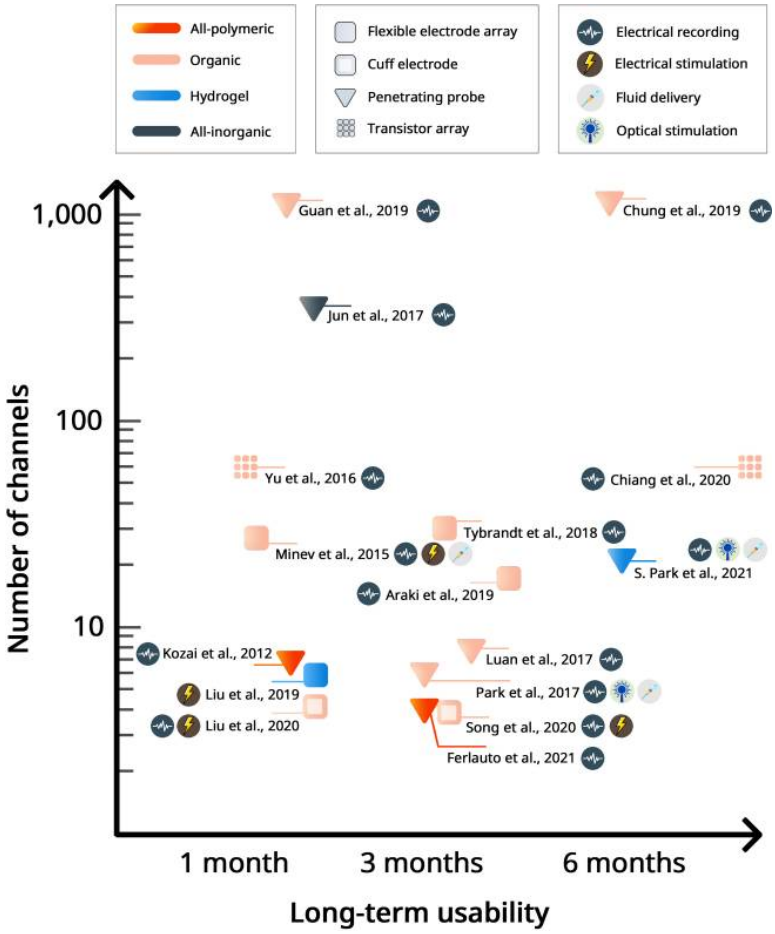


Figure 1.4: Overview of recording and stimulation techniques across recording timescales and channel number. References: Araki et al. (2019), Chiang et al. (2020), Chung et al. (2019), Ferlauto et al. (2021), Guan et al. (2019), Jun et al. (2017), Kozai et al. (2012), Liu et al. (2019, 2020a), Luan et al. (2017), Minev et al. (2015), Park et al. (2017, 2021), Song et al. (2020b), Tybrandt et al. (2018), and Yu et al. (2016). Figure and references reproduced from Hong et al. (2021), CC-BY.

et al., 2021).

The role of behavior in neuroscience

Neuroscience needs well-designed behavior paradigms: The aforementioned reductionist approach to neuroscience traditionally dismisses the importance of experimental design (Krakauer et al., 2017a). Behavioral design has important implications on the observable neural activity during an experiment, and its relevance to make conclusions about activity during naturalistic behavior: The activities observed during a lab experiment (left column in Figure 1.5) will generally be a subset of neural activity patterns we can observe during natural behavior (A). Carefully designed behavioral tasks allow to investigate a subset of natural behavior within a laboratory environment (B). However,

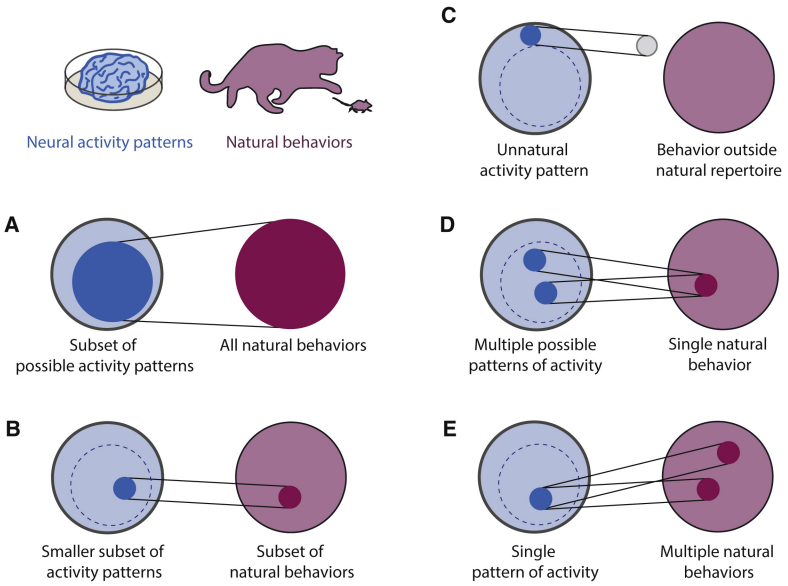


Figure 1.5: Relation between activity patterns measured in lab environments, and during natural behaviors. Reprinted from Krakauer et al. (2017a) with permission from Elsevier, also see Photographic credits.

it is also possible to design experiments yielding to neural activity with no meaningful relationship to activity we can expect during natural behavior (C). Finally, ambiguity is possible: Some experiments might give rise to ambiguity in the space of neural activity (D), others in the space of natural behaviors (E).

Ideally, like with neural data, we require data collection and processing methods for capturing as much behavioral information as possible. Advances in machine learning enabled a lot of progress in this area: A mainstream approach today is markerless tracking of animals using pose estimation tools (Mathis et al., 2018). On smaller model organisms, microscopes that allow whole-body tracking of animals and simultaneous recording of behavior and neural activity are becoming feasible (Cong et al., 2017; Nguyen et al., 2016; Symvoulidis et al., 2017), and virtual reality environment allow closed-loop and fine-grained control of the experimental environment.

Monitoring high dimensional animal behavior relies on machine learning techniques to digitize videos and other forms of raw recordings. Advances in computer vision and core machine learning – e.g. in robustness, or data efficiency – therefore often directly translates into advances in these algorithms (e.g., Mathis et al. (2020)).

Neuroscience needs new analysis methods

Right now, the field is increasingly moving towards naturalistic stimuli and tasks, and it has been suspected that this will give rise to different behavior in the neural code than the activity generated by standardized and stereotypical stimuli like gratings in vision science, or constrained behaviors in sensorimotor tasks.

Why do we work with constrained stimuli and behaviors? Controlled experimental conditions are an effective way of abstracting noise and unwanted processes, and allow clear scientific investigation of a phenomenon. From a standpoint of causal inference, all variables with potential influences on the process are fixed to a constant value, and only the variable under investigation is intervened upon. Consequently, established analysis methods like linear regression, generalized linear models, or linear dimensionality reduction algorithms like principal or independent component analysis are often sufficient to obtain insightful conclusions from such datasets.

However, since task, environment, and the brain form a dynamical system and interact with each other, fixing these experimental variables naturally also constrains the complexity of the generated behavior during the experiment, and potentially also the neural population code we can observe (Krakauer et al., 2017a). The more variables change over the course of an experiment, and the more naturalistic this variation, arguable the more diverse is the signaling activity we can measure.

How to move forward? We should both continue to investigate clearly controlled behaviors with classical statistical methods, but extend our experimental repertoire by moving to more naturalistic and complex behaviors and stimuli (Mathis and Mathis, 2020; Figure 1.6). The latter requires the introduction of analysis tools that can deal with this growing complexity of behaviors and neural activity, and allow to dissect their relationship post-hoc. This mandates the development of suitable processing and analysis algorithms which allow to understand and model the growing complexity of such datasets. Ideally, these algorithms also move beyond static and post-hoc data analysis, but can be integrated into closed-loop experimental systems (Hausmann et al., 2021).

Presently, especially the neuroscience community, is utilizing linear tools for modeling of the relationship between neural and behavioral data. Table 1.1 shows a summary of different algorithms currently used for representation learning and data analysis. An immediate observation is the focus on linear methods in most cases. A common theme is the use of generative models which perform representation learning by learning to reconstruct the original data (typically, neural activity, e.g., spikes or spiking rates of neurons).

A common approach is to input neural activity into an algorithm, compute *factors* from the input neural activity, and then reconstruct the activity from these discovered factors. This approach is applicable in both linear and non-linear ways: For instance, a principal component analysis (PCA), and variants thereof, map input neural activity onto vectors that explain the main axes of variance in the data. Such models can be interpreted as mapping neural activity onto a factor space, before projecting it back onto the neural activity. When reconstruction of neural activity is the goal, the quality of the representation can suffer in settings with recording noise.

Modern self-supervised representation learning methods can offer to circumvent this problem, by separating the processes of data generation and representation learning, and offer a potential improvement over auto-encoding approaches. In particular,

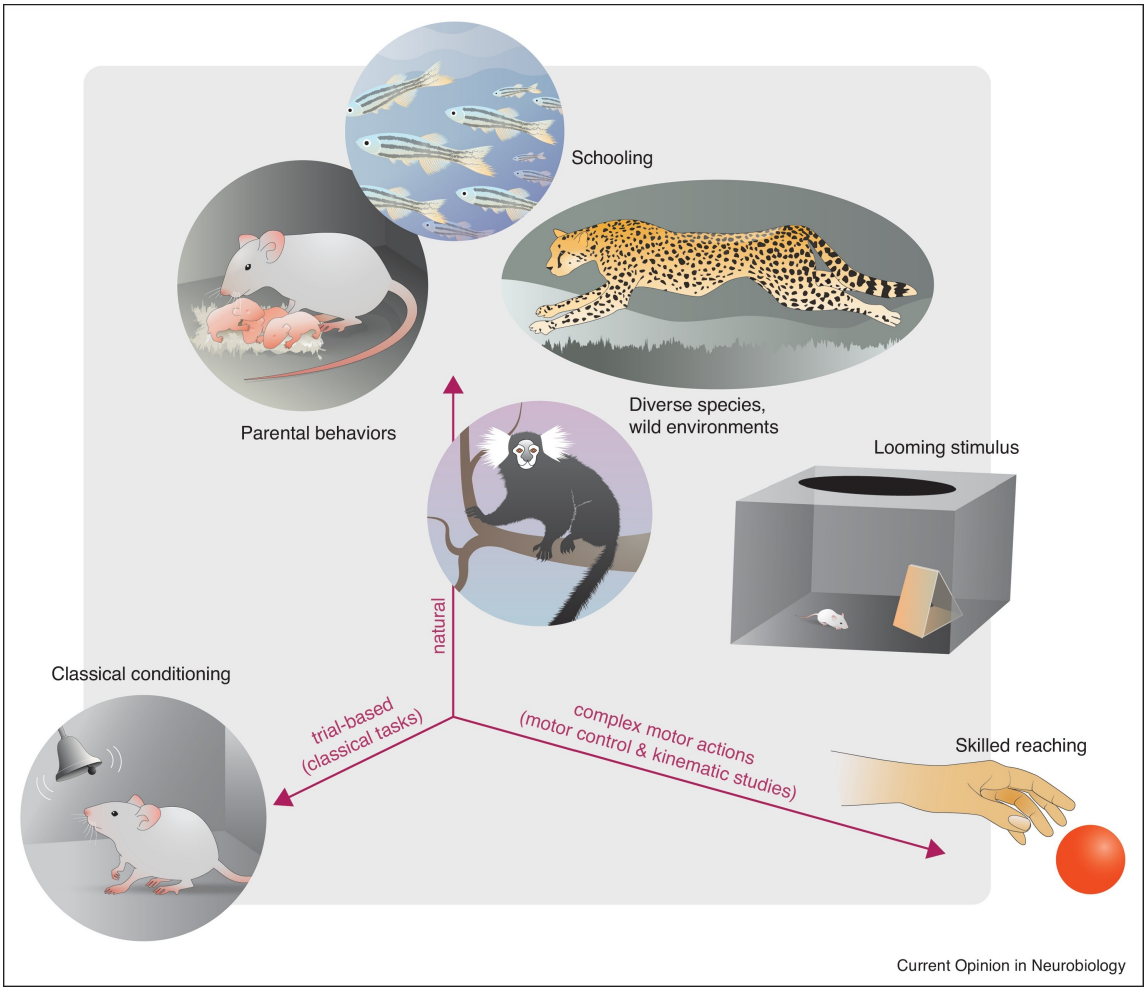


Figure 1.6: The "Behavior space" discussed by Mathis and Mathis (2020) spanning various experimental paradigms. The move towards natural behaviors is possible due to modern processing methods for behavioral recordings. Reproduced from Mathis and Mathis (2020), with permission from Elsevier.

auto-encoding approaches and generative models require to reconstruct the observed data, including the observed noise (Zhou & Wei, 2020). In contrast, self-supervised approaches merely learn an encoder model which maps activity into a latent space. Such an approach might be favorable in cases where reconstructing data is hard (e.g., because of limited data size).

What can machine learning offer to neuroscience?

The brain is a highly distributed processing system, which makes it particularly interesting to study properties of the neural population code – within and across different areas in the brain. If we look at how large machine learning models are built currently, single-neuron properties rarely play a role. Activity vectors, not scalars, are

Table 1.1: An overview of dimensionality reduction algorithms commonly used in neuroscience. Current algorithms can be categorized based on how they connect samples across time points (i.e., assumptions about modeling the trajectories), the assumed mapping function between latent variables and neural activity, assumptions about the observed neural activity, and whether they are computed on single trial data, or averages across recording sessions. I will later propose CEBRA (Chapter 7), which is the only technique not leveraging an explicit generative model, making it invariant to exact observation of the data. References for the displayed methods, from top to bottom: [1]Pearson (1901) and Roweis and Ghahramani (1999) [2]Kobak et al. (2016a) [3]Churchland et al. (2012a) [4]Smith and Brown (2003) [5]Macke et al. (2011) [6]Sani et al. (2021) [7]Gao et al. (2016b) [8]Petreska et al. (2011) [9]Linderman et al. (2017) and Zoltowski et al. (2020) [10]Durstewitz (2017) [11]Pandarinath et al. (2018a) [12]She and Wu (2020) [13]Yu et al. (2008a) [14]Rutten et al. (2020) [15]Zhao and Park (2017) [16]Wu et al. (2017). Reprinted and minimally adapted from Hurwitz et al. (2021) with permission from Elsevier.

Model	Trajectories	Mapping Function	Observation	Single-trial
PCA/FA[1]	Static	Linear	Gaussian	No
dPCA[2]	Static	Linear	Gaussian	No
jPCA[3]	Linear	Linear	Gaussian	No
LDS[4]	Linear	Linear	Gaussian	Yes
PLDS[5]	Linear	Linear	Poisson	Yes
PSID[6]	Linear	Linear	Gaussian	Yes
PfLDS[7]	Linear	Neural Network	Poisson	Yes
SLDS[8]	Switching Linear	Linear	Gaussian	Yes
RSLDS[9]	Recurrent Switching Linear	Linear	Gaussian	Yes
PLRNN-SSM[10]	Piecewise linear RNN	Linear	Gaussian	Yes
LFADS[11]	RNN	Linear	Poisson	Yes
GP-RNN[12]	RNN	Gaussian Process	Poisson/Gaussian	Yes
GPFA[13]	Gaussian Process	Linear	Gaussian	Yes
GPFADS[14]	Gaussian Process	Linear	Gaussian	Yes
vLGP[15]	Gaussian Process	Linear	Poisson	Yes
P-GPLVM[16]	Gaussian Process	Gaussian Process	Poisson	Yes

the fundamental building blocks for representing information in these systems.

In deep learning models, every synaptic weight, and all activity vectors are available in any given point in time. Yet, there is no automatic analysis tool yet that allows to to precisely characterize *why* and *how*, e.g., a large language model is performing a particular task. Likely, even the ability to record full-brain activity and behavior in a living organism would not allow us to actually fuel our understanding of the brain, and we can assert a bottleneck in the development of modern computational tools. It is likely that Jonas and Kording (2017) would reach similar conclusions today if they decided to understand a GPU running a transformer model instead of a microprocessor[4].

Our current lack of understanding of intelligent behavior in such "fully observable" systems is a possible indication that a lack of recording techniques in neuroscience is no longer a primary bottleneck for scientific discovery – data processing and in particular data analysis is. We have public datasets available comprised of 10,000–100,000 of neurons, which are arguably not exhaustively analyzed by their primary publications

[4]In their study "Could a Neuroscientist Understand a Microprocessor?", Jonas and Kording (2017) assessed the effectiveness of data analysis practices by studying a microprocessor using commonly used neuroscientific analysis tools.

de Vries et al. (2020) and International Brain Lab et al. (2023). Some of these projects draw an analogy to the scientific discovery process in physics which I covered in the introduction[5]. Centralized institutions collect and publish large datasets, and it is upon computationally focused research groups to re-analyse and draw additional insights from these datasets. For that, new methods are needed.

Some of the tremendous progress that happened in computer vision, natural language and speech processing was also transferred to neuroscience. The most prominent application is in data processing, where computer vision algorithms are routinely used to digitize behavioral recordings, segment neurons in optical imaging data, track animals and control experimental equipment in closed loop setups (e.g. Kane et al., 2020; Symvoulidis et al., 2017), spike-sort neurons after large scale electrophysiological recording, apply super resolution to microscopy images or deconvolved calcium activity into signals more similar to spiking rates, to name only a few applications.

A central motivation in this work is to work on two major roadblocks for the application of statistical and machine learning approaches for scientific inference: The robustness of algorithms, and their theoretical grounding required for data analysis. I will examine here which machine learning techniques can be used to aid scientific discovery in neuroscience.

Statistics vs. Machine Learning

An important distinction in the design of algorithms is their categorization into statistics or machine learning. I explore examples for both cases. Ij (2018) summarize: "Statistics draws population inferences from a sample, and machine learning finds generalizable predictive patterns.". We can make the following distinction:

In machine learning, the computational focus is to build a model (during a training phase) which can then be applied to new incoming data. A good machine learning model will be able to make predictions within its application scope. In the context of scientific inference, machine learning systems are prominently used in the data processing phase. For instance, detecting objects or keypoints in a video stream, proposing regions of interest in a microscopy image, or for spike sorting. The machine learning system as such does not directly aid in the analysis of the data: Instead, it cuts the complexity (e.g., poses are a very compact and low-dimensional representation of behavior), improves the signal-to-noise ratio (e.g., during spike sorting), or simply speeds up the data processing phase.

In statistics, the computational focus is on the *analysis* of the dataset. Statistical models are typically precisely theoretically grounded, and have clear underlying assumptions for their validity. Ideally, it is also testable whether a dataset sufficiently meets these assumptions. As a product of these assumptions and the dataset the method is applied to, we obtain an outcome that either increases or decreases our

[5]The OpenScope program initiated by the Allen Institute make this analogy explicit by claiming to be "the first astronomical observatory in Neuroscience"

confidence in the hypothesis motivating our original experiment. This clear theoretical grounding of a statistical technique allows discussion of its results and findings by examining if assumptions are suitable for the dataset at hand. If the assumptions (e.g., an assumed probability distribution) are met, the behavior of the method should be predefined.

This distinction between statistical methods and machine learning tools is useful to distinguish design goals for either category: In machine learning, it is sufficient if the resulting algorithm can be tested like other physical systems – as Wilson et al. (2014) state, such systems can be regarded as part of the experimental apparatus. For statistical analysis, however, we require tools that are grounded in theoretical assumptions about the data that is being analysed. In return, during evaluation of a study, it is possible to criticize the methodology as such (e.g., because dataset and assumptions do not match), but not the correctness of the statistical method in case the assumptions are met.

Core machine learning vs. applied machine learning

While sometimes, computational methodology can be co-developed with a study, modern machine learning tools and their increasing complexity require tools to be studied individually, also beyond the context of a particular study. This allows benchmarking of proposed approaches, and comparisons to prior state of the art not with respect to a particular dataset under investigation, but on a previously agreed set of benchmark problems.

To name one example, spike-sorting is a problem that is very relevant in many processing pipelines for electrophysiology. Yet, building a spike sorting algorithm specifically designed on a per-experiment basis based on the needs for a particular study is not necessarily desirable: Such an algorithm would need to undergo testing and validation, and will finally most likely not transfer to studies. Instead, recent work (Magland et al., 2020) started collecting and curating a variety of datasets. Now, without connection to a particular neuroscientific research question, it is possible to focus on the design and evaluation of a suitable method. The resulting method will – ideally, given a sufficient breadth of data in the benchmark dataset – be applicable to a variety of applications.

Some core machine learning problems relevant for the scope of the present work are robustification of machine learning methods, adaptation to distribution shifts at deployment time, and self-supervised learning. Within these disciplines, we study the behavior of learning algorithms on pre-defined benchmarks. For the robustification and adaptation works, the most relevant benchmarks are on vision datasets, but the developed techniques in principle also transfer to other fields and applications. For self-supervised representation learning, prior work in disentanglement was most relevant.

The importance of robust and adaptive data processing systems

In machine learning, a classical setup to benchmark and evaluate computer vision systems for robustness is adversarial robustness, and the robustness to (systematic) distribution shifts in the data. For applications in the sciences, robustness concerns over adversarial robustness are less prominent than the robustness concerns when it comes to systematic drift in the data distribution, often also called batch effects (for a review in the context of omics in biology, see Goh et al., 2017). Hence, I opt to primarily focus on investigating the robustness of ML models on systematic distribution shifts, either with synthetically induced shifts, or by selecting suitable images.

Just prior to the work on this dissertation, first benchmarks appeared on ImageNet scale (e.g. ImageNet-C, Hendrycks and Dietterich, 2019a) which offered the possibility to rigorously evaluate algorithms in a setup offering a proxy for real-world performance. Other benchmarks included ImageNet-R (Hendrycks et al., 2020a) and ImageNet-D, which I introduce in Chapter 3. Such benchmarks aim to address the reliability of a trained machine learning system at time of deployment. One trained on a large dataset, how does the model perform on data that systematically deviates from the training dataset, and how can we mitigate negative effects on model performance?

Contrastive and self-supervised representation learning

Analysis of high dimensional datasets hinges on learning suitable representations of the data. As Bengio et al. (2013) state in their review, "[in] the case of probabilistic models, a good representation is often one that captures the posterior distribution of the underlying explanatory factors for the observed input". Within representation learning, a central theme for this dissertation is to leverage self-supervised learning[6] algorithms, specifically contrastive learning.

An early successful variant of contrastive learning was the noise contrastive estimation (NCE) algorithm introduced by Gutmann and Hyvärinen (2010): The NCE objective contrasts positive against negative pairs, and casts this as a binary classification problem. Positive pairs are samples in the data should have a similar representation, negative pairs are samples in the data that should have a more dissimilar representation. How this notion of similarity is incorporated into the learning algorithm depends on the problem setting.[7]

Contrastive learning was popularized in applied machine learning. The hierarchical

[6] I will avoid the acronym "SSL" due to ambiguity between semi-supervised and self-supervised learning.

[7] The relation between noise-contrastive estimation (NCE) and InfoNCE is similar to the relation between the binary cross-entropy and the softmax cross-entropy. Noise contrastive estimation using a single negative example minimizes the expectation of

$$-\left[\log \sigma(\psi(\mathbf{x}, \mathbf{x}^+)) + \log \sigma(-\psi(\mathbf{x}, \mathbf{x}^-))\right]$$

over triple $(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)$ in the dataset, with $\sigma(y) = 1/(1+\exp(-y))$ denoting the sigmoid, sometimes with weighting coefficients between both losses, and varying number of samples depending on the exact work.

softmax formulation of the word2vec algorithm by Mikolov et al. (2013) suffered from the problem of marginalizing over a large set of classes (words in the dictionary). A convenient solution was to replace the exact classification problem by the sampling of a few indicative "negative examples". With that, the supervised classification problem became a *contrastive* learning problem which yielded similarly well-suited representations while being computationally more efficient (Mikolov et al., 2013). In this example, a positive pair would be formed by taking a sequence of words, and predicting the word in the center from the surrounding context, while for a negative sampling we would pair the context with a random word from our dictionary.

Later, this idea was generalized to continual data in speech processing. Oord et al. (2018) introduced the InfoNCE objective for "contrastive predictive coding" which performed next token prediction on continuous data using an RNN model. Schneider et al. (2019) performed next token prediction using convolutional networks and the NCE loss on raw speech signals, and showed how self-supervised learning can reduce the need for labeled speech data drastically by at least two orders of magnitude. Following work leveraged the InfoNCE loss with transformer networks (Baevski et al., 2020c) for further drastic improvements in data efficiency and downstream performance. Initially, the "next token prediction" tasks were also proposed for applications in vision (Oord et al., 2018), and later scaled to ImageNet size (Hénaff et al., 2020). Another approach by Misra and van der Maaten (2019) leveraged the NCE loss and demonstrated how self-supervised learning can outperform supervised pre-training for object detection tasks. They used augmented versions of the reference image to form positive pairs, in contrast to prior work which attempted at predicting the transformation (e.g., given an image, predicting the rotation of an image). Chen et al. (2020a) extended this approach further, and were the first to demonstrate a self-supervised pre-training approach to achieve state of the art performance on ImageNet.

Self-supervised learning evolved in various directions in the following years. One example is the development of non-contrastive approaches, which continued to use augmented views of images to obtain representations. DINO (Caron et al., 2021b) used a self-learning approach (similar to pseudo-labeling) on such augmented image pairs, and learned feature representations with favorable properties for k-nearest-neighbor readouts, and for decoding objects. Besides their algorithmic contributions, such works are particularly relevant for data pre-processing in the context of neuroscience: For instance, in Chapter 7 my colleagues and I leverage a pre-trained DINO model for extracting relevant visual features from naturalistic scenes. While today, a pre-dominant scheme for pre-training is next-token prediction via masking in both language Radford et al., 2019, vision He et al., 2021 and videos Tong et al., 2022, contrastive learning is

InfoNCE minimizes the expectation of

$$-\log \frac{\exp \psi(\mathbf{x}, \mathbf{x}^+)}{\sum_{\mathbf{x}^- \in N} \exp \psi(\mathbf{x}, \mathbf{x}^-)},$$

over triples $(\mathbf{x}, \mathbf{x}^+, N)$, in some formulations including the positive pair in the denominator.

still a state-of-the-art approach for multi-modal representation learning in models like contrastive image-language pre-training (CLIP; Radford et al., 2021).

Contrastive learning received popularity for identifiable representation learning. Time contrastive learning (TCL; Hyvarinen and Morioka, 2016) was shown to solve the independent component analysis (ICA) problem for stationary time-series, and permutation contrastive learning (PCL) could achieve a similar feat also for non-stationary time-series (Hyvarinen & Morioka, 2017). It was later shown that both approaches could be unified by introducing "auxiliary variables" used to condition the training process (Hyvarinen et al., 2019). Leveraging auxiliary variables[8] during contrastive learning is an interesting alternative (or rather, generalization) to using time-information, especially for scientific applications.

Dissertation outline

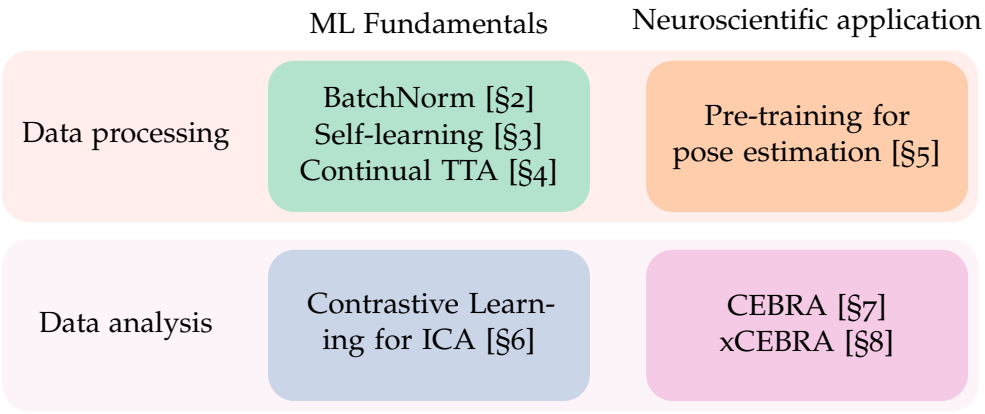


Figure 1.7: Overview of topics covered in this dissertation. We distinguish between methods for data processing and analysis as outlined in the introduction, and for both stages of the scientific discovery pipeline, investigate algorithms from two angles: ML fundamentals are chapters primarily written for machine learning audiences, but on problems relevant to neuroscientific inference. These insights can then be applied to neuroscience, as depicted in the right column.

In the light of the enormous potential of modern methodology in neuroscience, how can we speed up scientific discovery? This dissertation addresses the gaps in making data processing algorithms more robust and deployable in neuroscientific application scenarios, and proposes new statistical tools for the analysis of neuroscience datasets (and beyond). Figure 1.7 gives a birds-eye view on the categorization of covered topics, which I will discuss in more details in the following sections.

[8] As Hyvarinen et al. (2019) state, "Let us note that the existence of labels does not mean a nonlinear ICA model is not interesting, because our interest might not be in classifying the data using these labels, but rather in understanding the structure of the data [...]. In particular, with scientific data, the main goal is usually to understand its structure; if the labels correspond to different treatments, or experimental conditions, the classification problem in itself may not be of great interest."

Robust vision for scientific data processing.

Chapter 2 introduces an evaluation framework for robustness after test-time adaptation along with a simple yet effective method for boosting robustness scores of trained computer vision models. Building on this technique, Chapter 3 considers a family of optimization techniques that use self-learning to further improve test-time adaptation results. I show that test-time adaptation is highly effective at a range of tasks under static distribution shifts. While the aforementioned tasks are carried out in static evaluation scenarios, real-world algorithms often operate under changing environmental conditions which can yield collapse of the previously introduced algorithms. Chapter 4 proposes a new benchmark dataset for long-timescale evaluation scenarios, highlights limitations of existing test-time adaptation methods, and proposes a simple baseline method to overcome them. Computer vision is highly relevant for its application to pose estimation for digitizing poses of animals, e.g. during behavioral studies in neuroscience. Chapter 5 studies how transfer learning and adaptation techniques can be used on these tasks.

Neuroscientific data analysis using contrastive learning.

Moving from data acquisition to data analysis and modeling, Chapter 6 studies contrastive learning and builds a new connection between InfoNCE minimization and non-linear ICA. This allows for the design of new contrastive learning objectives with desirable statistical properties for identifiability. In neuroscience, such techniques are highly relevant for the post-processing steps of both behavioral data (e.g., acquired during pose estimation) and neural data (e.g., electrophysiology or calcium imaging data). Chapter 7 develops a generalized formulation of contrastive learning unifying self-supervised and supervised contrastive learning with discrete, continuous or mixed labels.

Closing the loop to infer and influence data collection.

Finally, we aim to close the loop and inform the data collection process through analysis mechanisms. Following analysis, we want to attribute descriptions about the underlying data structure back into the signal space, and onto brain areas, neurons, etc. Insight about areas and neurons involved in a computation allows to design interventions and form hypotheses for subsequent experiments. Interpretable machine learning effectively closes the loop from analysis back to hypothesis formation. For this purpose, in Chapter 8 I propose a technique for estimating attribution maps for data analysis algorithms using a new regularized contrastive learning objective.

Additional related work conducted

Some additional work, especially on applications of machine learning algorithms for data preprocessing was done besides this dissertation, but not included. For a primer

on pose estimation and their applications in neuroscience, see Mathis et al. (2020). For considerations on compositional generalization and object-centric representation learning, see Tangemann et al. (2023). For work on multi-animal pose estimation, see Lauer et al. (2022). For ongoing work towards animal and keypoint-agnostic pose estimation models, see Ye et al. (2023a). This includes the application of self-learning approaches for test-time adaptation.

2

Improving robustness against common corruptions by covariate shift adaptation

The following pages contain the postprint version of the published paper

> Steffen Schneider*, Evgenia Rusak*, Luisa Eck, Oliver Bringmann†, Wieland Brendel† and Matthias Bethge†. "Improving robustness against common corruptions by covariate shift adaptation." *Advances in Neural Information Processing Systems 33 (2020): 11539-11551.*

A short version of the paper was also previously selected for a contributed talk (top 5%) at the Workshop on Uncertainty & Robustness in Deep Learning (UDL) at ICML 2020[1].

Author contributions StS conceived the project and proposed the original formulation of batch norm adaptation, WB suggested the partial adaptation setting. StS and ER implemented and conducted the experiments with input from WB and MB. StS prepared visualizations and analysis with input from ER and WB. StS and LE conducted the theoretical analysis. ER, WB, StS and MB wrote the manuscript, all authors participated in reviewing and editing. *StS and ER contributed equally to the project. †WB, MB and OB contributed equally to advising the project.

[1] https://sites.google.com/view/udlworkshop2020

Summary

Today's state-of-the-art machine vision models are vulnerable to image corruptions like blurring or compression artefacts, limiting their performance in many real-world applications. We here argue that popular benchmarks to measure model robustness against common corruptions (like ImageNet-C) underestimate model robustness in many (but not all) application scenarios. The key insight is that in many scenarios, multiple unlabeled examples of the corruptions are available and can be used for unsupervised online adaptation. Replacing the activation statistics estimated by batch normalization on the training set with the statistics of the corrupted images consistently improves the robustness across 25 different popular computer vision models. Using the corrected statistics, ResNet-50 reaches 62.2% mCE on ImageNet-C compared to 76.7% without adaptation. With the more robust DeepAugment+AugMix model, we improve the state of the art achieved by a ResNet50 model up to date from 53.6% mCE to 45.4% mCE. Even adapting to a single sample improves robustness for the ResNet-50 and AugMix models, and 32 samples are sufficient to improve the current state of the art for a ResNet-50 architecture. We argue that results with adapted statistics should be included whenever reporting scores in corruption benchmarks and other out-of-distribution generalization settings.

Introduction

Deep neural networks (DNNs) are known to perform well in the independent and identically distributed (*i.i.d.*) setting when the test and training data are sampled from the same distribution. However, for many applications this assumption does not hold. In medical imaging, X-ray images or histology slides will differ from the training data if different acquisition systems are being used. In quality assessment, the images might differ from the training data if lighting conditions change or if dirt particles accumulate on the camera. Autonomous cars may face rare weather conditions like sandstorms or big hailstones. While human vision is quite robust to those deviations (Geirhos et al., 2018b), modern machine vision models are often sensitive to such image corruptions.

We argue that current evaluations of model robustness underestimate performance in many (but not all) real-world scenarios. So far, popular image corruption benchmarks like ImageNet-C [IN-C; Hendrycks and Dietterich, 2019a] focus only on ad hoc scenarios in which the tested model has zero prior knowledge about the corruptions it encounters during test time, even if it encounters the same corruption multiple times. In the example of medical images or quality assurance, the image corruptions do not change from sample to sample but are continuously present over a potentially large number of samples. Similarly, autonomous cars will face the same weather condition over a continuous stream of inputs during the same sand- or hailstorm. These (unlabeled) observations can allow recognition models to adapt to the change in the input distribution.

Such unsupervised adaptation mechanisms are studied in the field of domain adaptation (DA), which is concerned with adapting models trained on one domain (the source, here clean images) to another for which only unlabeled samples exist (the target, here the corrupted images). Tools and methods from domain adaptation are thus directly applicable to increase model robustness against common corruptions, but so far no results on popular benchmarks have been reported. The overall goal of this work is to encourage stronger interactions between the currently disjoint fields of domain adaptation and robustness towards common corruptions.

We here focus on one popular technique in DA, namely adapting batch normalization [BN; Ioffe and Szegedy, 2015] statistics (Cariucci et al., 2017; Li et al., 2017; Schneider et al., 2018). In computer vision, BN is a popular technique for speeding up training and is present in almost all current state-of-the-art image recognition models. BN estimates the statistics of activations for the training dataset and uses them to normalize intermediate activations in the network.

By design, activation statistics obtained during training time do not reflect the statistics of the test distribution when testing in out-of-distribution settings like corrupted images. We investigate and corroborate the hypothesis that high-level distributional shifts from clean to corrupted images largely manifest themselves in a difference of first and second order moments in the internal representations of a deep network, which can be mitigated by adapting BN statistics, i.e. by estimating the BN statistics on the corrupted images. We demonstrate that this simple adaptation alone can greatly increase recognition performance on corrupted images.

Our contributions can be summarized as follows:

- We suggest to augment current benchmarks for common corruptions with two additional performance metrics that measure robustness after partial and full unsupervised adaptation to the corrupted images.

- We draw connections to domain adaptation and show that even adapting to a single corrupted sample improves the baseline performance of a ResNet-50 model trained on IN from 76.7% mCE to 71.4%. Robustness increases with more samples for adaptation and converges to a mCE of 62.2%.

- We show that the robustness of a variety of vanilla models trained on ImageNet [IN; Deng et al., 2009a; Russakovsky et al., 2015] substantially increases after adaptation, sometimes approaching the current state-of-the-art performance on IN-C without adaptation.

- Similarly, we show that the robustness of state-of-the-art ResNet-50 models on IN-C consistently increases when adapted statistics are used. We surpass the best non-adapted model (52.3% mCE) by almost 7.0% points.

- We show results on several popular image datasets and discuss both the generality and limitations of our approach.

- We demonstrate that the performance degradation of a non-adapted model can be well predicted from the Wasserstein distance between the source and target statistics. We propose a simple theoretical model for bounding the Wasserstein distance based on the adaptation parameters.

Measuring robustness against common corruptions

The ImageNet-C benchmark (Hendrycks & Dietterich, 2019a) consists of 15 test corruptions and four hold-out corruptions which are applied with five different severity levels to the 50 000.0 test images of the ILSVRC2012 subset of ImageNet (Deng et al., 2009a). During evaluation, model responses are assumed to be conditioned only on single samples, and are not allowed to adapt to e.g. a batch of samples from the same corruption. We call this the ad hoc or non-adaptive scenario. The main performance metric on IN-C is the mean corruption error (mCE), which is obtained by normalizing the model's top-1 errors with the top-1 errors of AlexNet (Krizhevsky et al., 2012a) across the $C = 15$ test corruptions and $S = 5$ severities (cf. Hendrycks and Dietterich, 2019a):

$$\text{mCE(model)} = \frac{1}{C}\sum_{c=1}^{C}\frac{\sum_{s=1}^{S}\text{err}_{c,s}^{\text{model}}}{\sum_{s=1}^{S}\text{err}_{c,s}^{\text{AlexNet}}}. \tag{2.1}$$

Note that mCE reflects only one possible averaging scheme over the IN-C corruption types. We additionally report the overall top-1 accuracies and report results for all individual corruptions in the supplementary material and the project repository.

In many application scenarios, this ad hoc evaluation is too restrictive. Instead, often many unlabeled samples with similar corruptions are available, which can allow models to adapt to the shifted data distribution. To reflect such scenarios, we propose to also benchmark the robustness of adapted models. To this end, we split the 50 000.0 validation samples with the same corruption and severity into batches with n samples each and allow the model to condition its responses on the complete batch of images. We then compute mCE and top-1 accuracy in the usual way.

We consider three scenarios: In the *ad hoc* scenario, we set $n = 1$ which is the typically considered setting. In the *full adaptation* scenario, we set $n = 50\,000.0$, meaning the model may adapt to the full set of unlabeled samples with the same corruption type before evaluation. In the *partial adaptation* scenario, we set $n = 8$ to test how efficiently models can adapt to a relatively small number of unlabeled samples.

Correcting Batch Normalization statistics as a strong baseline for reducing covariate shift induced by common corruptions

We propose to use a well-known tool from domain adaptation—adapting batch normalization statistics (Cariucci et al., 2017; Li et al., 2017)—as a simple baseline to increase robustness against image corruptions in the adaptive evaluation scenarios. IN trained

models typically make use of batch normalization [BN; Ioffe and Szegedy, 2015] for faster convergence and improved stability during training. Within a BN layer, first and second order statistics μ_c, σ_c^2 of the activation tensors $\mathbf{z}_c$ are estimated across the spatial dimensions and samples for each feature map c. The activations are then normalized by subtracting the mean μ_c and dividing by σ_c^2. During training, μ_c and σ_c^2 are estimated *per batch*. During evaluation, μ_c and σ_c^2 are estimated *over the whole training dataset*, typically using exponential averaging (Paszke et al., 2017).

Using the BN statistics obtained during training for testing makes the model decisions deterministic but is also problematic if the input distribution changes. If the activation statistics μ_c, σ_c^2 change for samples from the test domain, then the activations of feature map c are no longer normalized to zero mean and unit variance, breaking a crucial assumption that all downstream layers depend on. Mathematically, this *covariate shift*[2] can be formalized as follows:

Definition 1 (Covariate Shift, cf. Schölkopf et al., 2012; Sugiyama and Kawanabe, 2012)**.** *There exists covariate shift between a source distribution with density $p_s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ and a target distribution with density $p_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, written as $p_s(\mathbf{x}, y) = p_s(\mathbf{x})p_s(y|\mathbf{x})$ and $p_t(\mathbf{x}, y) = p_t(\mathbf{x})p_t(y|\mathbf{x})$, if $p_s(y|\mathbf{x}) = p_t(y|\mathbf{x})$ and $p_s(\mathbf{x}) \neq p_t(\mathbf{x})$ where $y \in \mathcal{Y}$ denotes the class label.*

Removal of covariate shift. If covariate shift (Def. 1) only causes differences in the first and second order moments of the feature activations $\mathbf{z} = f(\mathbf{x})$, it can be removed by applying normalization:

$$p\left(\frac{f(\mathbf{x}) - \mathbb{E}_s[f(\mathbf{x})]}{\sqrt{\mathbb{V}_s[f(\mathbf{x})]}}\Big|\mathbf{x}\right) p_s(\mathbf{x}) \approx p\left(\frac{f(\mathbf{x}) - \mathbb{E}_t[f(\mathbf{x})]}{\sqrt{\mathbb{V}_t[f(\mathbf{x})]}}\Big|\mathbf{x}\right) p_t(\mathbf{x}). \tag{2.2}$$

Reducing the covariate shift in models with batch normalization is particularly straightforward: it suffices to estimate the BN statistics μ_t, σ_t^2 on (unlabeled) samples from the test data available for adaptation. If the number of available samples n is too small, the estimated statistics would be too unreliable. We therefore leverage the statistics μ_s, σ_s^2 already computed on the training dataset as a prior and infer the test statistics for each test batch as follows,

$$\bar{\mu} = \frac{N}{N+n}\mu_s + \frac{n}{N+n}\mu_t, \quad \bar{\sigma}^2 = \frac{N}{N+n}\sigma_s^2 + \frac{n}{N+n}\sigma_t^2. \tag{2.3}$$

The hyperparameter N controls the trade-off between source and estimated target statistics and has the intuitive interpretation of a *pseudo sample size* (p. 117, Bishop, 2006)

[2]Note that our notion of internal covariate shift differs from previous work (Ioffe & Szegedy, 2015; Santurkar et al., 2018): In *i.i.d.* training settings, Ioffe and Szegedy (2015) hypothesized that covariate shift introduced by changing lower layers in the network is reduced by BN, explaining the empirical success of the method. We do not provide evidence for this line of research in this work: Instead, we focus on the covariate shift introduced (by design) in datasets such as IN-C, and provide evidence for the hypothesis that high-level domain shifts in the input partly manifests in shifts and scaling of *internal* activations.

for samples from the training set. The case $N \to \infty$ ignores the test set statistics and is equivalent to the standard ad hoc scenario while $N = 0$ ignores the training statistics. Supported by empirical and theoretical results (see results section and appendix), we suggest using $N \in [8, 128]$ for practical applications with small $n < 32$.

Experimental Setup

Models. We consider a large range of models (cf. Table 2, Notes on the experimental setup, Full list of used models) and evaluate pre-trained variants of DenseNet (Huang et al., 2017), GoogLeNet (Szegedy et al., 2015), Inception and GoogLeNet (Szegedy et al., 2016), MNASnet (Tan et al., 2019), MobileNet (Sandler et al., 2018a), ResNet (He et al., 2016c), ResNeXt (Xie et al., 2017), ShuffleNet (Ma et al., 2018), VGG (Simonyan & Zisserman, 2015) and Wide Residual Network [WRN, Zagoruyko and Komodakis, 2016] from the `torchvision` library (Marcel & Rodriguez, 2010). All models are trained on the ILSVRC2012 subset of IN comprised of 1.2 million images in the training and a total of 1000.0 classes (Deng et al., 2009a; Russakovsky et al., 2015). We also consider a ResNeXt-101 variant pre-trained on a 3.5 billion image dataset and then fine-tuned on the IN training set (Mahajan et al., 2018). We evaluate 3 models from the SimCLRv2 framework (Chen et al., 2020c). We additionally evaluate the four leading methods from the ImageNet-C leaderboard, namely Stylized ImageNet training [SIN; Geirhos et al., 2019], adversarial noise training [ANT; Rusak et al., 2020] as well as a combination of ANT and SIN (Rusak et al., 2020), optimized data augmentation using AutoAugment [AugMix; Cubuk et al., 2019; Hendrycks et al., 2020b] and Assemble Net (Lee et al., 2020). For partial adaptation, we choose $N \in \{2^0, \cdots, 2^{10}\}$ and select the optimal value on the holdout corruption mCE.

Datasets. ImageNet-C [IN-C; Hendrycks and Dietterich, 2019a] is comprised of corrupted versions of the 50 000.0 images in the IN validation set. The dataset offers five severities per corruption type, for a total of 15 "test" and 4 "holdout" corruptions. ImageNet-A [IN-A; Hendrycks et al., 2019b] consists of unmodified real-world images which yield chance level classification performance in IN trained ResNet-50 models. ImageNet-V2 [IN-V2; Recht et al., 2020] aims to mimic the test distribution of IN, with slight differences in image selection strategies. ObjectNet [ON; Barbu et al., 2019] is a test set containing 50 000.0 images like IN organized in 313 object classes with 109 unambiguously overlapping IN classes. ImageNet-R [IN-R; Hendrycks et al., 2020a] contains 30 000.0 images with various artistic renditions of 200 classes of the original IN dataset. Additional information on the used models and datasets can be found in the appendix, "Notes on the experimental setup". For IN, we resize all images to 256×256px and take the center 224×224px crop. For IN-C, images are already cropped. We also center and re-scale the color values with $\mu_{RGB} = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$.

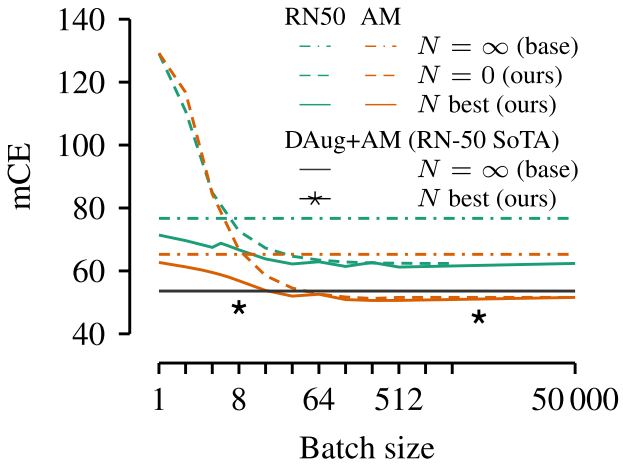


Figure 2.1: Sample size vs. performance tradeoff in terms of the mean corruption error (mCE) on IN-C for ResNet-50 and AugMix (AM). Black line corresponds to (non-adapted) ResNet50 state-of-the-art performance of DeepAug+AugMix.

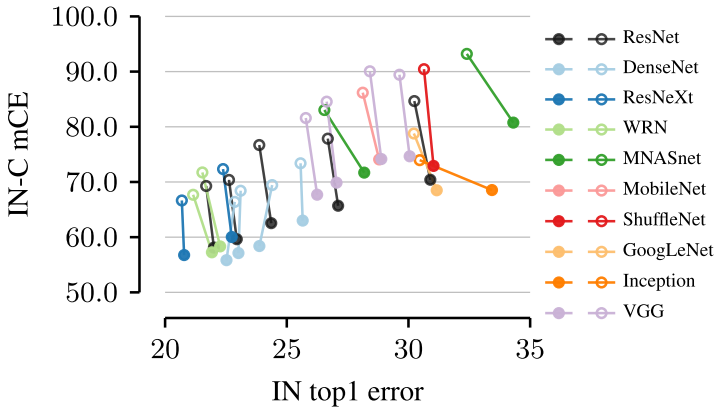


Figure 2.2: Across 25 model architectures in the `torchvision` library, the baseline mCE (○) improves with adaptation (●), often on the order of 10 points. Best viewed in color.

Results

Adaptation boosts robustness of a vanilla trained ResNet-50 model. We consider the pre-trained ResNet-50 architecture from the `torchvision` library and adapt the running mean and variance on all corruptions and severities of IN-C for different batch sizes. The results are displayed in Fig. 2.1 where different line styles of the green lines show the number of pseudo-samples N indicating the influence of the prior given by the training statistics. With $N = 16$, we see that even adapting to a single sample can suffice to increase robustness, suggesting that even the ad hoc evaluation scenario can benefit from adaptation. If the training statistics are not used as a prior ($N = 0$), then it takes around 8 samples to surpass the performance of the non-adapted baseline model (76.7% mCE). After around 16 to 32 samples, the performance quickly converges to 62.2% mCE, considerably improving the baseline result. These results highlight the

Table 2.1: Adaptation improves mCE (lower is better) and Top1 accuracy (higher is better) on IN-C for different models and surpasses the previous state of the art without adaptation. We consider $n = 8$ for partial adaptation.

	IN-C mCE (↘)				Top1 accuracy (↗)			
Model	w/o adapt	partial adapt	full adapt	Δ	w/o adapt	partial adapt	full adapt	Δ
Vanilla ResNet-50	76.7	65.0	62.2	(−14.5)	39.2	48.6	50.7	(+11.5)
SIN [3]	69.3	61.5	59.5	(−9.8)	45.2	51.6	53.1	(+7.9)
ANT [4]	63.4	56.1	53.6	(−9.8)	50.4	56.1	58.0	(+7.6)
ANT+SIN [5]	60.7	55.3	53.6	(−7.0)	52.6	56.8	58.0	(+5.4)
AugMix [AM][6]	65.3	55.4	51.0	(−14.3)	48.3	56.3	59.8	(+11.4)
Assemble Net [7]	52.3	–	50.1	(−1.2)	59.2	–	60.8	(+1.5)
DeepAug[8]	60.4	52.3	49.4	(−10.9)	52.6	59.0	61.2	(+8.6)
DeepAug+AM[9]	53.6	48.4	45.4	(−8.2)	58.1	62.2	64.5	(+6.4)
DeepAug+AM+RNXt101[10]	**44.5**	40.7	**38.0**	(−6.6)	**65.2**	68.2	**70.3**	(+5.1)

practical applicability of batch norm adaptation in basically all application scenarios, independent of the number of available test samples.

Adaptation consistently improves corruption robustness across IN trained models. To evaluate the interaction between architecture and BN adaptation, we evaluate all 25 pre-trained models in the `torchvision` package and visualize the results in Fig. 2.2. All models are evaluated with $N = 0$ and $n = 2000.0$. We group models into different families based on their architecture and observe consistent improvements in mCE for all of these families, typically on the order of 10% points. We observe that in both evaluation modes, DenseNets (Huang et al., 2017) exhibit higher corruption robustness despite having a comparable or even smaller number of trainable parameters than ResNets which are usually considered as the relevant baseline architecture. A take-away from this study is thus that model architecture alone plays a significant role for corruption robustness and the ResNet architecture might not be the optimal choice for practical applications.

Adaptation yields new state of the art on IN-C for robust models. We now investigate if BN adaptation also improves the most robust models on IN-C. The results are displayed in Table 2.1. All models are adapted using $n = 50\,000.0$ (vanilla) or $n = 4096.0$ (all other models) and $N = 0$. The performance of all models is considerably higher whenever the BN statistics are adapted. The DeepAugment+AugMix reaches a new state of the art on IN-C for a ResNet-50 architecture of 45.4% mCE. Evaluating the performance of AugMix over the number of samples for adaptation (Fig. 2.1, we find that as little as eight samples are sufficient to improve over AssembleNet (Lee et al., 2020), the current state-of-the-art ResNet-50 model on IN-C without adaptation. We have included additional results in the appendix, "Additional results".

Table 2.2: Improvements from adapting the BN parameters vanish for models trained with weakly supervised pre-training.

	IN-C mCE (↘)	
ResNeXt101	BN	BN+adapt
32x8d, IN	66.6	56.7 (−9.9)
32x8d, IG-3.5B	51.7	51.6 (−0.1)
32x48d, IG-3.5B	**45.7**	**47.3** (+1.6)

Analysis and Ablation Studies

Severity of covariate shift correlates with performance degradation. The relationship between the performance degradation on IN-C and the covariate shift suggests an unsupervised way of estimating the classification performance of a model on a new corruption. Taking the normalized Wasserstein distance (cf. appendix, "Distances and divergences for quantifying domain shift") between the statistics of the source and target domains[11] computed on all samples with the same corruption and severity and averaged across all network layers, we find a correlation with the top-1 error (Fig. 2.3 *i–iii*) of both non-adapted (*i*) and fully adapted model (*ii*) on IN-C corruptions. Within single corruption categories (noise, blur, weather, and digital), the relationship between top-1 error and Wasserstein distance is particularly striking: using linear regression, the top-1 accuracy of hold-out corruptions can be estimated with around 1–2% absolute mean deviation (cf. appendix, "Error prediction based on the Wasserstein distance") within a corruption, and with around 5–15% absolute mean deviation when the estimate is computed on the holdout corruption of each category (see Fig. 2.3, typically, a systematic offset remains). In Fig. 2.3*(iv–v)*, we display the Wasserstein distance across individual layers and observe that the covariate shift is particularly present in early and late downsampling layers of the ResNet-50.

Large scale pre-training alleviates the need for adaptation. Computer vision models based on the ResNeXt architecture (Xie et al., 2017) pretrained on a much larger dataset comprised of 3.5×10^9 Instagram images (IG-3.5B) achieve a 45.7% mCE on IN-C (Mahajan et al., 2018; Orhan, 2019). We re-evaluate these models with our proposed paradigm and summarize the results in Table 2.2. While we see improvements for the small model pre-trained on IN, these improvements vanish once the model is trained on the full IG-3.5B dataset. This observation also holds for the largest model, suggesting that training on very large datasets might alleviate the need for covariate shift adaptation.

Group Normalization and Fixup Initialization performs better than non-adapted batch norm models, but worse than batch norm with covariate shift adaptation. So far, we considered

[11] For computing the Wasserstein metric we make the simplifying assumption that the empirical mean and covariances fully parametrize the respective distributions.

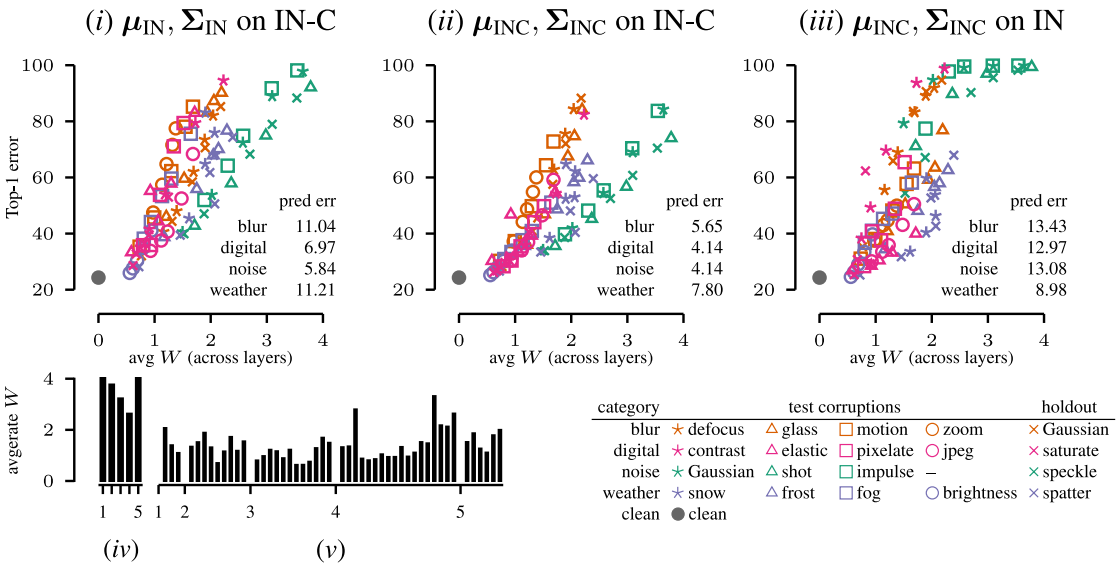


Figure 2.3: The Wasserstein metric between optimal source (IN) and target (IN-C) statistics correlates well with top-1 errors (*i*) of non-adapted models on IN-C, (*ii*) of adapted models on IN-C, indicating that even after reducing covariate shift, the metric is predictive of the remaining source–target mismatch (*iii*) IN-C adapted models on IN, the reverse case of (*i*). Holdout corruptions can be used to get a linear estimate on the prediction error of test corruptions (tables). We depict input and downsample (*iv*) as well as bottleneck layers (*v*) and notice the largest shift in early and late downsampling layers. The metric is either averaged across layers (*i–iii*) or across corruptions (*iv–v*).

Table 2.3: Fixup and GN trained models perform better than non-adapted BN models but worse than adapted BN models.

	IN-C mCE (↘)			
Model	Fixup	GN	BN	BN+adapt
ResNet-50	72.0	72.4	76.7	**62.2**
ResNet-101	68.2	67.6	69.0	**59.1**
ResNet-152	67.6	65.4	69.3	**58.0**

image classification models with BN layers and concluded that using training dataset statistics in BN generally degrades model performance in out-of-distribution evaluation settings. We now consider models trained without BN and study the impact on corruption robustness, similar to Galloway et al. (2019).

First, using Fixup initialization (Zhang et al., 2019) alleviates the need for BN layers. We train a ResNet-50 model on IN for 100 epochs to obtain a top-1 error of 24.2% and top-5 error of 7.6% (compared to 27.6% reported by Zhang et al. (2019) with shorter training, and the 23.9% obtained by our ResNet-50 baseline trained with BN). The model obtains an IN-C mCE of 72.0% compared to 76.7% mCE of the vanilla ResNet-50 model and 62.2% mCE of our adapted ResNet-50 model (cf. Table 2.3). Additionally, we train a ResNet-101 and a ResNet-152 with Fixup initialization with similar results. Second, GroupNorm [GN; Wu and He, 2018] has been proposed as a

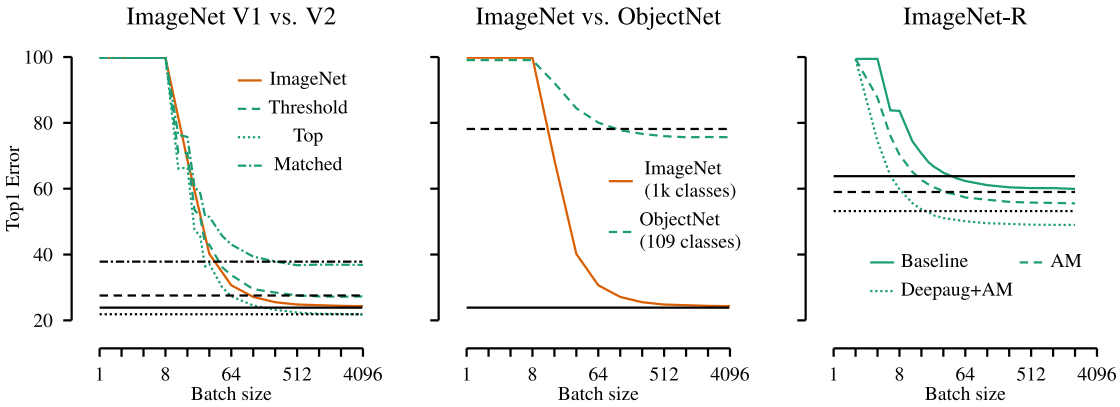


Figure 2.4: Batch size vs. performance trade-off for different natural image datasets with no covariate shift (IN, IN-V2), complex and shuffled covariate shift (ObjectNet), complex and systematic covariate shift (ImageNet-R). Straight black lines show baseline performance (no adaptation). ImageNet plotted for reference.

Table 2.4: GN and Fixup achieve the best results on ObjectNet (ON). After shuffling IN-C corruptions, BN adaptation does no longer decrease the error. Adaptation improves the performance of a vanilla ResNet50 on IN-R.

	ON		Mixed IN-C		IN-R
ResNet50	top-1	top-5	top-1	top-5	top-1
BN w/o adapt	78.2	60.9	61.1	40.8	63.8
BN w/ adapt	76.0	58.9	60.9	40.3	**59.9**
GroupNorm	**70.8**	**49.8**	57.3	36.0	61.2
Fixup	71.5	51.4	**56.8**	**35.4**	65.0

Table 2.5: Adaptation improves the performance (top-1 error) of robust models on IN-R (n=2048).

Model	base	adapt	Δ
ResNet50	63.8	59.9	-3.9
SIN	58.6	54.2	-4.4
ANT	61.0	58.0	-3.0
ANT+SIN	53.8	52.0	-1.8
AugMix (AM)	59.0	55.8	-3.2
DeepAug (DAug)	57.8	52.5	-5.3
DAug+AM	53.2	48.9	-4.3
DAug+AM+RNXt101	**47.9**	**44.0**	-3.9

batch-size independent normalization technique. We train a ResNet-50, a ResNet-101 and a ResNet-152 architecture for 100 epochs and evaluate them on IN-C and find results very similar to Fixup.

Results on other datasets: IN-A, IN-V2, ObjectNet, IN-R We use $N = 0$ and vary n in all ablation studies in this subsection. The technique does not work for the case of "natural adversarial examples" of IN-A (Hendrycks et al., 2019b) and the error rate stays above 99%, suggesting that the covariate shift introduced in IN-A by design is more severe compared to the covariate shift of IN-C and can not be corrected by merely calculating the correct BN statistics. We are not able to increase performance neither on IN nor on IN-V2, since in these datasets, no domain shift is present by design (see Fig. 2.4). For ON, the performance increases slightly when computing statistics on more than 64 samples. In Table 2.4 (first and second column), we observe that the GroupNorm and Fixup models perform better than our BN adaptation scheme: while there is a dataset shift in ON compared to IN, BN adaptation is only helpful for *systematic* shifts across

multiple inputs and this assumption is violated on ON. As a control experiment, we sample a dataset "Mixed IN-C" where we shuffle the corruptions and severities. In Table 2.4 (third and fourth column), we now observe that BN adaptation expectedly no longer improves performance. On IN-R, we achieve better results for the adapted model compared to the non-adapted model as well as the GroupNorm and Fixup models, see Table 2.4 (last column). Additionally, on IN-R, we decrease the top-1 error for a wide range of models through adaptation (see Table 2.5). For IN-R, we observe performance improvements for the vanilla trained ResNet50 when using a sample size of larger than 32 samples for calculating the statistics (Fig. 2.4, right-most plot).

A model for correcting covariate shift effects. We evaluate how the batch size for estimating the statistics at test time affects the performance on IN, IN-V2, ON and IN-R in Fig. 2.4. As expected, for IN the adaptation to test time statistics converges to the performance of the train time statistics in the limit of large batch sizes, see Fig. 2.4 middle. For IN-V2, we find similar results, see Fig. 2.4 left. This observation shows that (*i*) there is no systematic covariate shift between the IN train set and the IN-V2 validation set that could be corrected by using the correct statistics and (*ii*) is further evidence for the *i.i.d.* setting pursued by the authors of IN-V2. In case of ON (Fig. 2.4 right), we see slight improvements when using a batch size bigger than 128.

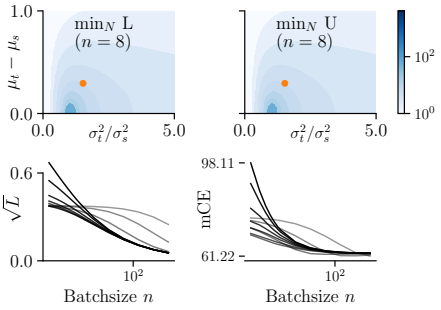


Figure 2.5: The bound suggests small optimal N for most parameters (i) and qualitatively explains our empirical observation (ii).

Choosing the number of pseudo-samples N offers an intuitive trade-off between estimating accurate target statistics (low N) and relying on the source statistics (large N). We propose a simple model to investigate optimal choices for N, disregarding all special structure of DNNs, and focusing on the statistical error introduced by estimating $\hat{\mu}_t$ and $\hat{\sigma}_t^2$ from a limited number of samples n. To this end, we estimate upper (U) and lower (L) bounds of the expected squared Wasserstein distance W_2^2 as a function of N and the covariate shift which provides good empirical fits between the estimated W and empirical performance for ResNet-50 for different N (Fig. 2.5; bottom row). Choosing N such that L or U are minimized (Fig. 2.5; example in top row) qualitatively matches the values we find, see appendix, "Analytical error model" for all details.

Proposition 1 (Bounds on the expected value of the Wasserstein distance between target and combined estimated target and source statistics)**.** *We denote the source statistics as μ_s, σ_s^2, the true target statistics as μ_t, σ_t^2 and the biased estimates of the target statistics as $\hat{\mu}_t, \hat{\sigma}_t^2$. For normalization, we take a convex combination of the source statistics and estimated target statistics as discussed in Eq. 2.3. At a confidence level $1 - \alpha$, the expectation value of the Wasserstein distance $W_2^2(\bar{\mu}, \bar{\sigma}, \mu_t, \sigma_t)$ between ideal and estimated target statistics w.r.t. to the distribution of sample mean $\hat{\mu}_t$ and sample variance $\hat{\sigma}_t^2$ is bounded from above and below with*

$L \leq \mathbb{E}[W_2^2] \leq U$, *where*

$$L = \left(\sigma_t - \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2}\right)^2 + \frac{N^2}{(N+n)^2}\left(\mu_t - \mu_s\right)^2 + \frac{n}{(N+n)^2}\sigma_t^2$$

$$U = L + \sigma_t^5 \frac{(n-1)}{2(N+n)^2}\left(\frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{1-\alpha/2,n-1}\sigma_t^2\right)^{-3/2}$$

The quantity $\chi^2_{1-\alpha/2,n-1}$ *denotes the left tail value of a chi square distribution with* $n-1$ *degrees of freedom, defined as* $P\left(X \leq \chi^2_{1-\alpha/2,n-1}\right) = \alpha/2$ *for* $X \sim \chi^2_{n-1}$. Proof: *See Appendix, Analytical error model.*

Related Work

The IN-C benchmark (Hendrycks & Dietterich, 2019a) has been extended to MNIST (Mu & Gilmer, 2019), several object detection datasets (Michaelis et al., 2019b) and image segmentation (Kamann & Rother, 2019) reflecting the interest of the robustness community. Most proposals for improving robustness involve special training protocols, requiring time and additional resources. This includes data augmentation like Gaussian noise (Ford et al., 2019), optimized mixtures of data augmentations in conjunction with a consistency loss (Hendrycks et al., 2020b), training on stylized images (Geirhos et al., 2019; Michaelis et al., 2019b; Mikołajczyk & Grochowski, 2018) or against adversarial noise distributions (Rusak et al., 2020). Other approaches tweak the architecture, e.g. by adding shift-equivariance with an anti-aliasing module, (Zhang, 2019) or assemble different training techniques (Lee et al., 2020).

Unsupervised domain adaptation (DA) is a form of transductive inference where additional information about the test dataset is used to adapt a model to the test distribution. Adapting feature statistics was proposed by Sun et al. (2017) and follow up work evaluated the performance of adapting BN parameters in unsupervised (Cariucci et al., 2017; Li et al., 2017) and supervised DA settings (Schneider et al., 2018). As an application example in medical imaging, Bug et al. (2017) show that adaptive normalization is useful for removing domain shifts on histopathological data. More involved methods for DA include self-supervised domain adaptation on single examples (Sun et al., 2019b) and pseudo-labeling French et al. (2017). Xie et al. (2020a) achieve the state of the art on IN-C with pseudo-labeling. In work concurrent to ours, Wang et al. (2020a) also show BN adaptation results on IN-C. They also perform experiments on CIFAR10-C and CIFAR100-C and explore other domain adaptation techniques.

Robustness scores obtained by adversarial training can be improved when separate BN or GroupNorm layers are used for clean and adversarial images (Xie & Yuille, 2020). The expressive power of adapting only affine BN parameters BN parameters was shown in multi-task (Rebuffi et al., 2017) and DA contexts (Schneider et al., 2018) and holds

even for fine-tuning randomly initialized ResNets (Frankle et al., 2020). Concurrent work shows additional evidence that BN adaptation yields increased performance on ImageNet-C (Nado et al., 2020).

Discussion and Conclusion

We showed that reducing covariate shift induced by common image corruptions improves the robustness of computer vision models trained with BN layers, typically by 10–15% points (mCE) on IN-C. Current state-of-the-art models on IN-C can benefit from adaptation, sometimes drastically like AugMix (−14.0% points mCE). This observation underlines that current benchmark results on IN-C underestimate the corruption robustness that can be reached in many application scenarios where additional (unlabeled) samples are available for adaptation.

Robustness against common corruptions improves even if models are adapted only to a single sample, suggesting that BN adaptation should always be used whenever we expect machine vision algorithms to encounter out-of-domain samples. Most further improvements can be reaped by adapting to 32 to 64 samples, after which additional improvements are minor.

Our empirical results suggest that the performance degradation on corrupted images can mostly be explained by the difference in feature-wise first and second order moments. While this might sound trivial, the performance could also degrade because models mostly extract features susceptible to common corruptions (Geirhos et al., 2020), which could not be fixed without substantially adapting the model weights. The fact that model robustness increases after correcting the BN statistics suggests that the features upon which the models rely on are still present in the corrupted images. The opposite is true in other out-of-domain datasets like IN-A or ObjectNet where our simple adaptation scheme does not substantially improve performance, suggesting that here the main problem is in the features that models have learned to use for prediction.

Batch Norm itself is not the reason why models are susceptible to common corruptions. While alternatives like Group Normalization and Fixup initialization slightly increase robustness, the adapted BN models are still substantially more robust. This suggests that non-BN models still experience an internal covariate shift on corrupted images, but one that is now absorbed by the model parameters instead of being exposed in the BN layers, making it harder to fix.

Large-scale pre-training on orders of magnitude more data (like IG-3.5B) can remove the first- and second-order covariate shift between clean and corrupted image samples, at least partially explaining why models trained with weakly supervised training (Mahajan et al., 2018) generalize so well to IN-C.

Current corruption benchmarks emphasize ad hoc scenarios and thus focus and bias future research efforts on these constraints. Unfortunately, the ad hoc scenario does not accurately reflect the information available in many machine vision applications like classifiers in medical computer vision or visual quality inspection algorithms,

which typically encounter a similar corruption continuously and could benefit from adaptation. This work is meant to spark more research in this direction by suggesting two suitable evaluation metrics—which we strongly suggest to include in all future evaluations on IN-C—as well as by highlighting the potential that even a fairly simple adaptation mechanism can have for increasing model robustness. We envision future work to also adopt and evaluate more powerful domain adaptation methods on IN-C and to develop new adaptation methods specifically designed to increase robustness against common corruptions.

Broader Impact

The primary goal of this paper is to increase the robustness of machine vision models against common corruptions and to spur further progress in this area. Increasing the robustness of machine vision systems can enhance their reliability and safety, which can potentially contribute to a large range of use cases including autonomous driving, manufacturing automation, surveillance systems, health care and others. Each of these uses may have a broad range of societal implications: autonomous driving can increase mobility of the elderly and enhance safety, but could also enable more autonomous weapon systems. Manufacturing automation can increase resource efficiency and reduce costs for goods, but may also increase societal tension through job losses or increase consumption and thus waste. Of particular concern (besides surveillance) is the use of generative vision models for spreading misinformation or for creating an information environment of uncertainty and mistrust.

We encourage further work to understand the limitations of machine vision models in out-of-distribution generalization settings. More robust models carry the potential risk of automation bias, i.e., an undue trust in vision models. However, even if models are robust to common corruptions, they might still quickly fail on slightly different perturbations like surface reflections. Understanding under what conditions model decisions can be deemed reliable or not is still an open research question that deserves further attention.

Acknowledgements

We thank Julian Bitterwolf, Roland S. Zimmermann, Lukas Schott, Mackenzie W. Mathis, Alexander Mathis, Asim Iqbal, David Klindt, Robert Geirhos, other members of the Bethge and Mathis labs and four anonymous reviewers for helpful suggestions for improving our manuscript and providing ideas for additional ablation studies. We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting E.R. and St.S.; St.S. acknowledges his membership in the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program. This work was supported by the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A), by the Deutsche Forschungsgemeinschaft

(DFG) in the priority program 1835 under grant BR2321/5-2 and by SFB 1233, Robust Vision: Inference Principles and Neural Mechanisms (TP3), project number: 276693517. The authors declare no conflicts of interests.

3

If your data distribution shifts, use self-learning

The following pages contain the postprint version of the published paper

> Evgenia Rusak*, Steffen Schneider*, George Pachitariu, Luisa Eck, Peter Vincent Gehler, Oliver Bringmann, Wieland Brendel†, and Matthias Bethge†. "If your data distribution shifts, use self-learning." *Transactions on Machine Learning Research (2022)*.

A short version of the paper was also previously selected for a contributed talk at the Weasul Workshop at ICML 2021.

Author Contributions The following list of contributions is reproduced from the respective section in the published paper. StS and ER conceived the project with suggestions from MB and WB. ER ran initial experiments. StS performed large scale experiments on ImageNet-C,R,A with support from PG, ER and WB. ER performed domain adaptation and CIFAR experiments with suggestions from StS. ER implemented all baselines that have been included as comparisons in the manuscript. GP ran DINO adaptation, WILDS, and self-ensembling experiments with suggestions from ER & StS. ER developed the ImageNet-D dataset with suggestions from StS and WB, and performed the large scale experiments on ImageNet-D. LE and StS developed the theoretical model, ER contributed the CIFAR-C experiments. WB, ER and StS wrote the initial manuscript with help from all authors. ER wrote an extensive related work section. ER reviewed and edited the manuscript in the course of the author-reviewer discussion period. *StS and ER contributed equally to the project. †WB and MB contributed equally to advising the project.

Summary

We demonstrate that self-learning techniques like entropy minimization and pseudo-labeling are simple and effective at improving performance of a deployed computer vision model under systematic domain shifts. We conduct a wide range of large-scale experiments and show consistent improvements irrespective of the model architecture, the pre-training technique or the type of distribution shift. At the same time, self-learning is simple to use in practice because it does not require knowledge or access to the original training data or scheme, is robust to hyperparameter choices, is straightforward to implement and requires only a few adaptation epochs. This makes self-learning techniques highly attractive for any practitioner who applies machine learning algorithms in the real world. We present state-of-the-art adaptation results on CIFAR10-C (8.5% error), ImageNet-C (22.0% mCE), ImageNet-R (17.4% error) and ImageNet-A (14.8% error), theoretically study the dynamics of self-supervised adaptation methods and propose a new classification dataset (ImageNet-D) which is challenging even with adaptation.

Introduction

Deep Neural Networks (DNNs) can reach human-level performance in complex cognitive tasks (Berner et al., 2019; Brown et al., 2020; He et al., 2016a) if the distribution of the test data is sufficiently similar to the training data. However, DNNs are known to struggle if the distribution of the test data is shifted relatively to the training data (Dodge & Karam, 2017; Geirhos et al., 2018b).

Two largely distinct communities aim to increase the performance of models under test-time distribution shifts: The *robustness community* generally considers ImageNet-scale datasets and evaluates models in an *ad-hoc* scenario. Models are trained on a clean source dataset like ImageNet (Deng et al., 2009a), using heavy data augmentation (Geirhos et al., 2019; Hendrycks et al., 2020a; Rusak et al., 2020) and/or large-scale pre-training (Mahajan et al., 2018; Xie et al., 2020a). The trained models are not adapted in any way to test-time distribution shifts. This evaluation scenario is relevant for applications in which very different distribution shifts are encountered in an unpredictable order, and hence misses out on the gains of adaptation to unlabeled samples of the target distribution.

The *unsupervised domain adaptation (UDA) community* often considers smaller-scale datasets and assumes that both the source and the (unlabeled) target dataset are known. Models are trained on both datasets, e.g., with an adversarial objective (Ganin et al., 2016; Hoffman et al., 2018; Tzeng et al., 2017), before evaluation on the target domain data. This evaluation scenario provides optimal conditions for adaptation, but the reliance on the source dataset makes UDA more computationally expensive, more impractical and prevents the use of pre-trained models for which the source dataset is unknown or simply too large. We refer the reader to Farahani et al. (2021) for a review

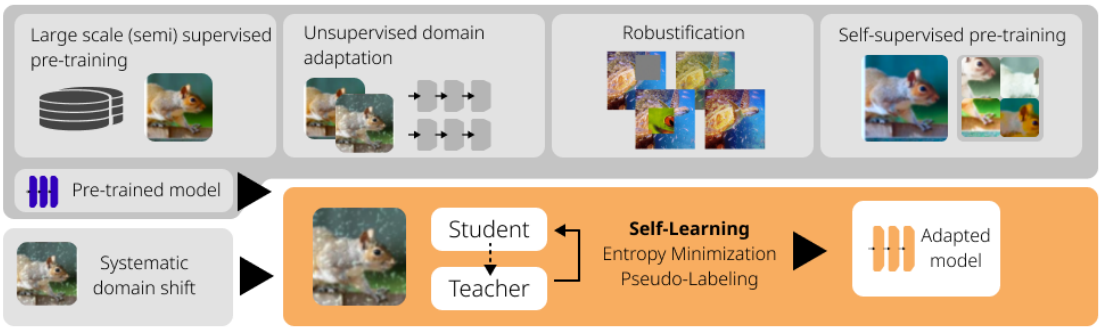


Figure 3.1: Robustness and adaptation to new datasets has traditionally been achieved by robust pre-training (with hand-selected/data-driven augmentation strategies, or additional data), unsupervised domain adaptation (with access to unlabeled samples from the test set), or, more recently, self-supervised learning methods. We show that on top of these different pre-training tasks, it is always possible (irrespective of architecture, model size or pre-training algorithm) to further adapt models to the target domain with simple self-learning techniques.

of UDA.

In this work, we consider the *source-free domain adaptation setting*, a middle ground between the classical ad-hoc robustness setting and UDA in which models can adapt to the target distribution but without using the source dataset (Kim et al., 2021; Kundu et al., 2020; Li et al., 2020a; Liang et al., 2020). This evaluation scenario is interesting for many practitioners and applications as an extension of the ad-hoc robustness scenario. It evaluates the possible performance of a *deployed* model on a systematic, unseen distribution shift at inference time: an embedded computer vision system in an autonomous car should adapt to changes without being trained on all available training data; an image-based quality control software may not necessarily open-source the images it has been trained on, but still has to be adapted to the lighting conditions at the operation location; a computer vision system in a hospital should perform robustly when tested on a scanner different from the one used for producing the training images—importantly, it might not be known at development time which scanner the vision system will be tested on, and it might be prohibited to share images from many hospitals to run UDA.

Can self-learning methods like *pseudo-labeling* and *entropy-minimization* also be used in this *source-free* domain adaptation setting?[1] To answer this question, we perform an extensive study of several self-learning variants, and find consistent and substantial gains in test-time performance across several robustness and out-of-domain benchmarks and a wide range of models and pre-training methods, including models trained with UDA methods that do not use self-learning, see Figure 3.1. We also find that self-learning outperforms state-of-the-art source-free domain adaptation methods, namely

[1]Self-learning was defined by Tsypkin (1968) to denote "learning when there is no external indication concerning the correctness of the response of the automatic system to the presented patterns". We opted to use this term to name the superset of pseudo-labeling, entropy minimization and self-supervised learning variants to highlight the fact these methods do not require ground truth labels for adaptation.

Test-Time Training which is based on a self-supervised auxiliary objective and continual training (Sun et al., 2019b), test-time entropy minimization (Wang et al., 2020a), Meta Test-Time Training (Bartler et al., 2022), and (gradient-free) BatchNorm adaptation (Nado et al., 2020; Schneider et al., 2020a), and can be improved with additional techniques such as diversity regularization (Mummadi et al., 2021). We perform a large number of ablations to study important design choices for self-learning methods in source-free domain adaptation. Furthermore, we show that a variant of pseudo-labeling with a robust loss function consistently outperforms entropy minimization on ImageNet-scale datasets.

We begin by positioning our work in the existing literature (§ 3) and proceed with an overview of various self-learning variants that have been applied over the past years, and propose a new technique for robust pseudo-labeling (§ 3). We then outline a rigorous experimental protocol that aims to highlight the strengths (and shortcomings) of various self-learning methods. We test various model architectures, with different pre-training schemes covering the most important models in unsupervised domain adaptation, robustness, and large-scale pre-training (§ 3). Using this protocol, we show the effectiveness of self-learning across architectures, models and pre-training schemes (§ 8). We proceed with an in-depth analysis of self-learning, both empirical (§ 3) and theoretical (§ 3). Since the outlined results on ImageNet-C (22.0% mCE), ImageNet-R (17.4% error) and ImageNet-A (14.8%) approach clean performance (11.6% error for our baseline), we propose ImageNet-D as a new benchmark, which we analyse in § 3. We conclude by proposing a set of best practices for evaluating test-time adaptation techniques in the future to ensure scientific rigor and to enable fair model and method comparisons (§ 3).

Related Work

Test-Time Adaptation The main question we ask in this work is whether self-learning methods such as entropy minimization and different variants of pseudo-labeling can improve the performance of models when adapted at test-time to data with a distribution shift relative to the training data. Our work is most similar to test-time entropy minimization (TENT; Wang et al., 2020a) since entropy minimization is one of our studied test-time adaptation techniques. The conceptual difference between our experiments and Wang et al. (2020a) is that Wang et al. (2020a) compare TENT to UDA while we argue that UDA can be regarded as a pretraining step, and self-learning can be used on top of any checkpoint pretrained with UDA. Wang et al. (2020a) study the effectiveness of entropy minimization across different models, datasets and tasks. We expand upon their experimental setup and show that entropy minimization is effective across a large range of model architectures (convolutional neural networks, Vision Transformers (Caron et al., 2021b; Dosovitskiy et al., 2021)), sizes (ResNet50 to EfficientNet-L2 (Tan & Le, 2019; Xie et al., 2020a)), and are orthogonal to robustification (e.g., DeepAugment (Hendrycks et al., 2020a)) and other pre-training schemes. Finally,

we perform a large hyperparameter study for test-time entropy minimization, and thereby further improve upon the results reported by Wang et al. (2020a).

We also compare our self-learning results to gradient-free adaptation of batch normalization (BN; Ioffe and Szegedy, 2015) statistics (BN adapt; Schneider et al., 2020a) who proposed re-estimating the BN statistics on the shifted data distribution.

In Test-Time Training (TTT; Sun et al., 2019b), the softmax cross-entropy loss is combined with a rotation prediction task (Gidaris et al., 2018) during pretraining on the source dataset. At test-time, the model is fine-tuned on the unlabeled test data with the self-supervised task. Sun et al. (2019b) report results when adapting to a single test example, and also for online adaptation where the model successively adapts to a stream of data, where the data can either come from the same or a gradually changing distribution. We added a detailed comparison to Sun et al. (2019b) in Appendix "Self-learning outperforms Test-Time Training (Sun et al., 2019b)". Liu et al. (2021) (TTT+++) replace the rotation prediction task with SimCLR (Chen et al., 2020a), and find this modification improves performance.

Eastwood et al. (2022) propose Feature Restoration where the approximate feature distribution under the target data is realigned with the feature distribution under source data. Eastwood et al. (2022) report results on CIFAR10 (among other datasets) for a ResNet18 architecture which allows us to include their method as a baseline. In a concurrent publication, Niu et al. (2022a) propose an anti-forgetting test-time adaptation method called EATA, which combine sample-efficient entropy minimization with anti-forgetting weight regularization. They report accuracy numbers on ImageNet-C for a ResNet50 which we include as a baseline. Mummadi et al. (2021) introduce a novel loss to stabilize entropy minimization by replacing the entropy by a non-saturating surrogate and a diversity regularizer based on batch-wise entropy maximization that prevents convergence to trivial collapsed solutions. To partially undo the distribution shift at test time, they additionally propose to add an input transformation module to the model. Mummadi et al. (2021) report results on the highest severity of ImageNet-C for a ResNet50 which we include as a comparison to our results. We note that Mummadi et al. (2021) constitute concurrent unpublished work.

Meta Test-Time Training (MT3; Bartler et al., 2022) combine meta-learning, self-supervision and test-time training to adapt a model trained on clean CIFAR10 (Krizhevsky, Hinton, et al., 2009) to CIFAR10-C (Hendrycks & Dietterich, 2019a). We added a detailed comparison to Bartler et al. (2022) in the Appendix, "Comparison to Meta Test-Time Training (Bartler et al., 2022)".

The following papers also consider the setting of test-time adaptation, but are not used as direct baselines in our work, because they study other datasets or tasks. Azimi et al. (2022) show performance improvements when using test-time adaptation on video data. They show adaptation results for the popular test-time adaptation techniques of BN adaptation (Schneider et al., 2020a), Test-Time Training (Sun et al., 2019b) and test-time entropy minimization (Wang et al., 2020a). MEMO (Zhang et al., 2021) maximizes the prediction consistency of different augmented copies regarding

a given test sample. We do not consider the single-sample adaptation setting, and comparing our self-learning techniques to MEMO is not a fair setting for MEMO; unsurprisingly, techniques benefiting from multiple samples such as TENT (Wang et al., 2020a) outperform MEMO. AdaContrast by Chen et al. (2022) combines pseudo-labeling with other techniques, such as self-supervised contrastive learning on the target domain, soft k-nearest neighbors voting to stabilize the pseudo-labels, as well as consistency and diversity regularization. A direct comparison with their results is difficult because they evaluate on VISDA-C and DomainNet, and so we would need to train our methods on either of the datasets and perform a full hyperparameter search for a fair comparison. We do have one point of comparison: in their paper, Chen et al. (2022) perform better than TENT (Wang et al., 2020a). However, we note that Chen et al. (2022) used the default hyperparameters of TENT and did not perform hyperparameter tuning on the new dataset, thus, we would expect the performance of properly tuned TENT to be better than reported in the paper. We expect the additional changes of AdaContrast to further improve upon our simple self-learning baselines. Our work is conceptually similar to virtual adversarial domain adaptation in the fine-tuning phase of DIRT-T (Shu et al., 2018). In contrast to DIRT-T, our objective is simpler and we scale the approach to considerably larger datasets on ImageNet scale. Iwasawa and Matsuo (2021) propose a test-time adaptation algorithm for the task of domain generalization based on computing the distance of each test sample and pseudo-prototypes for each class. Kumar et al. (2020) study the setting of self-learning for gradual domain adaptation. They find that self-learning works better if the data distribution changes slowly. The gradual domain adaptation setting differs from ours: instead of a gradual shift over time, we focus on a fixed shift at test time.

Self-learning for domain adaptation Xie et al. (2020b) introduce "In-N-Out" which uses auxiliary information to boost both in- and out-of-distribution performance. AdaMatch (Berthelot et al., 2021) builds upon FixMatch (Sohn et al., 2020) and can be used for the tasks of unsupervised domain adaptation, semi-supervised learning and semi-supervised domain adaptation as a general-purpose algorithm. Prabhu et al. (2021) propose SENTRY, an algorithm based on judging the predictive consistency of samples from the target domain under different image transformations. Zou et al. (2019) show that different types of confidence regularization can improve the performance of self-learning. A theoretically motivated framework for self-learning in domain adaptation based on consistency regularization has been proposed by Wei et al. (2020) and then extended by Cai et al. (2021).

The main differences from these works to ours are that they 1) utilize both source and target data during training (i.e., the classical UDA setup) whereas we only require access to unlabeled target data (source-free setup), 2) train their models from scratch whereas we adapt pretrained checkpoints to the unlabeled target data, and 3) are oftentimes more complicated (also in terms of the number of hyperparameters) than our approach due to using more than one term in the objective function. We would

like to highlight that utilizing source data should always result in better performance compared to not using source data. Our contribution is to show that self-learning can still be very beneficial with a small compute budget and no access to source data. Our setup targets "deployed systems", e.g., a self-driving car or a detection algorithm in a production line which adapts to the distribution shift "on-the-fly" and cannot (or should not) be retrained from scratch for every new domain shift.

Model selection Gulrajani and Lopez-Paz (2021) show that model selection for hyperparameter tuning is non-trivial for the task of domain generalization, and propose model selection criteria under which models should be selected for this task. Following their spirit, we identify a model selection criterion for test-time adaptation, and rigorously use it in all our experiments. We outperform state-of-the-art techniques which did not disclose their hyperparameter selection protocols.

Self-learning for Test-Time Adaptation

Tsypkin (1968) defines self-learning as "learning when there is no external indication concerning the correctness of the response of the automatic system to the presented patterns", and thus, we use this term as a superset of different variants of pseudo-labeling and entropy minimization to highlight that these methods can be used for adaptation to unlabeled data. Different versions of self-learning have been used in both unsupervised domain adaptation (French et al., 2017; Shu et al., 2018), self-supervised representation learning (Caron et al., 2021b), and in semi-supervised learning (Xie et al., 2020a). In a typical self-learning setting, a *teacher* network $\mathbf{f}^t$ trained on the source domain predicts labels on the target domain. Then, a *student* model $\mathbf{f}^s$ is fine-tuned on the predicted labels.

In the following, let $\mathbf{f}^t(\mathbf{x})$ denote the logits for sample $\mathbf{x}$ and let $p^t(j|\mathbf{x}) \equiv \sigma_j(\mathbf{f}^t(\mathbf{x}))$ denote the probability for class j obtained from a softmax function $\sigma_j(\cdot)$. Similarly, $\mathbf{f}^s(\mathbf{x})$ and $p^s(j|\mathbf{x})$ denote the logits and probabilities for the student model $\mathbf{f}^s$. For all techniques, one can optionally only admit samples where the probability $\max_j p^t(j|\mathbf{x})$ exceeds some threshold. We consider three popular variants of self-learning: Pseudo-labeling with hard or soft labels, as well as entropy minimization.

Hard Pseudo-Labeling (Galstyan & Cohen, 2007; Lee, 2013a). We generate labels using the teacher and train the student on pseudo-labels i using the softmax cross-entropy loss,

$$\ell_H(\mathbf{x}) := -\log p^s(i|\mathbf{x}), \quad i = \operatorname{argmax}_j p^t(j|\mathbf{x}) \tag{3.1}$$

Usually, only samples with a confidence above a certain threshold are considered for training the student. We test several thresholds but note that thresholding means discarding a potentially large portion of the data which leads to a performance decrease in itself. The teacher is updated after each epoch.

Soft Pseudo-Labeling (Galstyan & Cohen, 2007; Lee, 2013a). In contrast to the hard pseudo-labeling variant, we here train the student on class probabilities predicted by the teacher,

$$\ell_S(\mathbf{x}) := -\sum_j p^t(j|\mathbf{x}) \log p^s(j|\mathbf{x}). \tag{3.2}$$

Soft pseudo-labeling is typically not used in conjunction with thresholding, since it already incorporates the certainty of the model. The teacher is updated after each epoch.

*Entropy Minimization (*ENT*; Grandvalet and Bengio, 2004; Wang et al., 2020a).* This variant is similar to soft pseudo-labeling, but we no longer differentiate between a teacher and student network. It corresponds to an "instantaneous" update of the teacher. The training objective becomes

$$\ell_E(\mathbf{x}) := -\sum_j p^s(j|\mathbf{x}) \log p^s(j|\mathbf{x}). \tag{3.3}$$

Intuitively, self-learning with entropy minimization leads to a sharpening of the output distribution for each sample, making the model more confident in its predictions.

*Robust Pseudo-Labeling (*RPL*).* Virtually all introduced self-learning variants use the softmax cross-entropy classification objective. However, the softmax cross-entropy loss has been shown to be sensitive to label noise (Zhang et al., 2017; Zhang & Sabuncu, 2018). In the setting of domain adaptation, inaccuracies in the teacher predictions and, thus, the labels for the student, are inescapable, with severe repercussions for training stability and hyperparameter sensitivity as we show in the results.

As a straight-forward solution to this problem, we propose to replace the cross-entropy loss by a robust classification loss designed to withstand certain amounts of label noise (Ghosh et al., 2017; Shu et al., 2020; Song et al., 2020a; Zhang & Sabuncu, 2018). A popular candidate is the *Generalized Cross Entropy (GCE)* loss which combines the noise-tolerant Mean Absolute Error (MAE) loss (Ghosh et al., 2017) with the CE loss. We only consider the hard labels and use the robust GCE loss as the training loss for the student,

$$i = \text{argmax}_j\, p^t(j|\mathbf{x}), \quad \ell_{GCE}(\mathbf{x}, i) := q^{-1}(1 - p^s(i|\mathbf{x})^q), \tag{3.4}$$

with $q \in (0, 1]$. For the limit case $q \to 0$, the GCE loss approaches the CE loss and for $q = 1$, the GCE loss is the MAE loss (Zhang & Sabuncu, 2018). We test updating the teacher both after every update step of the student (RPL) and once per epoch (RPL$^{\text{ep}}$).

Adaptation parameters. Following Wang et al. (2020a), we only adapt the affine scale and shift parameters γ and β following the batch normalization layers (Ioffe & Szegedy,

2015) in most of our experiments. We verify that this type of adaptation works better than full model adaptation for large models in an ablation study in Section 3.

Additional regularization in self-learning Different regularization terms have been proposed as a means to stabilize entropy minimization. Niu et al. (2022a) propose an anti-forgetting weight regularization term, Chen et al. (2022), Liang et al. (2020), and Mummadi et al. (2021) add a diversity regularizer, and Chen et al. (2022) use an additional consistency regularizer. These methods show improved performance with these regularization terms over simple entropy minimization, but also introduce additional hyperparameters, the tuning of which significantly increases compute requirements. In this work, we do not experiment with additional regularization, as the main point of our analysis is to show that pure self-learning is effective at improving the performance over the unadapted model across model architectures/sizes and pre-training schemes. For practitioners, we note that regularization terms can further improve the performance if the new hyperparameters are tuned properly.

Experiment design

Datasets. ImageNet-C (IN-C; Hendrycks and Dietterich, 2019a) contains corrupted versions of the 50 000 images in the ImageNet validation set. There are fifteen test and four hold-out corruptions, and there are five severity levels for each corruption. The established metric to report model performance on IN-C is the mean Corruption Error (mCE) where the error is normalized by the AlexNet error, and averaged over all corruptions and severity levels, see Eq. B.11, Appendix, "Definition of the mean Corruption Error (mCE)". ImageNet-R (IN-R; Hendrycks et al., 2020a) contains 30 000 images with artistic renditions of 200 classes of the ImageNet dataset. ImageNet-A (IN-A; Hendrycks et al., 2019b) is composed of 7500 unmodified real-world images on which standard ImageNet-trained ResNet50 (He et al., 2016c) models yield chance level performance. CIFAR10 (Krizhevsky, Hinton, et al., 2009) and STL10 (Coates et al., 2011) are small-scale image recognition datasets with 10 classes each, and training sets of 50 000/5000 images and test sets of 10 000/8000 images, respectively. The digit datasets MNIST (Deng, 2012) and MNIST-M (Ganin et al., 2016) both have 60 000 training and 10 000 test images.

Hyperparameters. The different self-learning variants have a range of hyperparameters such as the learning rate or the stopping criterion. Our goal is to give a realistic estimation on the performance to be expected in practice. To this end, we optimize hyperparameters for each variant of pseudo-labeling on a hold-out set of IN-C that contains four types of image corruptions ("speckle noise", "Gaussian blur", "saturate" and "spatter") with five different strengths each, following the procedure suggested in Hendrycks and Dietterich (2019a). We refer to the hold-out set of IN-C as our *dev* set. On the small-scale datasets, we use the hold-out set of CIFAR10-C for hyperparameter

Table 3.1: Self-learning decreases the error on ImageNet-scale robustness datasets. Robust pseudo-labeling generally outperforms entropy minimization.

mCE [%] on IN-C test (↘)	number of parameters	w/o adapt	w/ adapt (Δ) RPL	w/ adapt (Δ) ENT
ResNet50 vanilla[1]	2.6×10^7	76.7	50.5 (-26.2)	51.6 (-25.1)
ResNet50 DAug+AM [2]	2.6×10^7	53.6	41.7 (-11.9)	42.6 (-11.0)
DenseNet161 vanilla [3]	2.8×10^7	66.4	47.0 (-19.4)	47.7 (-18.7)
ResNeXt101$_{32\times8d}$ vanilla[4]	8.8×10^7	66.6	43.2 (-23.4)	44.3 (-22.3)
ResNeXt101$_{32\times8d}$ DAug+AM[2]	8.8×10^7	44.5	34.8 (-9.7)	35.5 (-9.0)
ResNeXt101$_{32\times8d}$ IG-3.5B[5]	8.8×10^7	51.7	40.9 (-10.8)	40.8 (-10.9)
EfficientNet-L2 Noisy Student[6]	4.8×10^8	28.3	**22.0** (-6.3)	23.0 (-5.3)
top1 error [%] on IN-R (↘)				
ResNet50 vanilla[1]	2.6×10^7	63.8	54.1 (-9.7)	56.1 (-7.7)
EfficientNet-L2 Noisy Student[6]	4.8×10^8	23.5	**17.4** (-6.1)	19.7 (-3.8)
top1 error [%] on ImageNet-A (↘)				
EfficientNet-L2 Noisy Student[6]	4.8×10^8	16.5	**14.8** (-1.7)	15.5 (-1.0)

[1]He et al., 2016c [2]Hendrycks et al., 2020a [3]Huang et al., 2017 [4]Xie et al., 2017 [5]Mahajan et al., 2018 [6]Xie et al., 2020a

tuning. On all other datasets, we use the hyperparameters obtained on the hold-out sets of IN-C (for large-scale datasets) or CIFAR10-C (on small-scale datasets).

Models for ImageNet-scale datasets. We consider five popular model architectures: ResNet50 (He et al., 2016c), DenseNet161 (Huang et al., 2017), ResNeXt101 (Xie et al., 2017), EfficientNet-L2 (Tan & Le, 2019), and the Vision Transformer (ViT; Dosovitskiy et al., 2021) (see the Appendix, "Details on all hyperparameters we tested for different models" for details on the used models). For ResNet50, DenseNet and ResNeXt101, we include a simple *vanilla* version trained on ImageNet only. For ResNet50 and ResNeXt101, we additionally include a state-of-the-art robust version trained with DeepAugment and Augmix (DAug+AM; Hendrycks et al., 2020a)[2]. For the ResNeXt model, we also include a version that was trained on 3.5 billion weakly labeled images (IG-3.5B; Mahajan et al., 2018). For EfficientNet-L2 we select the current state of the art on IN-C which was trained on 300 million images from JFT-300M (Chollet, 2017; Hinton et al., 2014) using a noisy student-teacher protocol (Xie et al., 2020a). Finally, for the ViT, we use the model pretrained with DINO (Caron et al., 2021b). We validate the ImageNet and IN-C performance of all considered models and match the originally reported scores (Schneider et al., 2020a). For EfficientNet-L2, we match ImageNet top-1 accuracy up to 0.1% points, and IN-C up to 0.6% points mCE.

Models for CIFAR10/ MNIST-scale datasets. For CIFAR10-C experiments, we use three WideResNets (WRN; Zagoruyko and Komodakis, 2016): the first one is trained on

[2]see leaderboard at github.com/hendrycks/robustness

clean CIFAR10 and has a depth of 28 and a width of 10, the second one is trained with CIFAR10 with the AugMix protocol (Hendrycks et al., 2020b) and has a depth of 40 and a width of 2, and the third one has a depth of 26 layers, and is pre-trained on clean CIFAR10 using the default training code from https://github.com/kuangliu/pytorch-cifar. We used this code-base to also train the ResNet18 and the ResNet50 models on CIFAR10. The remaining small-scale models are trained with UDA methods. We propose to regard any UDA method which requires joint training with source and target data as a pre-training step, similar to regular pre-training on ImageNet, and use self-learning on top of the final checkpoint. We consider two popular UDA methods: self-supervised domain adaptation (UDA-SS; Sun et al., 2019a) and Domain-Adversarial Training of Neural Networks (DANN; Ganin et al., 2016). In UDA-SS, the authors seek to align the representations of both domains by performing an auxiliary self-supervised task on both domains simultaneously. In all UDA-SS experiments, we use a WideResNet with a depth of 26 and a width of 16. In DANN, the authors learn a domain-invariant embedding by optimizing a minimax objective. For all DANN experiments except for MNIST→MNIST-M, we use the same WRN architecture as above. For the MNIST→MNIST-M experiment, the training with the larger model diverged and we used a smaller WideResNet version with a width of 2. We note that DANN training involves optimizing a minimax objective and is generally harder to tune.

Self-learning universally improves models

Self-learning is a powerful learning scheme, and in the following section we show that it allows to perform test-time adaptation on robustified models, models obtained with large-scale pre-training, as well as (already) domain adapted models across a wide range of datasets and distribution shifts. Our main results on large-scale and small-scale datasets are shown in Tables 3.1 and 3.2. These summary tables show final results, and all experiments use the hyperparameters we determined separately on the dev set. The self-learning loss function, i.e. soft- or hard-pseudo-labeling / entropy minimization / robust pseudo-labeling, is a hyperparameter itself, and thus, in Tables 3.1 and 3.2, we show the overall best results. Results for the other loss functions can be found in Section 3 and in the Appendix, "Detailed and additional Results on IN-C".

Self-learning successfully adapts ImageNet-scale models across different model architectures on IN-C, IN-A and IN-R (Table 3.1) . We adapt the vanilla ResNet50, ResNeXt101 and DenseNet161 models to IN-C and decrease the mCE by over 19 percent points in all models. Further, self-learning works for models irrespective of their size: Self-learning substantially improves the performance of the ResNet50 and the ResNext101 trained with DAug+AM, on IN-C by 11.9 and 9.7 percent points, respectively. Finally, we further improve the current state of the art model on IN-C—the EfficientNet-L2 Noisy Student model—and report a new state-of-the-art result of 22% mCE (which corresponds to a

Table 3.2: Self-learning decreases the error on small-scale datasets, for models pre-trained using data augmentation and unsupervised domain adaptation. Entropy minimization outperforms robust pseudo-labeling.

top1 error [%] on CIFAR10-C (↘)	parameters		RPL	ENT
WRN-28-10 vanilla (Zagoruyko & Komodakis, 2016)	3.6×10^7	26.5	13.7 (-12.8)	13.3 (-13.2)
WRN-40-2 AM (Hendrycks et al., 2020b)	2.2×10^6	11.2	9.0 (-2.2)	8.5 (-2.7)
WRN-26-1-GN (Bartler et al., 2022)	1.5×10^6	18.6	18.0 (-0.6)	18.4 (0.2)
WRN-26-1-BN (Zagoruyko & Komodakis, 2016)	1.5×10^6	25.8	15.1 (-10.7)	13.1 (-12.7)
WRN-26-16 vanilla (Zagoruyko & Komodakis, 2016)	9.3×10^7	24.2	11.8 (-12.4)	11.2 (-13.0)
WRN-26-16 UDA-SS (Sun et al., 2019a)	9.3×10^7	27.7	18.2 (-9.5)	16.7 (-11.0)
WRN-26-16 DANN (Ganin et al., 2016)	9.3×10^7	29.7	28.6 (-1.1)	28.5 (-1.2)
UDA CIFAR10→STL10, top1 error on target [%](↘)				
WRN-26-16 UDA-SS (Sun et al., 2019a)	9.3×10^7	28.7	22.9 (-5.8)	21.8 (-6.9)
WRN-26-16 DANN (Ganin et al., 2016)	9.3×10^7	25.0	24.0 (-1.0)	23.9 (-1.1)
UDA MNIST→MNIST-M, top1 error on target [%](↘)				
WRN-26-16 UDA-SS (Sun et al., 2019a)	9.3×10^7	4.8	2.4 (-2.4)	2.0 (-2.8)
WRN-26-2 DANN (Ganin et al., 2016)	1.5×10^6	11.4	5.2 (-6.2)	5.1 (-6.3)

top1 error of 17.1%) on this benchmark with test-time adaptation (compared to 28% mCE without adaptation).

Self-learning is not limited to the distribution shifts in IN-C like compression artefacts or blur. On IN-R, a dataset with renditions, self-learning improves both the vanilla ResNet50 and the EfficientNet-L2 model, the latter of which improves from 23.5% to a new state of the art of 17.4% top-1 error. For a vanilla ResNet50, we improve the top-1 error from 63.8% (Hendrycks et al., 2020a) to 54.1%. On IN-A, adapting the EfficientNet-L2 model using self-learning decreases the top-1 error from 16.5% (Xie et al., 2020a) to 14.8% top-1 error, again constituting a new state of the art with test-time adaptation on this dataset. Self-learning can also be used in an online adaptation setting, where the model continually adapts to new samples on IN-C in Fig. B.5(i) or IN-R Fig. B.5(ii), Appendix B.

Adapting a ResNet50 on IN-A with RPL increases the error from 0% to 0.13% (chance level: 0.1%). Thus, an unadapted ResNet50 has 0% accuracy on IN-A by design and this error "is increased" to chance-level with self-learning. Since all labels on ImageNet-A are wrong by design, predicting wrong labels as pseudo-labels does not lead to improvements beyond restoring chance-level performance.

The finding that self-learning can be effective across model architectures has also been made by Wang et al. (2020a) who show improved adaptation performance on CIFAR100-C for architectures based on self-attention (Zhao et al., 2020a) and equilibrium solving (Bai et al., 2020), and also by Mummadi et al. (2021) who showed adaptation results for TENT and TENT+ which combines entropy minimization with a diversity regularizer for a DenseNet121 (Huang et al., 2017), a MobileNetV2 (Sandler et al., 2018a), a ResNeXt50 (Xie et al., 2017), and a robust model trained with DAug+AM on IN-C and IN-R.

The improvements of self-learning are very stable: In Table B.15, Appendix B, we

show the averaged results across three different seeds for a ResNet50 model adapted with ENT and RPL. The unbiased std is roughly two orders of magnitude lower than any improvement we report in our paper, showcasing the robustness of our results to random initialization.

Self-learning improves robustified and domain adapted models on small-scale datasets (Table 3.2). We test common domain adaptation techniques like DANN (Ganin et al., 2016) and UDA-SS (Sun et al., 2019a), and show that self-learning is effective at further tuning such models to the target domain. We suggest to view unsupervised source/target domain adaptation as a step comparable to pre-training under corruptions, rather than an adaptation technique specifically tuned to the target set—indeed, we can achieve error rates using, e.g., DANN + target adaptation previously only possible with source/target based pseudo-labeling, across different common domain adaptation benchmarks.

For the UDA-SS experiments, we additionally trained a vanilla version with the same architecture using the widely used training code for CIFAR10 at https://github.com/kuangliu/pytorch-cifar (1.9k forks, 4.8k stars), and find that the vanilla trained model performs better both with and without adaptation compared to the UDA-SS model. We think that the UDA-SS model would need hyperparameter tuning; we did not perform any tuning for this model, especially because the authors provided scripts with hyperparameters they found to be optimal for different setups. In addition, the clean accuracy of the vanilla model on CIFAR10 (96.5%) is much higher than the average clean accuracy of the UDA-SS model (82.6%), which may explain or imply generally higher robustness under distribution shift (Miller et al., 2021). The finding that self-learning is more effective in the vanilla model compared to the UDA-SS model points towards the hypothesis that the network weights of the vanilla model trained on the source distribution are sufficiently general and can be tuned successfully using only the affine BN statistics, while the weights of the UDA-SS model are already co-adapted to both the source and the target distribution, and thus, self-learning is less effective.

Self-learning also decreases the error on CIFAR10-C of the Wide ResNet model trained with AugMix (AM, Hendrycks et al., 2020b) and reaches a new state of the art on CIFAR10-C of 8.5% top1 error with test-time adaptation.

Table 3.3: Unlike batch norm adaptation, self-learning adapts large-scale models trained on external data.

mCE, test [%] (↘)	w/o adapt	BN adapt	RPL
ResNeXt101 vanilla	66.6	56.8	43.2
ResNeXt101 IG-3.5B	51.7	51.8	**40.9**

Self-learning also improves large pre-trained models (Table 3.3). Unlike BatchNorm adaptation (Schneider et al., 2020a), we show that self-learning transfers well to models

Table 3.4: Self-learning outperforms other test-time-adaptation techniques on IN-C.

mCE [%] on IN-C test (↘)	w/o adapt	BN Adapt	TENT	EATA(lifelong)	RPL	ENT
ResNet50	76.7	62.2	53.5	51.2	**50.5**	51.6

pre-trained on a large amount of unlabeled data: self-learning decreases the mCE on IN-C of the ResNeXt101 trained on 3.5 billion weakly labeled samples (IG-3.5B, Mahajan et al., 2018) from 51.7% to 40.9%.

Self-learning outperforms other test-time adaptation techniques on IN-C (Tables 3.4 and 3.5). The main point of our paper is showing that self-learning is effective across datasets, model sizes and pretraining methods. Here, we analyze whether our simple techniques can compete with other state-of-the-art adaptation methods. Overall, we find that self-learning outperforms several state-of-the-art techniques, but underperforms in some cases, especially when self-learning is combined with other techniques.

By rigorous and fair model selection, we are able to improve upon TENT (Wang et al., 2020a), and find that RPL performs better than entropy minimization on IN-C. We also compare to BN adaptation (Schneider et al., 2020a), and find that 1) self-learning further improves the performance upon BN adapt, and 2) self-learning improves performance of a model pretrained on a large amount of data (Mahajan et al., 2018) (Table 3.3) which is a setting where BN adapt failed.

ENT and RPL outperform the recently published EATA method (Niu et al., 2022a). In EATA, high entropy samples are excluded from optimization, and a regularization term is added to prevent the model from forgetting the source distribution. EATA requires tuning of two additional parameters: the threshold for high entropy samples to be discarded and the regularization trade-off parameter β.

RPL, ENT and simple hard PL outperform TTT (Sun et al., 2019b); in particular, note that TTT requires a special loss function at training time, while our approach is agnostic to the pre-training phase. A detailed comparison to TTT is included in the Appendix, "Self-learning outperforms Test-Time Training (Sun et al., 2019b)". Mummadi et al. (2021) (SLR) is unpublished work and performs better than ENT and RPL on the highest severity of IN-C. SLR is an extension of entropy minimization where the

Table 3.5: Self-learning can further be improved when combining it with other techniques.

	literature results		our results		
	w/o adapt	w/ adapt	w/o adapt	w/ adapt (RPL)	w/ adapt (ENT)
top1 error [%] on IN-C test, sev. 5(↘)					
TTT, ResNet18 (Sun et al., 2019b)	86.6	66.3	85.4	**61.9**	62.8
SLR, ResNet50 (Mummadi et al., 2021)	82.0	(46.9)	82.0	54.6	54.7
top1 error [%] on CIFAR10-C (↘)					
MT3, WRN-26-1-GN (Bartler et al., 2022)	35.7	24.4	18.6	18.4	**18.0**
TTT+++, ResNet50 (Liu et al., 2021)	29.1	**9.8**	24.9	14.6	12.4
BUFR, ResNet18 (Eastwood et al., 2022)	42.4	**10.6**	25.5	13.4	12.9

entropy minimization loss is replaced with a version to ensure non-vanishing gradients of high confidence samples, as well as a diversity regularizer; in addition, a trainable module is prepended to the network to partially undo the distribution shift. Mummadi et al. (2021) introduce the hyperparameters δ as the trade-off parameter between their two losses, as well as κ as the momentum in their diversity regularization term. The success of SLR over ENT and RPL shows the promise of extending self-learning methods by additional objectives, and corroborates our findings on the effectiveness of self-learning.

We also compare our approach to Meta Test-Time Training (MT3, Bartler et al., 2022), which combines meta-learning, self-supervision and test-time training for test-time adaptation. We find that both ENT and RPL perform better than MT3: using the same architecture as Bartler et al. (2022), our best error on CIFAR10-C is 18.0% compared to their best result of 24.4%. When exchanging GroupNorm layers (Wu & He, 2018) for BN layers, the error further reduces to 13.1% (Table 3.11). We thus find that self-learning is more effective when adapting affine BN parameters instead of GN layers, which is consistent with the findings in Schneider et al. (2020a). We included a detailed comparison to Bartler et al. (2022) in the Appendix, "Comparison to Meta Test-Time Training (Bartler et al., 2022)."

TTT+++ (Liu et al., 2021) outperforms both ENT and RPL on CIFAR10-C. Since Liu et al. (2021) do not report results on IN-C, it is impossible to judge whether their gains would generalize, although they do report much better results compared to TENT on Visda-C, so TTT+++ might also be effective on IN-C. Similar to TTT, TTT+++ requires a special loss function during pretraining and thus, cannot be used as an out-of-the-box adaptation technique on top of any pretrained checkpoint.

Bottom-Up Feature Restoration (BUFR; Eastwood et al., 2022) outperforms self-learning on CIFAR10-C. The authors note that BUFR is applicable to dataset shifts with measurement shifts which stem from measurement artefacts, but not applicable to more complicated shifts where learning new features would be necessary.

Table 3.6: Vision Transformers can be adapted with self-learning.

mCE on IN-C test [%] (↘)	w/o adapt	w/ adapt affine layers	w/ adapt bottleneck layers	w/ adapt lin. layers	w/ adapt all weights
ViT-S/16	62.3	51.8	46.8	45.2	**43.5**

Self-supervised methods based on self-learning allow out-of-the-box test-time adaptation (Table 3.6). The recently published DINO method (Caron et al., 2021b) is another variant of self-supervised learning that has proven to be effective for unsupervised representation learning. At the core, the method uses soft pseudo-labeling. Here, we test whether a model trained with DINO on the source dataset can be test-time adapted on IN-C using DINO to further improve out-of-distribution performance. We highlight that we specifically test the self-supervised DINO objective for its practicality as a test-time adaptation method, and did not switch the DINO objective for ENT or RPL to do test-

time adaptation. Since the used model is a vision transformer model, we test different choices of adaptation parameters and find considerable performance improvements in all cases, yielding an mCE of 43.5% mCE at a parameter count comparable to a ResNet50 model. For adapting the affine layers, we follow Houlsby et al. (2019).

Understanding test-time adaptation with self-learning

In the following section, we show ablations and interesting insights of using self-learning for test-time adaptation. If not specified otherwise, all ablations are run on the hold-out corruptions of IN-C (our dev set) with a vanilla ResNet50.

Robust pseudo-labeling outperforms entropy minimization on large-scale datasets while the reverse is true on small-scale datasets (Table 3.7). We find that robust pseudo-labeling consistently improves over entropy minimization on IN-C, while entropy minimization performs better on smaller scale data (CIFAR10, STL10, MNIST). The finding highlights the importance of testing both algorithms on new datasets. The improvement is typically on the order of one percent point.

Table 3.7: RPL (ENT) performs better on IN-C (CIFAR10-C).

	mCE, IN-C dev			err, C10-C
	ResNet50	ResNeXt-101	EffNet-L2	WRN-40
ENT	50.0 ± 0.04	43.0	22.2	**8.5**
RPL	**48.9** ± 0.02	**42.0**	**21.3**	9.0

Robust pseudo-labeling allows usage of the full dataset without a threshold (Table 3.8). Classical hard labeling needs a confidence threshold (T) for best performance, thereby reducing the dataset size, while best performance for RPL is reached for full dataset training with a threshold T of 0.0. We corroborate the results of Wang et al. (2020a) who showed that TENT outperforms standard hard labeling with a threshold on the highest severity of CIFAR10-C and CIFAR100-C. We show that this result transfers to IN-C, for a variety of thresholds and pseudo-labeling variants.

Table 3.8: RPL performs best without a threshold.

threshold	0.0	0.5	0.9
mCE on IN-C dev [%]			
no adapt	69.5		
soft PL	60.1		
hard PL	53.8	51.9	52.4
RPL	**49.7**	49.9	51.8

Short update intervals are crucial for fast adaptation (Table 3.9). Having established that RPL generally performs better than soft- and hard-labeling, we vary the update interval for the teacher. We find that instant updates are most effective. In entropy minimization, the update interval is instant per default.

Table 3.9: RPL performs best with instantaneous updates (ResNet50).

Update interval (RPL)	w/o adapt	none	epoch	instant
mCE, IN-C dev [%]	69.5	54.0	49.7	**49.2**

Adaptation of only affine layers is important in CNNs (Table 3.10). On IN-C, adapting only the affine parameters after the normalization layers (i.e., the rescaling and shift parameters β and γ) works better on a ResNet50 architecture than adapting all parameters or only the last layer. We indicate the number of adapted parameters in brackets. Note that for Vision Transformers, full model adaptation works better than affine adaptation (see Table 3.6). We also noticed that on convolutional models with a smaller parameter count like ResNet18, full model adaptation is possible. Wang et al. (2020a) also used the affine BN parameters for test-time adaptation with TENT, and report that last layer optimization can improve performance but degrades with further optimization. They suggest that full model optimization does not improve performance at all. In contrast, we find gains with full model adaptation, but stronger gains with adaptation of only affine parameters.

Table 3.10: RPL performs best when affine BN parameters are adapted (ResNet50).

Mechanism	w/o adapt	last layer	full model	affine
mCE, IN-C dev [%]	69.5	60.2	51.5	**48.9**
adapted parameters	0	2M	22.6M	5.3k

Affine BN parameters work better for test-time adaptation compared to GN parameters. (Table 3.11). Schneider et al. (2020a) showed that models with batch normalization layers are less robust to distribution shift compared to models with group normalization (Wu & He, 2018) layers. However, after adapting BN statistics, the adapted model outperformed the non-adapted GN model. Here, we show that these results also hold for test-time adaptation when adapting a model with GN or BN layers. We show that a WideResNet-26-1 (WRN-26-1) vanilla model with BN layers pretrained on clean CIFAR10 has a much higher error on CIFAR10-C than the same model with GN layers, but it has a much lower error after adaptation. The full results for the WRN-26-1 model can be found in the Appendix, "Comparison to Meta Test-Time Training (Bartler et al., 2022)". Further, we test the pretrained BigTransfer (Kolesnikov et al., 2020) models which have GN layers, and find only small improvements with RPL , and no

improvements with ENT. There are no pretrained weights released for the BigTransfer models which have BN layers, thus, a comparison similar to the WRN-26-1 model is not possible. A more detailed discussion on our BigTransfer results as well as a hyperparameter selection study can be found in the Appendix, "Small improvements on BigTransfer models with Group normalization layers".

Table 3.11: Self-learning works better in models with BN layers compared to models with GN layers, although un-adapted models with GN are more stable under distribution shift compared to models with BN layers.

top1 error [%] on CIFAR10-C (↘)	number of parameters	w/o adapt	w/ adapt (Δ) RPL	w/ adapt (Δ) ENT
WRN-26-1-BN (Zagoruyko & Komodakis, 2016)	1.5×10^6	25.8	15.1 (-10.7)	13.1 (-12.7)
WRN-26-1-GN (Bartler et al., 2022)	1.5×10^6	18.6	18.4 (-0.2)	18.0 (-0.6)
mCE [%] on IN-C test (↘)				
ResNet50 BigTransfer (Kolesnikov et al., 2020)	2.6×10^7	55.0	54.4 (-0.6)	56.4 (+1.4)

Hyperparameters obtained on corruption datasets transfer well to real world datasets. When evaluating models, we select the hyperparameters discussed above (the learning rate and the epoch used for early stopping are the most critical ones) on the dev set (full results in Appendix B). We note that this technique transfers well to IN-R and -A, highlighting the practical value of corruption robustness datasets for adapting models on real distribution shifts.

Learning rate and number of training epochs are important hyperparameters We tune the learning rate as well as the number of training epochs for all models, except for the EfficientNet-L2 model where we only train for one epoch due to computational constraints. In Tables B.3, B.4, B.5, and B.6 in the Appendix, "Detailed results for tuning epochs and learning rates", we show that both ENT and RPL collapse after a certain number of epochs, showing the inherent instability of pseudo-labeling. While Wang et al. (2020a) trained their model only for one epoch with one learning rate, we rigorously perform a full hyperparameter selection on a hold-out set (our dev set), and use the optimal hyperparameters on all tested datasets. We believe that this kind of experimental rigor is essential in order to be able to properly compare methods.

Our analysis confirms hyperparameter choices from the literature Having searched over a broad space of hyperparameters and self-learning algorithms, we identified ENT and RPL as the best performing variants across different model sizes, architectures and pretraining techniques. We summarize the most important hyperparameter choices in Table 3.12 and compare them to those used in the literature when applying self-learning for test-time adaptation. Wang et al. (2020a) identified that on CIFAR-C, entropy minimization outperforms hard PL, and found that affine adaptation of BN parameters works better than full model adaptation, and adapted to the full dataset

since TENT does not have a threshold in contrast to hard PL. EATA (Niu et al., 2022a) and SRL (Mummadi et al., 2021) are extensions of TENT, and thus, followed their hyperparameter choices. EATA does not adapt to the full dataset as they do not adapt to very similar samples or samples with high entropy values. MEMO (Zhang et al., 2021) adapts to a single sample and adapts all model weights. It would be interesting to study whether the performance of MEMO can be improved when adapting only affine BN parameters.

Table 3.12: Our expanded analysis confirms hyperparameter choices from the literature.

Method	short updates	adapt affine params	use BN instead of GN	adapt to full dataset
TENT (Wang et al., 2020a)	✓	✓	✓	✓
EATA (Niu et al., 2022a)	✓	✓	✓	✗
SRL (Mummadi et al., 2021)	✓	✓	✓	✓
MEMO (Zhang et al., 2021)	✗	✓	✓	✗
RPL/ENT(ours)	✓	✓	✓	✓

Pure self-learning hurts calibration, (Table 3.13) Eastwood et al. (2022) show that approaches based on entropy minimization and pseudo-labeling hurt calibration due to the objective of confidence maximization. We corroborate their results and report the Expected Calibration Error (ECE; Naeini et al., 2015) when adapting a vanilla ResNet50 model with RPL and ENT. We report the mean ECE (lower is better) and standard deviation across corruptions (and severities) below. We observe that both methods increase ECE compared to the unadapted model. The increase in ECE is higher for more severe corruptions which can be explained by a successively stronger distribution shift compared to the source dataset.

Table 3.13: Self-learning leads to an increased ECE compared to the unadapted model.

adaptation	IN-C full	IN-C sev 1	IN-C sev 2	IN-C sev 3	IN-C sev 4	IN-C sev 5
w/o adapt	2.3 ± 0.8	2.3 ± 0.3	2.1 ± 0.3	2.1 ± 0.3	2.2 ± 0.4	2.9 ± 1.6
RPL	10.6 ± 7.3	6.6 ± 0.5	7.9 ± 1.5	8.9 ± 2.1	11.2 ± 3.3	18.7 ± 12.6
ENT	10.6 ± 7.3	6.6 ± 0.5	7.9 ± 1.5	8.9 ± 2.1	11.2 ± 3.3	18.7 ± 12.6

Self-learning leads to slightly decreased accuracy on the source dataset (Table 3.14) To judge the forgetting effect on the source distribution, we calculated the accuracy on clean ImageNet for our adapted ENT and RPL checkpoints. We note that success of self-learning can partially be attributed to correcting the BN statistics of the vanilla model with respect to the distribution shift (Schneider et al., 2020a). Thus, we only wish to examine the effect of fine-tuning of the affine BN parameters to the target distribution with respect to the source distribution. Thus, we again correct the mismatched statistics to the source dataset when calculating the accuracy.

We report the mean and standard deviation across corruptions (and severities) below. We observe that both ENT and RPL lead to a decrease in performance on

the source dataset, an effect which has also been observed by Niu et al. (2022a). The decrease in performance is higher for more severe corruptions which can be explained by a increasingly stronger distribution shift compared to the source dataset. We find that the effect of forgetting is less pronounced in RPL compared to ENT.

Table 3.14: Self-learning leads to slightly decreased accuracy on the source dataset (clean IN).

model	top1 accuracy on ImageNet val [%]
w/o adapt	74.2
RPL adapted to IN-C (avg over corruptions, sev. 1)	74.7 ± 0.6
RPL adapted to IN-C (avg over corruptions, sev. 2)	73.9 ± 0.9
RPL adapted to IN-C (avg over corruptions, sev. 3)	73.0 ± 1.3
RPL adapted to IN-C (avg over corruptions, sev. 4)	71.8 ± 1.8
RPL adapted to IN-C (avg over corruptions, sev. 5)	69.8 ± 3.0
ENT adapted to IN-C (avg over corruptions, sev. 1)	74.3 ± 0.6
ENT adapted to IN-C (avg over corruptions, sev. 2)	73.2 ± 1.1
ENT adapted to IN-C (avg over corruptions, sev. 3)	72.3 ± 1.7
ENT adapted to IN-C (avg over corruptions, sev. 4)	70.7 ± 2.3
ENT adapted to IN-C (avg over corruptions, sev. 5)	68.0 ± 3.9

Additional experiments and ablation studies, as well as detailed results for all models and datasets can be found in the Appendix, "Detailed and additional Results on IN-C". We discuss additional proof-of-concept implementations on the WILDS benchmark (Koh et al., 2021), BigTransfer (BiT; Chen et al., 2020c) models and on self-learning based UDA models in the Appendix, "Additional experiments". On WILDS, self-learning is effective for the Camelyon17 task with a systematic shift between train, validation and test sets (each set is comprised of different hospitals), while self-learning fails to improve on tasks with mixed domains, such as on the RxRx1 and the FMoW tasks.These results support our claim that self-learning is effective, while showing the important limitation when applied to more diverse shifts.

A simple model of stability in self-learning

We observed that different self-learning schemes are optimal for small-scale vs. large-scale datasets and varying amount of classes. We reconsider the used loss functions, and unify them into

$$\begin{aligned} \ell(\mathbf{x}) &= -\sum_j \sigma_j \left(\frac{\mathbf{f}^t(\mathbf{x})}{\tau_t} \right) \log \left(\sigma_j \left(\frac{\mathbf{f}^s(\mathbf{x})}{\tau_s} \right) \right), \\ \mathbf{f}^t(\mathbf{x}) &= \begin{cases} \mathbf{f}(\mathbf{x}), & \text{entropy minimization} \\ \text{sg}(\mathbf{f}(\mathbf{x})), & \text{pseudo-labeling.} \end{cases} \end{aligned} \tag{3.5}$$

where we introduced student and teacher temperature τ_s and τ_t as parameters in the softmax function and the stop gradient operation sg. To study the learning dynamics,

we consider a linear student network $\mathbf{f}^s = \mathbf{w}^s \in \mathbb{R}^d$ and a linear teacher network $\mathbf{f}^t = \mathbf{w}^t \in \mathbb{R}^d$ which are trained on N data points $\{\mathbf{x}_i\}_{i=1}^N$ with a binary cross-entropy loss function $\mathcal{L}$ defined as

$$\begin{aligned}\mathcal{L} &= -\sum_{i=1}^{N} \ell(\mathbf{x}_i) \\ &= -\sum_{i=1}^{N} \left(\sigma_t(\mathbf{x}_i^\top \mathbf{w}^t) \log \sigma_s(\mathbf{x}_i^\top \mathbf{w}^s) + \sigma_t(-\mathbf{x}_i^\top \mathbf{w}^t) \log \sigma_s(-\mathbf{x}_i^\top \mathbf{w}^s)\right), \\ &\text{where } \sigma_t(z) = \frac{1}{1+e^{-z/\tau_t}} \text{ and } \sigma_s(z) = \frac{1}{1+e^{-z/\tau_s}}.\end{aligned} \tag{3.6}$$

With stop gradient, student and teacher evolve in time according to

$$\dot{\mathbf{w}}^s = -\nabla_{\mathbf{w}^s} \mathcal{L}\left(\mathbf{w}^s, \mathbf{w}^t\right), \quad \dot{\mathbf{w}}^t = \alpha(\mathbf{w}^s - \mathbf{w}^t), \tag{3.7}$$

where α is the learning rate of the teacher. Without stop gradient, student and teacher are set equal to each other (following the instant updates we found to perform empirically best), and they evolve as

$$\dot{\mathbf{w}} = -\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}), \text{ where } \mathbf{w}^s = \mathbf{w}^t = \mathbf{w}. \tag{3.8}$$

We restrict the theoretical analysis to the time evolution of the components of $\mathbf{w}^{s,t}$ in direction of two data points $\mathbf{x}_k$ and $\mathbf{x}_l$, $y_k^{s,t} \equiv \mathbf{x}_k^\top \mathbf{w}^{s,t}$ and $y_l^{s,t} \equiv \mathbf{x}_l^\top \mathbf{w}^{s,t}$. All other components $y_i^{s,t}$ with $i \neq k, l$ are neglected to reduce the dimensionality of the equation system. It turns out that the resulting model captures the neural network dynamics quite well despite the drastic simplification of taking only two data points into account (see Figure 3.2 for a comparison of the model vs. self-learning on the CIFAR-C dataset). We obtain the dynamics:

$$\begin{aligned}\text{with stop gradient: } \dot{y}_k^s &= -\mathbf{x}_k^\top \nabla_{\mathbf{w}^s} \left(\ell(\mathbf{x}_k) + \ell(\mathbf{x}_l)\right), \\ \dot{y}_l^s &= -\mathbf{x}_l^\top \nabla_{\mathbf{w}^s} \left(\ell(\mathbf{x}_k) + \ell(\mathbf{x}_l)\right), \\ \dot{y}_k^t &= \alpha(y_k^t - y_k^s), \quad \dot{y}_l^t = \alpha(y_l^t - y_l^s), \\ \text{without stop gradient: } \dot{y}_k &= -\mathbf{x}_k^\top \nabla_{\mathbf{w}} \left(\ell(\mathbf{x}_k) + \ell(\mathbf{x}_l)\right), \\ \dot{y}_l &= -\mathbf{x}_l^\top \nabla_{\mathbf{w}} \left(\ell(\mathbf{x}_k) + \ell(\mathbf{x}_l)\right).\end{aligned} \tag{3.9}$$

With this setup in place, we can derive

Proposition 1 (Collapse in the two-point model)**.** *The student and teacher networks* $\mathbf{w}_s$ *and* $\mathbf{w}_t$ *trained with stop gradient do not collapse to the trivial representation* $\forall \mathbf{x} : \mathbf{x}^\top \mathbf{w}^s = 0, \mathbf{x}^\top \mathbf{w}^t = 0$ *if* $\tau_s > \tau_t$*. The network* $\mathbf{w}$ *trained without stop gradient does not collapse if* $\tau_s > \tau_t/2$*. Proof. see § B.* □

We validate the proposition on a simulated two datapoint toy dataset, as well as on the CIFAR-C dataset (Figure 3.2). In general, the size and location of the region where

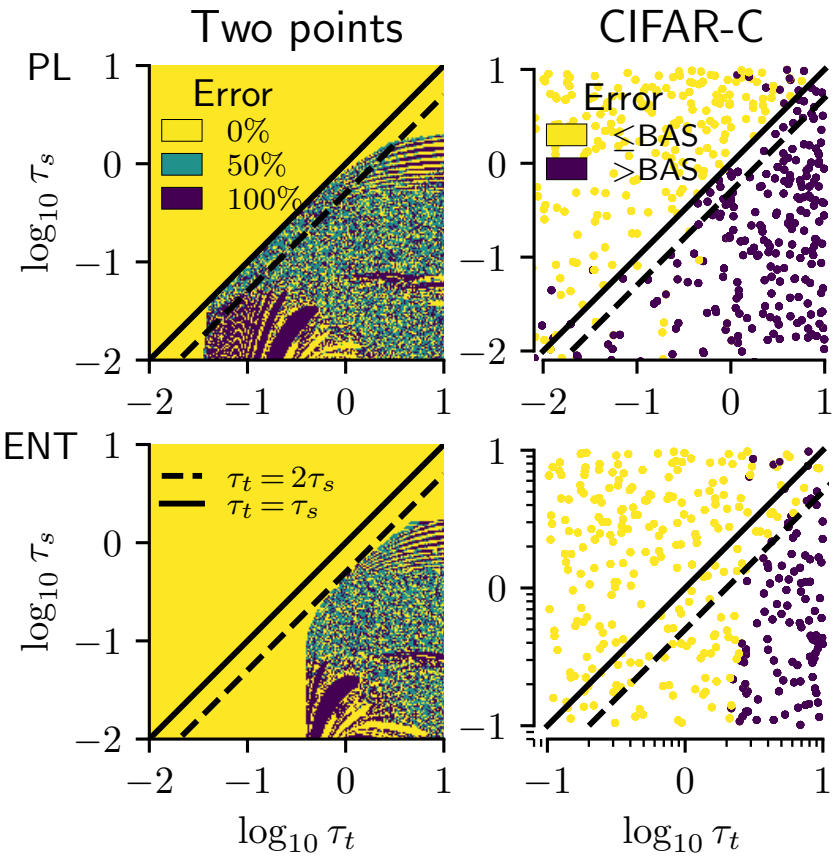


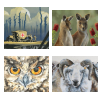
Figure 3.2: For the two point model, we show accuracy, and for the CIFAR10-C simulation, we show improvement (yellow) vs. degradation (purple) over the non-adapted baseline (BAS). An important convergence criterion for pseudo-labeling (top row) and entropy minimization (bottom row) is the ratio of student and teacher temperatures; it lies at $\tau_s = \tau_t$ for PL, and $2\tau_s = \tau_t$ for ENT. Despite the simplicity of the two-point model, the general convergence regions transfer to CIFAR10-C.

collapse is observed in the simulated model also depends on the initial conditions, the learning rate and the optimization procedure. An in depth discussion, as well as additional simulations are given in the Appendix, "A two-point model of self-learning".

Entropy minimization with standard temperatures ($\tau_s = \tau_t = 1$) and hard pseudo-labeling ($\tau_t \to 0$) are hence stable. The two-point learning dynamics vanish for soft pseudo-labeling with $\tau_s = \tau_t$, suggesting that one would have to analyze a more complex model with more data points. While this does not directly imply that the learning is unstable at this point, we empirically observe that both entropy minimization and hard labeling outperform soft-labeling in practice.

The finding aligns with empirical work: For instance, Caron et al. (2021b) fixed τ_s and varied τ_t during training, and empirically found an upper bound for τ_t above which the training was no longer stable. It also aligns with our findings suggesting that

Table 3.15: Self-learning decreases the top1 error on IN-D domains with strong initial performance, but fails to improve performance on challenging domains.

													
domain	Real		Painting		Clipart		Sketch		Infograph		Quickdraw		ImageNet
adapt	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o
model													
EffNet-L2 Noisy Student	29.2	**27.9**	42.7	**40.9**	45.0	**37.9**	56.4	**51.5**	**77.9**	94.3	**98.4**	99.4	11.6
ResNet50 DAug+AM	39.2	36.5	58.7	53.4	68.4	57.0	75.2	61.3	88.1	83.2	98.2	99.1	23.3
ResNet50 vanilla	40.1	37.3	65.1	57.8	76.0	63.6	82.0	73.0	89.6	85.1	99.2	99.8	23.9

hard-labeling tends to outperform soft-labeling approaches, and soft-labeling performs best when selecting lower teacher temperatures.

In practice, the result suggests that *student temperatures should always exceed the teacher temperatures for pseudo-labeling*, and *student temperatures should always exceed half the teacher temperature for entropy minimization*, which narrows the search space for hyperparameter optimization considerably.

Adapting models on a wider range of distribution shifts reveals limitations of robustification and adaptation methods

Robustness datasets on ImageNet-scale have so far been limited to a few selected domains (image corruptions in IN-C, image renditions in IN-R, difficult images for ResNet50 classifiers in IN-A). In order to test our approach on a wider range of complex distribution shifts, we re-purpose the dataset from the Visual Domain Adaptation Challenge 2019 (DomainNet; Saenko et al., 2019) as an additional robustness benchmark.

Creation of ImageNet-D The original DomainNet dataset comes with six image styles: "Clipart", "Real", "Infograph", "Painting", "Quickdraw" and "Sketch", and has 345 classes in total, out of which 164 overlap with ImageNet. We map these 164 DomainNet classes to 463 ImageNet classes, e.g., for an image from the "bird" class in DomainNet, we accept all 39 bird classes in ImageNet as valid predictions. ImageNet also has ambiguous classes, e.g., it has separate classes for "cellular telephone" and "dial phone". For these cases, we accept both predictions as valid. In this sense, the mapping from DomainNet to ImageNet is a one-to-many mapping. We refer to the smaller version of DomainNet that is now compatible with ImageNet-trained models as ImageNet-D (IN-D). The benefit of IN-D over DomainNet is this re-mapping to ImageNet classes which allows robustness researchers to easily benchmark on this dataset, without the need of re-training a model (as is common in UDA). We show example images from IN-D in Table 3.15. The detailed evaluation protocol on IN-D, our label-mapping procedure from DomainNet to ImageNet along with justifications

for our design choices and additional analysis are outlined in the Appendix, "Detailed and additional Results on IN-D".

The most similar robustness dataset to IN-D is IN-R which contains renditions of ImageNet classes, such as art, cartoons, deviantart, graffiti, embroidery, graphics and others. The benefit of IN-D over IN-R is that in IN-D, the images are separated according to the domain allowing for studying of systematic domain shifts, while in IN-R, the different domains are not distinguished. ImageNet-Sketch (Wang et al., 2019) is a dataset similar to the "Sketch" domain of IN-D.

More robust models perform better on IN-D. To test whether self-learning is helpful for more complex distribution shifts, we adapt a vanilla ResNet50, several robust IN-C models and the EfficientNet-L2 Noisy Student model on IN-D. We use the same hyperparameters we obtained on IN-C dev for all our IN-D experiments[3]. We show our main results in Table 3.15. Comparing the performance of the vanilla ResNet50 model to its robust DAug+AM variant, we find that the DAug+AM model performs better on all domains, with the most significant gains on the "Clipart", "Painting" and "Sketch" domains. We show detailed results for all domains and all tested models in the Appendix, "Detailed results for robust ResNet50 models on IN-D", along with results on IN-C and IN-R for comparison. We find that the best performing models on IN-D are also the strongest ones on IN-C and IN-R which indicates good generalization capabilities of the techniques combined for these models, given the large differences between the three considered datasets. The Spearman's rank correlation coefficient between IN-C and IN-D (averaged over all domains) is 0.54, and 0.73 between IN-R and IN-D. Thus, the errors on IN-R are strongly correlated to errors on IN-D which can be explained by the similarity of IN-D and IN-R. We show Spearman's rank correlation cofficients for the individual domains versus IN-C/IN-R in Fig. B.7 in the Appendix, "Detailed results on the error analysis on IN-D", and find correlation values above 0.8 between IN-R and IN-D for all domains except for the "Real" domain where the coefficient is almost zero. Further, we find that even the best models perform 20 to 30 percentage points worse on IN-D compared to their performance on IN-C or IN-R, indicating that IN-D might be a more challenging benchmark.

All models struggle with some domains of IN-D. The EfficientNet-L2 Noisy Student model obtains the best results on most domains. However, we note that the overall error rates are surprisingly high compared to the model's strong performance on the other considered datasets (IN-A: 14.8% top-1 error, IN-R: 17.4% top-1 error, IN-C: 22.0% mCE). Even on the "Real" domain closest to clean ImageNet where the EfficientNet-L2 model has a top-1 error of 11.6%, the model only reaches a top-1 error of 29.2%. Self-learning decreases the top-1 error on all domains except for "Infograph" and "Quickdraw".

[3]In regards to hyperparameter selection, we performed a control experiment where we selected hyperparameters with leave-one-out cross validation—this selection scheme actually performed worse than IN-C parameter selection (see Appendix, "Leave-one-out-cross-validation").

We note that both domains have very high error rates from the beginning and thus hypothesize that the produced pseudo-labels are of low quality.

Error analysis on IN-D. We investigate the errors a ResNet50 model makes on IN-D by analyzing the most frequently predicted classes for different domains to reveal systematic errors indicative of the encountered distribution shifts. We find most errors interpretable: the classifier assigns the label "comic book" to images from the "Clipart" or "Painting" domains, "website" to images from the "Infograph" domain, and "envelope" to images from the "Sketch" domain. Thus, the classifier predicts the domain rather than the class. We find no systematic errors on the "Real" domain which is expected since this domain should be similar to ImageNet. Detailed results on the most frequently predicted classes for different domains can be found in Fig. B.7 (Appendix, "Detailed results on the error analysis on IN-D").

IN-D should be used as an additional robustness benchmark. While the error rates on IN-C, -R and -A are at a well-acceptable level for our largest EfficientNet-L2 model after adaptation, IN-D performance is consistently worse for all models. We propose to move from isolated benchmark settings like IN-R (single domain) to benchmarks more common in domain adaptation (like DomainNet) and make IN-D publicly available as an easy to use dataset for this purpose.

Best practices and evaluation in test-time adaptation

Based on our results as well as our discussion on previous work, we arrive at several proposals on how test-time adaptation should be evaluated in future work to ensure scientific rigor:

1. *Cross-validation:* We propose using the hold-out set of IN-C for model selection of all relevant hyperparameters, and then using these hyperparameters for testing on different datasets.

2. *Comparison to simple baselines:* With proper hyperparameter tuning, very simple baselines can perform on par with sophisticated approaches. This insight is also discussed by Gulrajani and Lopez-Paz (2021) for the setting of domain generalization and by Rusak et al. (2020) for robustness to common corruptions.

3. *Using more robust models:* Test-time adaptation can further improve upon robust models, which were pre-trained with more data or with UDA, or using protocols to increase robustness. A test-time adaptation method will be much more relevant to practitioners if it can improve upon the most robust model they can find for their task.

4. *Important hyperparameters:* We identify several important hyperparameters which affect the final performance in a crucial way, and thus, should be tuned to ensure fair comparisons:

    - *Number of adaptation epochs and learning rate*: The final performance crucially depends on both of these parameters for all models and all methods that we have studied.
    - *Adaptation parameters* While adaptation of affine batch normalization parameters works best for adaptation of CNNs, full adaptation performs best for ViTs. Therefore, it is important to benchmark test-time adaptation for different model architectures and adaptation parameters.

Conclusion

We evaluated and analysed how self-learning, an essential component in many unsupervised domain adaptation and self-supervised pre-training techniques, can be applied for adaptation to both small and large-scale image recognition problems common in robustness research. We demonstrated new state-of-the-art adaptation results with the EfficientNet-L2 model on the benchmarks ImageNet-C, -R, and -A, and introduced a new benchmark dataset (ImageNet-D) which remains challenging even after adaptation. Our theoretical analysis shows the influence of the temperature parameter in the self-learning loss function on the training stability and provides guidelines how to choose a suitable value. Based on our extensive experiments, we formulate best practices for future research on test-time adaptation. Across the large diversity of (systematic) distribution shifts, architectures and pre-training methods we tested in this paper, we found that self-learning almost universally improved test-time performance. An important limitation of current self-learning methods is the observed instability over longer adaptation time frames. While we mitigate this issue through model selection (and show its robustness across synthetic and natural distribution shifts), this might not universally hold across distribution shifts encountered in practice. Concurrent work, e.g. Niu et al. (2022a) tackle this problem through modifications of self-learning algorithms, and we think this direction will be important to continue to explore in future work. That being said, we hope that our results encourage both researchers and practitioners to *experiment with self-learning if their data distribution shifts.*

Reproducibility Statement

We attempted to make our work as reproducible as possible: We mostly used pre-trained models which are publicly available and we denoted the URL addresses of all used checkpoints; for the checkpoints that were necessary to retrain, we report the Github directories with the source code and used an official or verified reference

implementation when available. We report all used hyperparameters in the Appendix and released the code.

Software and Data

Code for reproducing results of this paper is available at https://github.com/bethgelab/robustness.

Acknowledgements

We thank Roland S. Zimmermann, Yi Zhu, Raghavan Manmatha, Matthias Kümmerer, Matthias Tangemann, Bernhard Schölkopf, Justin Gilmer, Shubham Krishna, Julian Bitterwolf, Berkay Kicanaoglu and Mohammadreza Zolfaghari for helpful discussions on pseudo labeling and feedback on our paper draft. We thank Yasser Jadidi and Alex Smola for support in the setup of our compute infrastructure. We thank the anonymous reviewers from TMLR and our Action Editor for their constructive feedback.

We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting E.R. and St.S.; St.S. acknowledges his membership in the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program. This work was supported by the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A), by the Deutsche Forschungsgemeinschaft (DFG) in the priority program 1835 under grant BR2321/5-2 and in the SFB 1233, Robust Vision: Inference Principles and Neural Mechanisms (TP3), project number: 276693517. WB acknowledges financial support via an Emmy Noether Grant funded by the German Research Foundation (DFG) under grant no. BR 6382/1-1 and via the Open Philantropy Foundation funded by the Good Ventures Foundation. WB is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645.

4

RDumb: A simple approach that questions our progress in continual test-time adaptation

The following pages contain the postprint version of the paper

> Ori Press, Steffen Schneider, Matthias Kümmerer†, and Matthias Bethge†. "RDumb: A simple approach that questions our progress in continual test-time adaptation." *Advances in Neural Information Processing Systems 36 (2023)*

A short version of an earlier version of the paper was previously presented at the Principles of Distribution Shift (PODS) workshop at ICML 2022. A short version of the CCC dataset what presented at the Shift Happens Workshop at ICML 2022.

Author Contributions The order below is determined by contribution among the respective category. Conceptualization, RDumb: OP with input from all authors; Conceptualization, CCC: StS, MB, MK; Methodology: OP, StS; Software: OP; Data Curation: OP; Investigation: OP; Formal analysis: MK, StS, OP; Visualization: StS, OP with input from all authors; Writing, Original Draft: OP, StS, MK; Writing, Review & Editing: all authors; Supervision: MB, MK, StS. †MB and MK contributed equally to advising the project.

Summary

Test-Time Adaptation (TTA) allows to update pretrained models to changing data distributions at deployment time. While early work tested these algorithms for individual fixed distribution shifts, recent work proposed and applied methods for continual adaptation over long timescales. To examine the reported progress in the field, we propose the Continuously Changing Corruptions (CCC) benchmark to measure asymptotic performance of TTA techniques. We find that eventually all but one state-of-the-art methods collapse and perform worse than a non-adapting model, including models specifically proposed to be robust to performance collapse. In addition, we introduce a simple baseline, *RDumb*[1], that periodically resets the model to its pretrained state. RDumb performs better or on par with the previously proposed state-of-the-art in all considered benchmarks. Our results show that previous TTA approaches are neither effective at regularizing adaptation to avoid collapse nor able to outperform a simplistic resetting strategy.

Introduction

Biological vision is remarkably robust at adapting to continually changing environments. Imagine cycling through the forest on a cloudy day and observing the world around you: You will encounter a wide variety of animals and objects, and be able to recognize them without effort. Even as the weather changes, rain sets in, or you start cycling faster, the human visual system effortlessly adapts and robustly estimates the surroundings (Van de Ven & Tolias, 2019). Equipping machine vision with similar capabilities is a long-standing and unsolved challenge, with numerous applications in autonomous driving, medical imaging, and quality control, to name a few.

Techniques for improving the robustness to domain shifts of ImageNet-scale (Russakovsky et al., 2015) classification models include pre-training of large models on diverse and/or large-scale datasets (Mahajan et al., 2018; Radford et al., 2021; Xie et al., 2020a) and robustification of smaller models by specifically designed data augmentation (Hendrycks et al., 2020b, 2020a; Rusak et al., 2020). While these techniques are applied during training time, recent work (Goyal et al., 2022; Mummadi et al., 2021; Nado et al., 2020; Niu et al., 2022b; Rusak et al., 2021; Schneider et al., 2020a; Wang et al., 2020b; Wang et al., 2022) explored possibilities of further adapting models by Test-Time Adaptation (TTA). Such methods continuously update a given pretrained model exclusively using their input data, without having access to its labels. Test-time entropy minimization (TENT; Wang et al., 2020b) has become a foundation for state-of-the-art TTA methods. Given an input stream of images, Tent updates a pretrained classification model by minimizing the entropy of its outputs, thereby continuously increasing the model's confidence in its predictions for every input image.

[1]The name is inspired by the name and methodology of GDumb (Prabhu, Torr and Dokania, 2022) in continual learning.

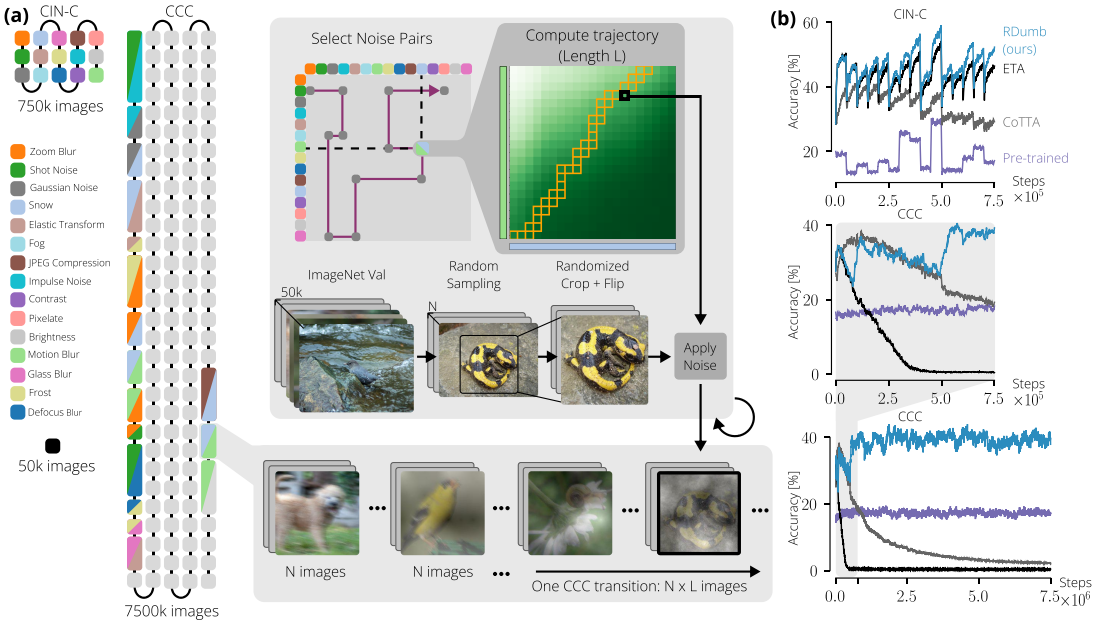


Figure 4.1: Continuously Changing Corruptions show limitations of existing TTA methods. (a) Comparison between ImageNet-Val, CIN-C and CCC. The proposed version of CCC is 10× longer than CIN-C and could naturally be extended even further without repeating images. CCC consists of sequences of smooth transitions from one ImageNet-C noise to another one. For each such pair, we construct a trajectory continuously interpolating from one pure noise to the other pure noise such that baseline accuracy is kept constant. For each point along the trajectory, we sample a batch of 1k, 2k, or 5k images from ImageNet-Val, randomly crop and flip it and apply the noise combination. (b) Due to its short length and high variability in difficulty, CIN-C (top) is unable to reveal the collapse of methods such as ETA and CoTTA, while CCC (middle and bottom) can.

Previous TTA work (Goyal et al., 2022; Mummadi et al., 2021; Nado et al., 2020; Rusak et al., 2021; Schneider et al., 2020a; Wang et al., 2020b; Zhang et al., 2022) evaluate their models on ImageNet-C (Hendrycks & Dietterich, 2019a) or smaller scale image classification benchmarks (Krizhevsky, Hinton, et al., 2009; LeCun et al., 2010). ImageNet-C consists of 75 copies of the ImageNet validation set, wherein each copy is corrupted according to 15 different noises at 5 different severity levels. When TTA models are evaluated on ImageNet-C, they are adapted on each noise and severity combination individually starting from their pretrained weights. Such a one-time adaptation approach is of little relevance when it comes to deploying TTA models in realistic scenarios. Instead, stable performance over a long run time after deployment is the desirable goal.

TTA methods are by design readily applicable to this setting and recently the field has started to move towards testing TTA models in continual adaptation settings (Gong et al., 2022; Niu et al., 2022b; Wang et al., 2022). Strikingly, this revealed that the dominant TTA approach Tent (Wang et al., 2020b) decreases in accuracy over time, eventually being less accurate than a non-adapting, pretrained model (Niu et al., 2022b;

Wang et al., 2022). In this work, we refer to any model whose classification accuracy falls below that of a non-adapting, pretrained model, as having "collapsed".

This collapsing behaviour of Tent shows that it cannot be used in continual adaptation over long time scales without modifications. While previous benchmarking of TTA methods already managed to reveal the collapse of Tent, our work shows that in fact all current TTA methods collapse sooner or later, *including methods with explicit built-in anti-collapse strategies*.

Since current benchmarks have not been sufficient to detect collapse in several models, we introduce an image classification benchmark designed to thoroughly evaluate TTA models for their long-term behavior. Our benchmark, *Continuously Changing Corruptions* (CCC), tests models on their ability to adapt to image corruptions that are constantly changing, much like when fog turns to rain or day turns to night. CCC allows us to easily control different factors that could affect the ability of a given method to continuously adapt: the corruptions and their order, the difficulty of the images themselves, and the speed at which corruptions transition. Most importantly, the length of our benchmark is ten times longer than that of previous benchmarks, and more diverse by including all kinds of combinations of corruptions (see Figure 4.1a). Using CCC, we discover that seven recently published state-of-the-art TTA methods are less accurate than a non-adapting, pretrained model. While Tent was already shown to collapse (Gong et al., 2022; Niu et al., 2022b; Wang et al., 2022), we show that this problem is not specific to Tent, and that many other methods – including specifically designed continual adaptation methods – collapse as well.

Finally, we propose "*RDumb*" [2] as a minimalist baseline mechanism that simply *Resets* the model to its pretrained weights at regular intervals. Previous work employs more sophisticated methods combining entropy minimization with various regularization approaches, yet we show that RDumb is superior on both existing benchmarks and ours (CCC). Our results call the progress made in continual TTA so far into question, and provide a richer set of benchmarks for realistic evaluation of future methods.

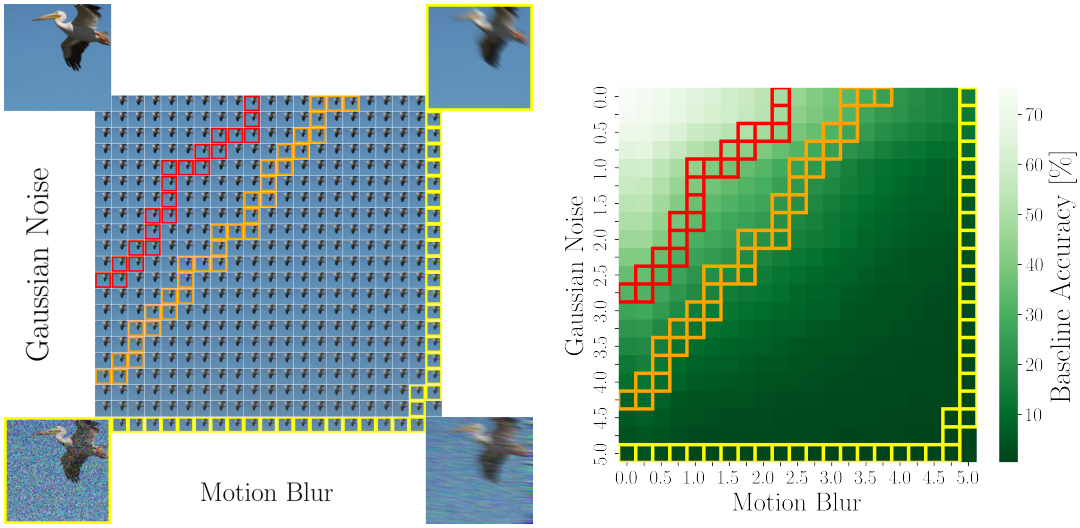
Our contributions are:

- We introduce the continual adaptation benchmark CCC. We show that previous benchmarks are too short to meaningfully assess long-term continual adaptation behaviour, and are too uncontrolled to assess the short-term learning dynamics.

- Using CCC, we show that the performances of all but one current TTA methods drop below a non-adapting, pre-trained baseline when trained over long timescales.

- We propose "*RDumb*" as a baseline and show that it outperforms all previous methods with a minimalist resetting strategy.

[2] The name was inspired by GDumb (Prabhu et al., 2020).



(a) (b)

Figure 4.2: (a) Each corruption of CCC consists of applying two ImageNet-C corruptions at different severities. We extend the individual severities to be more fine-grained than in ImageNet-C, allowing for smoother noise changes, and exponentially more (noise, severity) combinations. The corners are enlarged for easier viewing, zoom in for greater detail. (b) Sample dataset sequences with a constant baseline accuracy. The sequences start from the left where Motion Blur is zeroed out, and end at the top with Gaussian noise zeroed out. The colors red, orange, and yellow correspond to trajectories in CCC-Easy, CCC-Medium and CCC-Hard, respectively.

CCC: Towards Infinite Testing with Continuously Changing Corruptions

Until recently, it was common to evaluate TTA methods only on datasets on individual domain shifts such as the corruptions of ImageNet-C (Hendrycks & Dietterich, 2019a). However, the world is steadily changing and recently the community started moving towards continual adaptation, i.e., evaluating methods with respect to their ability to adapt to ongoing domain shifts (Niu et al., 2022b; Sun et al., 2019b; Wang et al., 2022).

The dominant method of evaluating continual adaptation on ImageNet scale is to concatenate the top severity datasets of the 15 ImageNet-C corruptions into one big dataset. We refer to the variant of this dataset introduced by Wang et al. (2022) as *Concatenated ImageNet-C* (CIN-C). CIN-C was used to demonstrate the collapse of Tent and the stability of recent TTA methods by Gong et al. (2022), Niu et al. (2022b), and Wang et al. (2022).

In Figure 4.1b, we evaluate a range of TTA methods on CIN-C and notice three potential problems: Firstly, ETA (Niu et al., 2022b) appears to be stable and better than a non-adapting, pretrained baseline, but is revealed to collapse when tested on CCC. Additionally, while CoTTA (Wang et al., 2022) clearly goes down in performance, it is not yet clear whether it collapses or stabilizes above or below baseline performance. Fundamentally, CIN-C turns out to be too short to yield reliable, conclusive results.

Secondly, assessing adaptation dynamics is further obscured by the considerable variations of the baseline performance among the different corruptions in CIN-C. This is not only a factor that affects adaptation itself (shown by Niu et al. (2022b) and Wang et al. (2020b)), it also leads to substantial fluctuations in performance across multiple runs, making it difficult to obtain a clear and reliable assessment. Finally, CIN-C features exclusively abrupt transitions between different corruption types. In contrast, in the real world, domain changes may often be smooth and subtle with varying speeds: day to night, rain to sunshine, or the accumulation of dust on a camera. Therefore, it is important to also probe TTA methods on continual domain changes that are not tied to a specific point in time and thus constitute a relevant test for stable continual adaptation.

Here we propose a new benchmark, *Continuously Changing Corruptions* (CCC), to address these issues. CCC solves the issues of benchmark length, uncontrolled baseline difficulty, and transition smoothness in a simple and effective manner. Firstly, the length issue is remedied because the individual runs of CCC are constructed by a generation process which can generate very long datasets without reusing images. In this work we use runs of 7.5M images, which is 10 times as long as CIN-C. If required to compare methods in future work (where collapse is even slower), it is straightforward to generate even longer benchmarks within the CCC framework. Secondly, since both (Niu et al., 2022b; Wang et al., 2020b) have shown that dataset difficulty is a confounder when studying adaptation, the difficulty of individual benchmark runs is kept stable. Additionally, we examine three different difficulty levels to ensure a comprehensive yet controlled evaluation. Finally, CCC exhibits smooth domain shifts: it applies two corruptions to each image. Over time, the severity of one corruption is smoothly increased while the severity of the other is decreased, maintaining the desired difficulty. We also study three different speeds for applying this process. We will now outline the generation procedure of the dataset.

Continuously changing image corruptions To allow smooth transitions between corruptions, we introduce a more fine-grained severity level system to the ImageNet-C dataset. We interpolate the parameters of the original corruptions (integer-valued severities from 1 to 5) to finer grained severity levels from 0 to 5 in steps of 0.25. We apply two different ImageNet-C corruptions to each image, such that we can decrease the severity of one corruption while increasing the severity of another one. Hence, the corruptions of CCC are given by quadruples (c_1, s_1, c_2, s_2), where c_1 and c_2 are ImageNet-C corruption types and s_1 and s_2 are severity levels. When applying such a corruption, we first apply c_1 and then c_2 at their respective severities (see Figure 4.2a).

Calibration to desired baseline accuracy In order to control baseline accuracy, we need to know how difficult each combination of 2 noises and their respective severities is. To that end, we first select a subset of 5,000 images from the ImageNet validation set. For each corruption (c_1, s_1, c_2, s_2), we corrupt all 5,000 images accordingly and evaluate the

resulting images with a pre-trained ResNet-50 (He et al., 2016a). The resulting accuracy is what we refer to as *baseline accuracy* and what we use for controlling difficulty. In total, we evaluate more than 463 million corrupted images. Previous work, (Hendrycks & Dietterich, 2019b), measures normalized accuracy using AlexNet (Krizhevsky et al., 2012b), which is less pertinent in present-day contexts. In addition, the accuracy of non-adapting Vision Transformers are stable on CCC as well (Figure 4.4).

Generating Benchmark Runs Having calibrated the corruptions pairs, we prepare benchmark runs with different baseline accuracies, transition speeds, and noise orderings. We pick 3 different baseline accuracies: 34%, 17%, and 2% (CCC-Easy, CCC-Medium, CCC-Hard respectively). For each one of the difficulties, we select a further 3 transition speeds: 1k, 2k, 5k. Lastly, for each difficulty and transition speed combination we use 3 different noise orderings, determined by 3 random seeds. To generate each run, we first select the initial corruption at the severity which according to our calibration is closest to the desired baseline accuracy. We then transition to the second corruption of the noise ordering by repeatedly either decreasing the severity of the first noise by 0.25 or increasing the severity of the second noise by 0.25 such that the baseline accuracy is as close to the target as possible (see Figure 4.2). In each step along each path, we sample 1k, 2k, or 5k images from the ImageNet validation set depending on the desired transition speed. Each image is randomly cropped and flipped for increasing the diversity of the dataset, and then corrupted.

Once the path from the initial to the second corruption is finished, the process is repeated for transitioning to the third corruption and so on (for more details see Appendix, "Path Finding Algorithm"). In the end, we have 3 difficulties consisting of 9 benchmark runs each. CCC-Medium at a speed of 2k corresponds roughly to CIN-C's difficulty and transition speed.

RDumb: Turning your model off and on again

Continual test-time adaptation needs to successfully adapt models over arbitrarily long timescales during deployment. Resetting a model to its initial weights at fixed intervals fulfills this criterion by design, yet allows to benefit from adaptation over short time scales. Surprisingly, such an approach has never been tried before (see Appendix, "Novelty of Resetting" for more discussion).

Regarding the choice of the adaptation loss, we build on the weighted entropy used in ETA (Niu et al., 2022b). For a stream of input images $\mathbf{x}_1, \mathbf{x}_2, \ldots$, we compute class probabilities $\mathbf{y}_t = f_{\Theta_t}(\mathbf{x}_t)$ and optimize the loss function

$$L(\mathbf{y}_t; \overline{\mathbf{y}}_{t-1}) = \left(\frac{\mathbb{1}[(|\cos(\mathbf{y}_t, \overline{\mathbf{y}}_{t-1})| < \epsilon) \wedge (H(\mathbf{y}_t) < H_0)]}{\exp(H(\mathbf{y}_t) - H_0)} \right) H(\mathbf{y}_t) \tag{4.1}$$

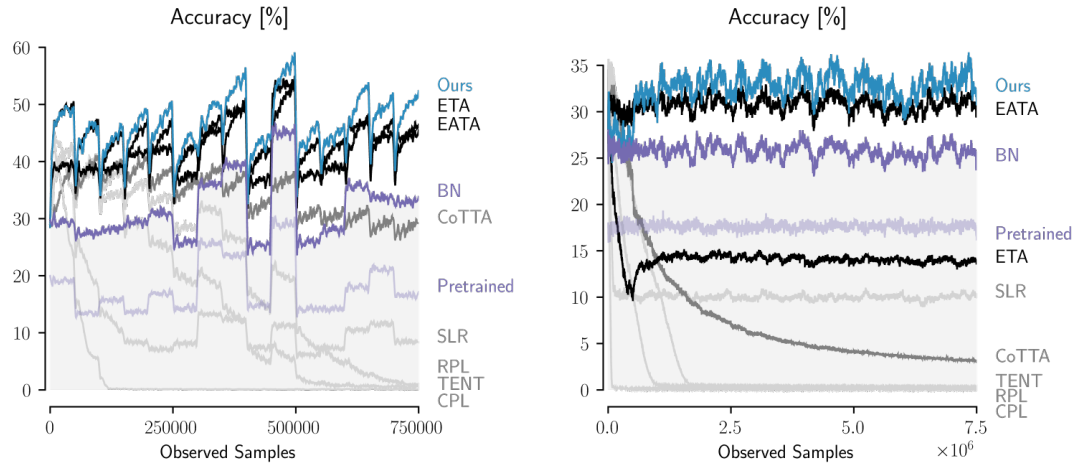
which weights the entropy $H(\mathbf{y}_t) = -\mathbf{y}_t^\top (\log \mathbf{y}_t)$ of each prediction using the similarity

to averaged previously predicted class probabilities, $\bar{\mathbf{y}}_t = (\mathbf{y}_1 + \cdots + \mathbf{y}_t)/t$, and a comparison to a fixed entropy threshold H_0. $\cos(\mathbf{u}, \mathbf{v})$ refers to the cosine similarity between vectors $\mathbf{u}$ and $\mathbf{v}$. At each step, (part of) the weights Θ_t are updated using the Adam optimizer (Kingma & Ba, 2014). At every T-th step, Θ_t is reset to the baseline weights Θ_0. We use $\epsilon = 0.05$ and $H_0 = 0.4 \times \ln 10^3$ following Niu et al. (2022b), and select $T = 1000$ based on the holdout noises in IN-C (see section "Optimal reset intervals").

Experiment Setup

We benchmark RDumb alongside a range of recently published TTA models. For all models, we use a batch size of 64. In all models, the BatchNorm statistics are estimated on the fly, and the affine shift and scale parameters are optimized according to a model-specific strategy outlined below.

- **BatchNorm (BN) Adaptation** (Nado et al., 2020; Schneider et al., 2020a) estimates the BatchNorm statistics (mean and variance) separately for each batch at test time. The affine transformation parameters are not adapted.
- **Tent** (Wang et al., 2020a) optimizes the entropy objective on the test set in order to update the scale and shift parameters of BatchNorm (in addition to learning the statistics).
- **Robust Pseudo-Labeling (RPL)** (Rusak et al., 2021) uses a teacher-student approach in combination with a label noise resistant loss.
- **Conjugate Pseudo Labels (CPL)** (Goyal et al., 2022) use meta learning to learn the optimal adaptation objective function across a class of possible functions.
- **Soft Likelihood Ratio (SLR)** (Mummadi et al., 2021) uses a loss function that is similar to entropy, but without vanishing gradients. *Anti-Collapse Mechanism:* An additional loss is used to encourage the model to have uniform predictions over the classes, and the last layer of the network is kept frozen.
- **Continual Test Time Adaptation (CoTTA)** (Wang et al., 2022) uses a teacher student approach in combination with augmentations. *Anti-Collapse Mechanism:* Every iteration, 0.1% of the weights are reset back to their pretrained values.
- **Efficient Test Time Adaptation (EATA)** (Niu et al., 2022b) uses 2 weighing functions to weigh its outputs: the first based on their entropy (lower entropy outputs get a higher weight), the second based on diversity (outputs that are similar to seen before outputs are excluded). *Anti-Collapse Mechanism:* An L_2 regularizer loss is used to encourage the model's weights to stay close to their initial values.



(a) (b)

Figure 4.3: Adaptation performance of all evaluated models depending on the number of observed samples so far. (a) CIN-C. Model performances are averaged over the 10 runs of the benchmark. (b) CCC. Model performances are averaged over the 27 runs of the three difficulty levels. See Appendix, "CCC Plots", Figure C.4 for separate plots for CCC Easy, Medium and Hard.

- **EATA Without Weight Regularization (ETA)** For completeness, we also test ETA, which is EATA but without the regularizer loss, proposed in (Niu et al., 2022b).
- **RDumb** is our proposed baseline to mitigate collapse via resetting. We reset every $T = 1,000$ steps, as determined by a hyperparameter search on the holdout set (Section "Optimal reset intervals").

Following the original implementations, Tent, ETA, EATA, and RDumb use SGD with a learning rate of $2.5 \cdot 10^{-4}$. RPL uses SGD with a learning rate of $5 \cdot 10^{-4}$. SLR uses the Adam optimizer with a learning rate of $6 \cdot 10^{-4}$. CoTTA uses SGD with a learning rate of 0.01, and CPL uses SGD with a learning rate of 0.001.

Results

CCC reveals collapse during continual adaptation, unlike CIN-C. For three models that were evaluated, evaluation on CIN-C yielded inconclusive or inaccurate results in detecting collapse: CoTTA collapses on CCC, while CIN-C shows it to be on a downward trend, but end performance still outperformed the baseline (Figure 4.3a). Additionally, ETA shows no signs of collapse on CIN-C, while collapsing very clearly on CCC (Figure 4.3b, more precisely on CCC-Medium and CCC-Hard, see Appendix Figure C.4). When tested using ViT backbones, EATA is better than the pretrained model on CIN-C (Figure 4.4a), but worse than the pretrained model on CCC (Figure 4.4b, Table 4.2,4.3). Lastly, SLR on CIN-C appears to be somewhat stable, but only at around 10% accuracy. CCC

Table 4.1: Mean accuracy of ResNet-50 models on CIN-C, CIN-3DCC and CCC. For each CCC split (Easy, Medium, and Hard), a mean of 9 runs is taken. For the CIN-C and CIN-3DCC experiments, the accuracy reported is the mean of 10 different noise permutations. Grey indicates collapse. [1]Nado et al. (2020) and Schneider et al. (2020a).

Adaptation method	CIN-C	CIN-3DCC	CCC-Easy	CCC-Medium	CCC-Hard	Average
Pretrained (He et al., 2016a)	18.0 ± 0.0	31.5 ± 0.0	34.1 ± 0.22	17.3 ± 0.21	1.5 ± 0.02	20.5
BN[1]	31.5 ± 0.02	35.7 ± 0.02	42.6 ± 0.39	27.9 ± 0.74	6.8 ± 0.31	28.9
Tent (Wang et al., 2020b)	15.6 ± 3.5	24.4 ± 3.5	3.9 ± 0.58	1.4 ±0.17	0.51 ± 0.07	9.2
RPL (Rusak et al., 2021)	21.8 ± 3.6	30.0 ± 3.6	7.5 ± 0.83	2.7 ± 0.36	0.67 ± 0.14	12.5
SLR (Mummadi et al., 2021)	12.4 ± 7.7	12.2 ± 7.7	22.2 ± 18.4	7.7 ± 9.0	0.66 ± 0.57	11.0
CPL (Goyal et al., 2022)	3.0 ± 3.3	5.7 ± 3.3	0.41 ± 0.06	0.22 ± 0.03	0.14 ± 0.01	1.9
CoTTA (Wang et al., 2022)	34.0 ± 0.68	37.6 ± 0.68	14.9 ± 0.88	7.7 ± 0.43	1.1 ± 0.16	19.1
EATA (Niu et al., 2022b)	41.8 ± 0.98	43.6 ± 0.98	48.2 ± 0.6	35.4 ±1.0	8.7 ±0.8	35.5
ETA (Niu et al., 2022b)	43.8 ± 0.33	42.7 ± 0.33	41.4 ± 0.95	1.1 ± 0.43	0.23 ± 0.05	25.8
RDumb (ours)	**46.5 ± 0.15**	**45.2 ± 0.15**	**49.3 ± 0.88**	**38.9 ± 1.4**	**9.6 ± 1.6**	**37.9**

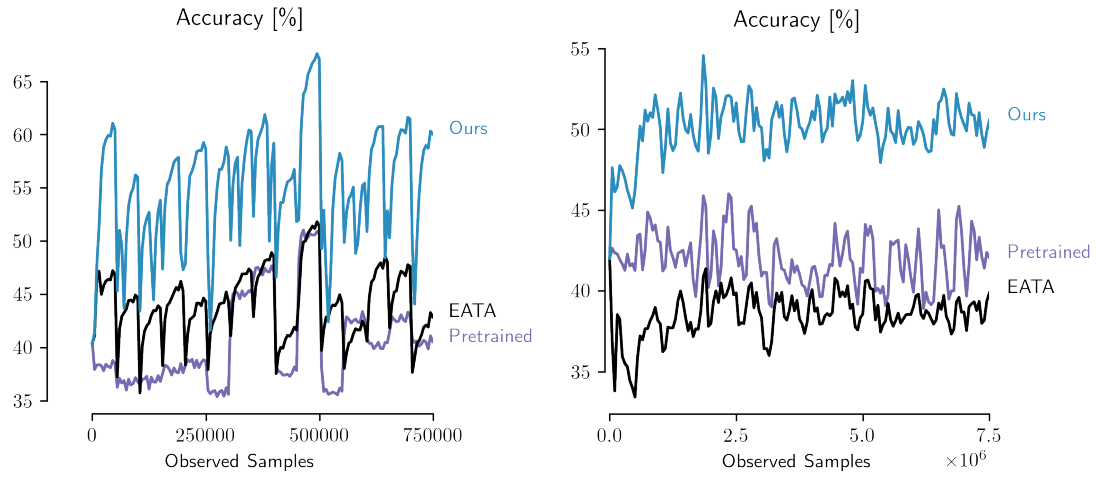
reveals this to be only partly true: on CCC-Hard, SLR is not stable and collapses to nearly chance accuracy. In summary, models evaluated on CCC show clear limits, which are impossible to see on CIN-C because of the high difficulty variance between runs, and its short length.

RDumb is a strong baseline for continual adaptation. RDumb outperforms all previous methods on both established benchmarks (CIN-C, CIN-3DCC) as well as our continual adaptation benchmark, CCC (Table 4.1 and Figure 4.3). Concretely, we outperform EATA and increase accuracy by more than 11% on CIN-C (improving from 41.8% points to 46.5% points), and by almost 7% when averaged all evaluation datasets. While not able to outperform RDumb, we note that EATA is also a strong method for preventing collapse except for the counterexample in Table 4.2.

The results transfer to Imagenet-3D Common Corruptions. To further demonstrate the effectiveness of our method, we show results on Imagenet-3DCC (Kar et al., 2022), which features 12 types of corruptions, which take the geometry and distances between objects into account when applied to an image.

Similarly to CIN-C, we test our models on 10 different permutations of concatenations of all the noises of IN-3DCC, which we call CIN-3DCC. As in the case of CIN-C and CCC, RDumb outperforms all previous methods (Table 4.1).

The results transfer to Vision Transformers. To further validate our claims, we test both EATA and our method with a Vision Transformer (ViT) (Dosovitskiy et al., 2021) backbone (Figure 4.4). The difference in average accuracy between our method and EATA is larger when using a ViT, as compared to a ResNet-50: on CIN-C and CCC the gap is 10.9% points and 11.7% points respectively. Additionally, EATA's accuracy on



(a) (b)

Figure 4.4: TTA using a ViT backbone: (a) On CIN-C, EATA is better than the pretrained baseline (44.4% points vs 40.1% points). (b) On CCC-Medium, EATA is worse than the pretrained baseline (38.5% points vs 42.0% points). RDumb (ours) is consistently better than both EATA and the baseline.

Table 4.2: Mean accuracy of different backbone architectures on CCC-Medium. Accuracy reported is an average across 9 runs. Backbones used: (Dosovitskiy et al., 2021; He et al., 2016a; Liu et al., 2022; Tu et al., 2022; Xie et al., 2017), †: AugMix (Hendrycks et al., 2020b), ‡: DeepAugment (Hendrycks et al., 2020a). Grey indicates collapse.

Method	RN18	RN34	RN50	RN50†	RN50 †‡	RNXt101 †‡	ViT-B16	MaxViT-T	SwinViT-T
Pretrained	12.2	17.3	17.3	27.9	38.9	47.8	42.0	45.1	33.2
EATA	26.8	30.8	35.4	46.5	**52.3**	**58.5**	38.5	47.1	35.6
RDumb	**32.5**	**37.2**	**38.9**	**47.0**	51.9	58.4	**50.2**	**49.9**	**36.5**

CCC is below that of a pretrained, non-adapting model [3]. This collapse can only be seen by using CCC, and not when evaluating on CIN-C.

RDumb allows adaptation of a variety of architectures without tuning. We evaluate RDumb and EATA across a range of popular backbone architectures. Out of the nine architectures evaluated (see Table 4.2,4.3), RDumb outperformed EATA by an average margin of 4.5% points on seven of them, and worse by an average margin of only 0.25% points on the remaining two.

[3]Increasing the regularizer parameter value does not help stabilize the model, see Appendix, "EATA Implementation and Ablations".

Table 4.3: Mean accuracy of different backbone architectures on CCC-Hard. Accuracy reported is an average across 9 runs. Backbones used: (Dosovitskiy et al., 2021; He et al., 2016a; Liu et al., 2022; Tu et al., 2022; Xie et al., 2017), †: AugMix (Hendrycks et al., 2020b), ‡: DeepAugment (Hendrycks et al., 2020a). Grey indicates collapse.

Method	RN18	RN34	RN50	RN50†	RN50 †‡	RNXt101 †‡	ViT-B16	MaxViT-T	SwinViT-T
Pretrained	0.82	1.3	1.5	5.6	24.3	15.6	**22.0**	**22.0**	**9.3**
EATA	6.4	7.6	8.7	**17.7**	**30.3**	**36.3**	8.6	15.4	8.6
RDumb	**8.3**	**10.7**	**9.6**	14.7	29.9	35.6	**23.8**	**22.0**	8.0

Table 4.4: Accuracy of our method for different resetting times on CIN-C-Holdout

T (steps)	125	250	500	1000	1500	2000
Acc. [%]	42.1	44.4	46.0	**46.7**	46.5	46.4

Analysis and Ablations

Optimal reset intervals. To determine the optimal reset interval, we run ETA with reset intervals $T \in [125, 250, 500, 1000, 1500, 2000]$ on CIN-C using the IN-C holdout noises. We concatenate the 4 holdout noises at severity 5 as our base test set. This base test set is repeated until the model sees 750k images, which is equal to the length of CIN-C. We do this for every permutation of the 4 holdout corruptions. On this holdout set, we find that the optimal T is equal to 1,000.

RDumb is less sensitive to hyperparameters. An added benefit to our method is that it is less sensitive to hyperparameters than EATA. We conduct a simple hyperparameter search of the E_0 parameter—the hyperparameter that controls how many outputs get filtered out because of their high entropy. Our method consistently outperforms EATA across every hyperparameter tested (Table 4.5), and for the highest value, 0.7, EATA collapses to almost chance accuracy on all splits, while our method does not. In addition, RDumb's performance benefits from finetuning ($H_0 = \{0.2, 0.3\}$), while EATA is not able to improve.

Table 4.5: Average accuracy on all of CCC splits on a variety of H_0 values. For all other experiments in this paper we use $H_0 = 0.4 \times \ln(10^3)$, as in (Niu et al., 2022b).

$H_0 \times \ln 10^3$	0.1	0.2	0.3	0.4	0.5	0.6	0.7
EATA	27.8	27.9	29.9	**30.8**	28.7	28.0	0.33
RDumb (ours)	31.6	32.9	**33.1**	32.6	30.7	25.7	16.8

RDumb is effective because ETA reaches maximum adaptation performance fast. ETA is quick to adapt to new noises from scratch. On each of the holdout set noises and severities, ETA reaches its maximum accuracy after seeing only about 12,500 samples, which is about 200 adaptation steps (Figure 4.5a). After that, accuracy decays at a pace slower than its initial increase. Therefore, when resetting and readapting from scratch, only a few steps with substantially suboptimal predictions are encountered before performance is again close to optimal.

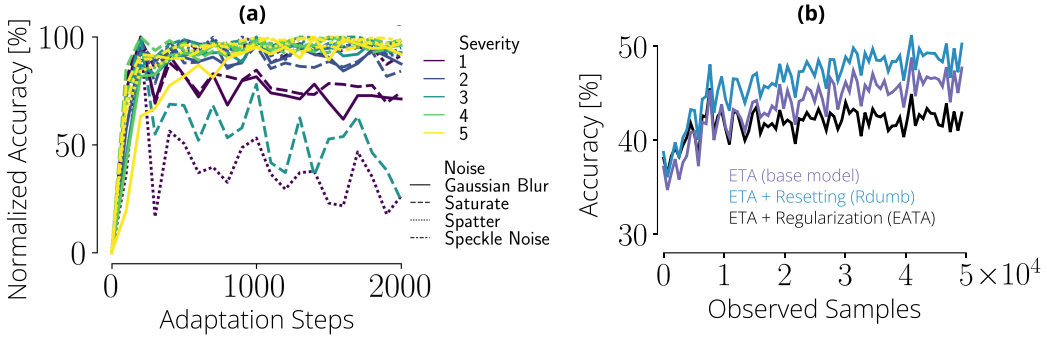


Figure 4.5: (a) ETA's normalized accuracy over time, on the ImageNet-C holdout noises and each of their severities. For every noise in the holdout set, ETA reaches its maximum accuracy very quickly. (b) Rdumb shares ETA's property of fast adaptation, while regularization in EATA slows adaptation.

Comparing resetting to regularization. Previous works typically optimize two loss terms: one loss encourages adaptation on unseen data, another loss regularizes the model to prevent collapse. Having to optimize two losses should be harder than optimizing just one – we see evidence for this in short term adaptation on CIN-C (Figure 4.5b): ETA and EATA optimize the same loss, but EATA additionally optimizes an anti-collapse loss. Consequently, ETA beats EATA by 2% points on CIN-C.

Collapse Analysis. We now investigate potential causes and effects of the observed collapse behavior. We propose a theoretical model, fully specified in Appendix C, which can explain both collapsing and non-collapsing runs. The model consists of a batch norm layer followed by a linear layer trained with the Tent objective. Within this model, we can present two scenarios. In the first, the model successfully adapts and plateaus at high accuracy (Figure C.1a). In the second, we see early adaptation which is then followed by collapse (Figure 4.6a,C.1a). The properties of noise in the data influence whether we observe the case of successful or unsuccessful adaptation.

Interestingly, the model predicts that the magnitude of weights increases over the course of optimization; this signature of entropy minimization can be found in both the theoretical model and a real experiment using RDumb without resetting on CCC-Medium (Figure 4.6). Unfortunately, weight explosion happens only *after* model

performance is already collapsed (Figure C.2b). The effect is observable across all layers (Figure 4.6c,C.2).

Discussion and Related Work

Domain Adaptation. In practice, the data distribution at deployment is different from training, and hence the task of *domain adaptation*, i.e., the task of adapting models to different target distributions has received a lot of attention (Geirhos et al., 2018a; Hendrycks et al., 2020b; Lee, 2013a; Liang et al., 2021; Rusak et al., 2020; Sun et al., 2019b; Wang et al., 2020b). The methods on domain adaptation split into different categories based on what information is assumed to be available during adaptation. While some methods assume access to labeled data for the target distribution (Motiian et al., 2017; Yue et al., 2021), *unsupervised domain adaptation* methods assume that the model has access to labeled source data and unlabeled target data at adaptation time (Ganin et al., 2016; Lee, 2013a; Liang et al., 2021; Sun et al., 2019a). Most useful for practical applications is the case of *test-time adaptation*, where the task is to adapt to the target data on the fly, without having access to the full target distribution, or the original training distribution (Nado et al., 2020; Niu et al., 2022b; Rusak et al., 2021; Schneider et al., 2020a; Sun et al., 2019b; Wang et al., 2020b; Zhang et al., 2022).

In addition to the division made above, one can further distinguish what assumptions are made about how the target domain is changing. Many academic benchmarks focus on one-time distribution shifts. However, in practical applications, the target distribution can easily change perpetually over time, e.g., due to changing weather and lightness conditions, or due to sensor corruptions. Therefore, the latter setting of *continual adaptation* has been receiving increasing attention recently. The earliest example of adapting a classifier to an evolving target domain that we are aware of is (Hoffman et al., 2014), which learn a series of transformations to keep the data representation similar over time. (Ganin et al., 2016; Tzeng et al., 2017; Wulfmeier et al., 2018) use an adversarial domain adaptation approach for this. (Bobu et al., 2018) pointed out that two of these approaches can be prone to catastrophic forgetting (Ratcliff, 1990). To deal with this, different solutions have been proposed (Abnar et al., 2021; Bobu et al., 2018; Liu et al., 2020b; Mummadi et al., 2021; Niu et al., 2022b; Wang et al., 2022).

Test-time adaptation methods have classically been applied in the setting of one-time domain change, but can be readily applied in the setting of continual adaptation, and some recent methods have been explicitly designed and tested with continual adaptation in mind (Niu et al., 2022b; Wang et al., 2020b; Wang et al., 2022). Because TTA methods use only test data and don't alter the training procedure, they are particularly easy to apply and have been shown to be superior to other domain adaptation approaches (Geirhos et al., 2018a; Nado et al., 2020; Rusak et al., 2020; Schneider et al., 2020a), Therefore, we focus only on TTA methods, which we discussed in more detail in Section 4.

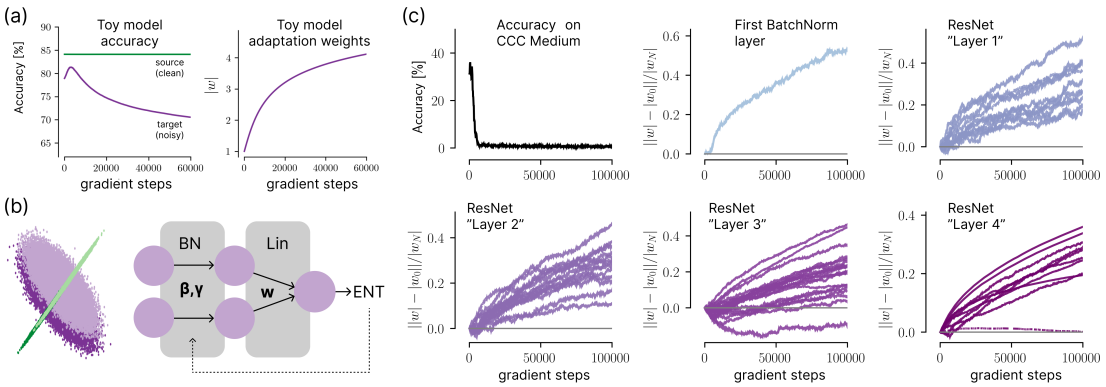


Figure 4.6: Analysis of entropy minimization collapse on synthetic and real data. Learning dynamics in terms of accuracy and weight magnitude are shown in (a) for a two layer toy model consisting of batch norm and linear layer (b). Consistent with the theoretical analysis, we find that the adaptation weights in all layers increase over time continually (c), even long after the collapse as indicated by Accuracy on CCC-Medium has happened. Refer to Figure C.1–C.2 and Appendix C for additional properties of the toy model (a–b) and a zoomed-in view on (c).

Continual Adaptation Benchmarks. While continually changing datasets are used in the continual learning literature, e.g. (Feng et al., 2019; Han et al., 2021; Lomonaco & Maltoni, 2017; Shi et al., 2020; Sun et al., 2020; Yu et al., 2020), they have been used in TTA benchmarks only very recently. In contrast to all previous benchmarks, we want to evaluate how continual adaptation methods change over long periods of time, when the noise changes in a continuous manner. The longest datasets for TTA were made up of hundreds of thousands of labeled images in total, while we adapt to 7.5M images per run. Other datasets are comprised of short video clips (Lomonaco & Maltoni, 2017; Shi et al., 2020; Sun et al., 2020) 10-20 seconds in length. Besides maximizing its length, we set out to create a dataset that is well calibrated and closely related to the well-known ImageNet-C dataset. Additionally, with our noise synthesis, we can guarantee a wide variety of noises in each one of our evaluation runs, we can control the speed at which the noise changes, and we can control the difficulty of the generated noise. Lastly, CCC accounts for different adaptation speeds, as demonstrated by (Rusak et al., 2021) and (Mummadi et al., 2021). They showed that training their methods on ImageNet-C for more than one epoch leads to better performance.

Conclusion

TTA techniques are increasingly applied to continual adaptation settings. Yet, we show that all current TTA techniques collapse in some continual adaptation settings, eventually performing worse than even non-adapting models. And while some methods are stable in some situations, they are still outperformed by our simplistic baseline "RDumb", which avoids collapse by resetting the model to its pretrained state periodically. These observations were made possible by our new benchmark for continual

adaptation (CCC), which was carefully designed for the precise assessment of long and short term adaptation behaviour of TTA methods and we envision it to be a helpful tool for the development of new, more stable adaptation methods.

Acknowledgements

We thank Evgenia Rusak, Çağatay Yıldız, Shyamgopal Karthik, and Ofir Press for helpful discussions and feedback on the manuscript.

We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting OP and StS; StS acknowledges his membership in the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program. StS was supported by a Google Research PhD Fellowship (StS). MB is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project No 390727645 and acknowledges support by the German Research Foundation (DFG): SFB 1233, Robust Vision: Inference Principles and Neural Mechanisms, TP 4, Project No: 276693517.

The authors declare no conflicts of interests.

5

Pretraining boosts out-of-domain robustness for pose estimation

The following pages contain the postprint version of the published paper

> Alexander Mathis*, Thomas Biasi*, Steffen Schneider, Mert Yuksekgonul, Byron Rogers, Matthias Bethge, and Mackenzie W. Mathis. "Pretraining boosts out-of-domain robustness for pose estimation." *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1859-1868. 2021.*

Author Contributions StS designed the Horse-C benchmark and led the investigation on the respective part of the paper. StS implemented and conducted experiments on batch norm adaptation on Horse-10 and Horse-C, including formal analysis and visualization, and participated in editing the original draft of the paper.

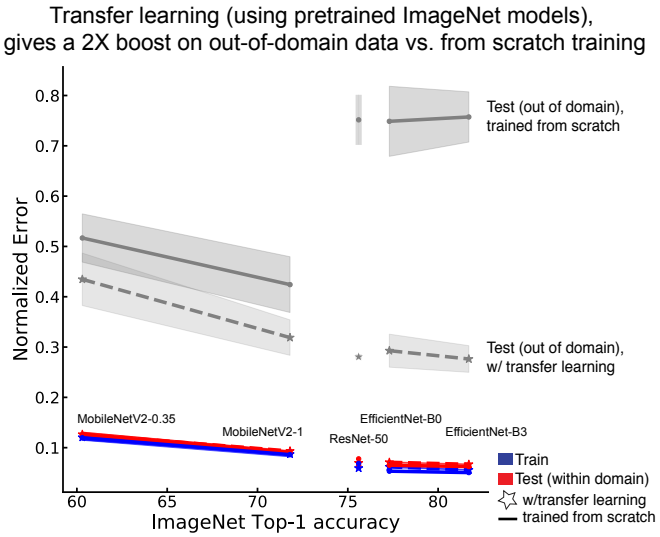


Figure 5.1: Transfer Learning boosts performance, especially on out-of-domain data. Normalized pose estimation error vs. ImageNet Top 1% accuracy with different backbones. While training from scratch reaches the same task performance as fine-tuning, the networks remain less robust as demonstrated by poor accuracy on out-of-domain horses. Mean ± SEM, 3 shuffles.

Summary

Neural networks are highly effective tools for pose estimation. However, as in other computer vision tasks, robustness to out-of-domain data remains a challenge, especially for small training sets that are common for real-world applications. Here, we probe the generalization ability with three architecture classes (MobileNetV2s, ResNets, and EfficientNets) for pose estimation. We developed a dataset of 30 horses that allowed for both "within-domain" and "out-of-domain" (unseen horse) benchmarking—this is a crucial test for robustness that current human pose estimation benchmarks do not directly address. We show that better ImageNet-performing architectures perform better on both within- and out-of-domain data if they are first pretrained on ImageNet. We additionally show that better ImageNet models generalize better across animal species. Furthermore, we introduce Horse-C, a new benchmark for common corruptions for pose estimation, and confirm that pretraining increases performance in this domain shift context as well. Overall, our results demonstrate that transfer learning is beneficial for out-of-domain robustness.

Introduction

Pose estimation is an important tool for measuring behavior, and thus widely used in technology, medicine and biology (Bachmann et al., 2015; Maceira-Elvira et al., 2019; Mathis & Mathis, 2020; Ostrek et al., 2019). Due to innovations in both deep learning algorithms (Cao et al., 2017; Cheng et al., 2020; He et al., 2017; Insafutdinov et al., 2017;

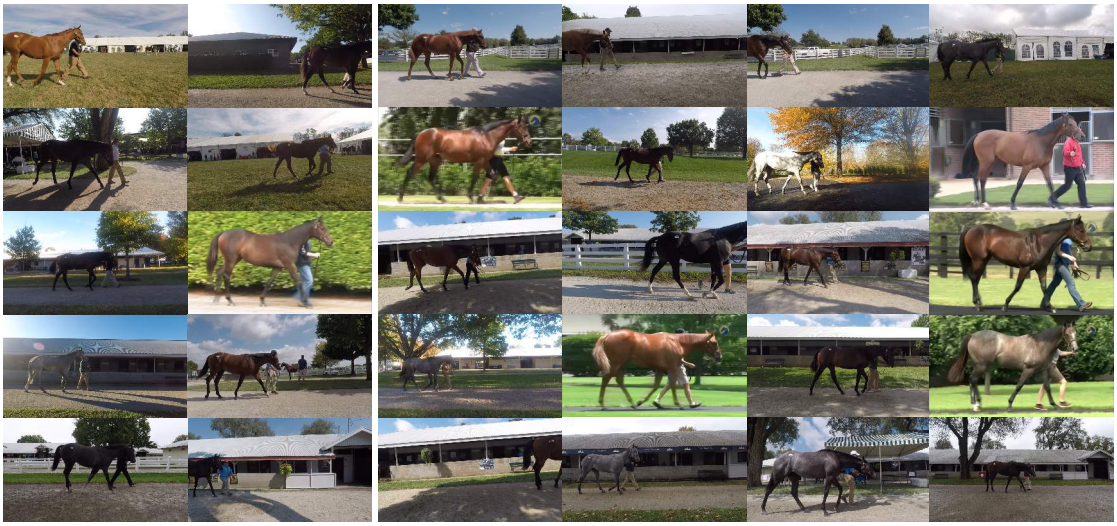


Figure 5.2: Horse Dataset: Example frames for each Thoroughbred horse in the dataset. The videos vary in horse color, background, lighting conditions, and relative horse size. The sunlight variation between each video added to the complexity of the learning challenge, as well as the handlers often wearing horse-leg-colored clothing. Some horses were in direct sunlight while others had the light behind them, and others were walking into and out of shadows, which was particularly problematic with a dataset dominated by dark colored coats. To illustrate the Horse-10 task we arranged the horses according to one split: the ten leftmost horses were used for train/test within-domain, and the rest are the out-of-domain held out horses.

Kreiss et al., 2019; Ning et al., 2020) and large-scale datasets (Andriluka et al., 2014, 2018; Lin et al., 2014) pose estimation on humans has become very powerful. However, typical human pose estimation benchmarks, such as MPII pose and COCO (Andriluka et al., 2014, 2018; Lin et al., 2014), contain many different individuals (>10k) in different contexts, but only very few example postures per individual. In real world applications of pose estimation, users often want to create customized networks that estimate the location of user-defined bodyparts by only labeling a few hundred frames on a small subset of individuals, yet want this to generalize to new individuals (Maceira-Elvira et al., 2019; Mathis & Mathis, 2020; Ostrek et al., 2019; Sanakoyeu et al., 2020). Thus, one naturally asks the following question: Assume you have trained an algorithm that performs with high accuracy on a given (individual) animal for the whole repertoire of movement—how well will it generalize to different individuals that have slightly or dramatically different appearances? Unlike in common human pose estimation benchmarks, here the setting is that datasets have many (annotated) poses per individual (>200) but only a few individuals ($\approx$10).

To allow the field to tackle this challenge, we developed a novel benchmark comprising 30 diverse Thoroughbred horses, for which 22 body parts were labeled by an expert in 8114 frames (Dataset available at http://horse10.deeplabcut.org). Horses have various coat colors and the "in-the-wild" aspect of the collected data at various Thoroughbred farms added additional complexity. With this dataset we could directly

test the effect of pretraining on out-of-domain data. Here we report two key insights: (1) ImageNet performance predicts generalization for both within domain and on out-of-domain data for pose estimation; (2) While we confirm that task-training can catch up with fine-tuning pretrained models given sufficiently large training sets (He et al., 2018), we show this is not the case for out-of-domain data (Figure 5.1). Thus, transfer learning improves robustness and generalization. Furthermore, we contrast the domain shift inherent in this dataset with domain shift induced by common image corruptions (Hendrycks & Dietterich, 2019b; Michaelis et al., 2019a), and we find pretraining on ImageNet also improves robustness against those corruptions.

Related Work

Pose and keypoint estimation

Typical human pose estimation benchmarks, such as MPII pose and COCO (Andriluka et al., 2014, 2018; Lin et al., 2014) contain many different individuals ($> 10k$) in different contexts, but only very few example postures per individual. Along similar lines, but for animals, Cao et al. created a dataset comprising a few thousand images for five domestic animals with one pose per individual (Cao et al., 2019). There are also papers for facial keypoints in horses (Rashid et al., 2017) and sheep (Hewitt & Mahmoud, 2019; Yang et al., 2016) and recently a large scale dataset featuring 21.9k faces from 334 diverse species was introduced (Khan et al., 2020). Our work adds a dataset comprising multiple different postures per individual (>200) and comprising 30 diverse race horses, for which 22 body parts were labeled by an expert in 8114 frames. This pose estimation dataset allowed us to address within and out of domain generalization. Our dataset could be important for further testing and developing recent work for domain adapation in animal pose estimation on a real-world dataset (Li et al., 2020b; Mu et al., 2020; Sanakoyeu et al., 2020).

Transfer learning

Transfer learning has become accepted wisdom: fine-tuning pretrained weights of large scale models yields best results (**Yosinski2014**; Donahue et al., 2014; Kümmerer et al., 2016; Li et al., 2019; Mathis et al., 2018; Zhuang et al., 2019). He et al. nudged the field to rethink this accepted wisdom by demonstrating that for various tasks, directly training on the task-data can match performance (He et al., 2018). We confirm this result, but show that on held-out individuals ("out-of-domain") this is not the case. Raghu et al. showed that for target medical tasks (with little similarity to ImageNet) transfer learning offers little benefit over lightweight architectures (Raghu et al., 2019). Kornblith et al. showed for many object recognition tasks, that better ImageNet performance leads to better performance on these other benchmarks (Kornblith et al., 2019b). We show that this is also true for pose-estimation both for within-domain and out-of-domain

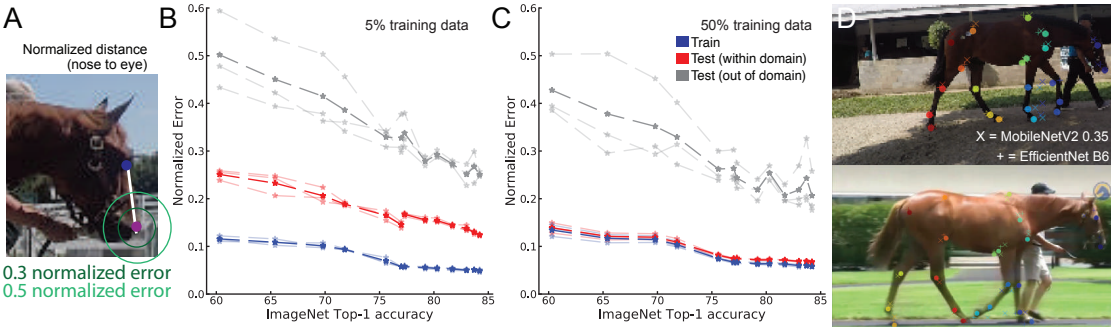


Figure 5.3: Transfer Learning boosts performance, especially on out-of-domain data. **A:** Illustration of normalized error metric, i.e. measured as a fraction of the distance from nose to eye (which is approximately 30 cm on a horse). **B:** Normalized Error vs. Network performance as ranked by the Top 1 % accuracy on ImageNet (order by increasing ImageNet performance: MobileNetV2-0.35, MobileNetV2-0.5, MobileNetV2-0.75, MobileNetV2-1.0, ResNet-50, ResNet-101, EfficientNets Bo to B6). The faint lines indicate data for the three splits. Test data is in red, train is blue, grey is out-of-domain data **C:** Same as B but with 50 % training fraction. **D:** Example frames with human annotated body parts vs. predicted body parts for MobileNetV2-0.35 and EfficientNet-B6 architectures with ImageNet pretraining on out-of-domain horses.

data (on different horses, and for different species) as well as for corruption resilience.

What is the limit of transfer learning? Would ever larger data sets give better generalization? Interestingly, it appears to strongly depend on what task the network was pretrained on. Recent work by Mahajan et al. showed that pretraining for large-scale hashtag predictions on Instagram data (3.5 billion images) improves classification, while at the same time possibly harming localization performance for tasks like object detection, instance segmentation, and keypoint detection (Mahajan et al., 2018). This highlights the importance of the task, rather than the sheer size as a crucial factor. Further corroborating this insight, Li et al. showed that pretraining on large-scale object detection task can improve performance for tasks that require fine, spatial information like segmentation (Li et al., 2019). Thus, one interesting future direction to boost robustness could be to utilize networks pretrained on OpenImages, which contains bounding boxes for 15 million instances and close to 2 million images (Kuznetsova et al., 2018).

Robustness

Studying robustness to common image corruptions based on benchmarks such as ImageNet-C (Hendrycks & Dietterich, 2019b; Michaelis et al., 2019a; Schneider et al., 2020b) is a fruitful avenue for making deep learning more robust. Apart from evaluating our pose estimation algorithms on novel horses (domain-shift), we also investigate the robustness with respect to image corruptions. Hendrycks et al. study robustness to out-of distribution data on CIFAR 10, CIFAR 100 and TinyImageNet (but not pose estimation). The authors report that pretraining is important for adversarial robustness (Hendrycks et al., 2019a). Shah et al. found that pose estimation algorithms

are highly robust against adversarial attacks (Shah et al., 2019), but neither directly test out-of-domain robustness on different individuals, nor robustness to common image corruptions as we do in this study.

Data and Methods

Datasets and evaluation metrics

We developed a novel horse data set comprising 8114 frames across 30 different horses captured for 4-10 seconds with a GoPro camera (Resolution: 1920×1080, Frame Rate: 60 FPS), which we call Horse-30 (Figure 5.2). We downsampled the frames by a factor of 15% to speed-up the benchmarking process (288×162 pixels; one video was downsampled to 30%). We annotated 22 previously established anatomical landmarks for equines (Anderson & McIlwraith, 2004; Magnusson & Thafvellin, 1990). The following 22 body parts were labeled in 8114 frames: Nose, Eye, Nearknee, Nearfrontfetlock, Nearfrontfoot, Offknee, Offfrontfetlock, Offfrontfoot, Shoulder, Midshoulder, Elbow, Girth, Wither, Nearhindhock, Nearhindfetlock, Nearhindfoot, Hip, Stifle, Offhindhock, Offhindfetlock, Offhindfoot, Ischium. We used the DeepLabCut 2.0 toolbox (Nath et al., 2019) for labeling. We created 3 splits that contain 10 randomly selected training horses each (referred to as Horse-10). For each training set we took a subset of 5 % ($\approx$160 frames), and 50 % ($\approx$ 1470 frames) of the frames for training, and then evaluated the performance on the training, test, and unseen (defined as "out-of-domain") horses (i.e. the other horses that were not in the given split of Horse-10). As the horses could vary dramatically in size across frames, due to the "in-the-wild" variation in distance from the camera, we normalized the raw pixel errors by the eye-to-nose distance and report the fraction of this distance (normalized error) as well as percent correct keypoint metric (Andriluka et al., 2014); we used a matching threshold of 30 % of the head segment length (nose to eye per horse; see Figure 5.3A).

For the generalization experiments, we also tested on the Animal Pose dataset (Cao et al., 2019) to test the generality of our findings (Figure 5.4). We extracted all single animal images from this dataset, giving us 1091 cat, 1176 dog, 486 horse, 237 cow, and 214 sheep images. To note, we corrected errors in ground truth labels for the dog's (in about 10 % of frames). Because nearly all images in this dataset are twice the size of the Horse-10 data, we also downsampled the images by a factor of 2 before training and testing. Given the lack of a consistent eye-to-nose distance across the dataset due to the varying orientations, we normalized as follows: the raw pixel errors were normalized by the square root of the bounding box area for each individual. For training the various architectures, the best schedules from cross validation on Horse-10 were used (see Section 3.2).

We also applied common image corruptions (Hendrycks & Dietterich, 2019b) to the Horse-10 dataset, yielding a variant of the benchmark which we refer to as Horse-C. Horse-C images are corrupted with 15 forms of digital transforms, blurring filters,

point-wise noise or simulated weather conditions. All conditions are applied following the evaluation protocol and implementation by Michaelis et al. (Michaelis et al., 2019a). In total, we arrived at 75 variants of the dataset (15 different corruptions at 5 different severities), yielding over 600k images.

Architectures and Training Parameters

We utilized the pose estimation toolbox called DeepLabCut (Insafutdinov et al., 2016; Mathis et al., 2018; Nath et al., 2019), and added MobileNetV2 (Sandler et al., 2018b) and EfficientNet backbones (Tan & Le, 2019) to the ResNets (He et al., 2016d) that were present, as well as adding imgaug for data augmentation (Jung et al., 2020). The TensorFlow (Abadi et al., 2016)-based network architectures could be easily exchanged while keeping data loading, training, and evaluation consistent. The feature detectors in DeepLabCut consist of a backbone followed by deconvolutional layers to predict pose scoremaps and location refinement maps (offsets), which can then be used for predicting the pose while also providing a confidence score. As previously, for the ResNet backbones we utilize an output stride of 16 and then upsample the filter banks with deconvolutions by a factor of two to predict the heatmaps and location-refinement at 1/8-th of the original image size scale (Insafutdinov et al., 2016; Mathis et al., 2018). For MobileNetV2 (Sandler et al., 2018b), we configured the output-stride as 16 (by changing the last stride 2 convolution to stride 1).

We utilized four variants of MobileNetV2 with different expansion ratios (0.35, 0.5, 0.75 and 1) as this ratio modulates the ImageNet accuracy from 60.3 % to 71.8 %, and pretrained models on ImageNet from TensorFlow (Abadi et al., 2016; Sandler et al., 2018b).

The baseline EfficientNet model was designed by Tan et al. (Tan & Le, 2019) through a neural architecture search to optimize for accuracy and inverse FLOPS. From B0 to B6, compound scaling is used to increase the width, depth, and resolution of the network, which directly corresponds to an increase in ImageNet performance (Tan & Le, 2019). We used the AutoAugment pretrained checkpoints from TensorFlow as well as adapted the EfficientNet's output-stride to 16 (by changing the (otherwise) last stride 2 convolution to stride 1).

The training loss is defined as the cross entropy loss for the scoremaps and the location refinement error via a Huber loss with weight 0.05 (Mathis et al., 2018). The loss is minimized via ADAM with batch size 8 (Kingma & Ba, 2014). For training, a cosine learning rate schedule, as in (Kornblith et al., 2019b) with ADAM optimizer and batchsize 8 was used; we also performed augmentation, using imgaug (Jung et al., 2020), with random cropping and rotations. Initial learning rates and decay target points were cross-validated for MobileNetV2 0.35 and 1.0, ResNet-50, EfficientNet B0, B3, and B5 for the pretrained and from scratch models (see Supplementary Material). For each model that was not cross validated (MobileNetV2 0.5 and 0.75, ResNet-101, EfficientNet B1, B2, B4, B6), the best performing training parameters from the most

similar cross validated model was used (i.e. the cross validated EfficientNet-B0 schedule was used for EfficientNet-B1; see Supplementary Material). For MobileNetV2s, we trained the batch normalization layers too (this had little effect on task performance for MobileNetV2-0.35). Pretrained models were trained for 30k iterations (as they converged), while models from scratch were trained for 180k iterations. From scratch variants of the architectures used He-initialization (He et al., 2015), while all pretrained networks were initialized from their ImageNet trained weights.

Cross Validation of Learning Schedules

To fairly compare the pose estimation networks with different backbones, we cross-validated the learning schedules. For models with pretraining and from scratch, we cross validated the cosine learning rate schedules by performing a grid search of potential initial learning rates and decay targets to optimize their performance on out of domain data. Given that our main result is that while task-training can catch up with fine-tuning pretrained models given sufficiently large training sets on within-domain-data (consistent with (He et al., 2018)), we will show that this is not the case for out-of-domain data. Thus, in order to give models trained from scratch the best shot, we optimized the performance on out of domain data. Tables in the Supplementary Material describe the various initial learning rates explored during cross validation as well as the best learning schedules for each model.

Similarity Analysis

To elucidate the differences between pretrained models and models trained from scratch, we analyze the representational similarity between the variants. We use linear centered kernel alignment (CKA) (Kornblith et al., 2019a) to compare the image representations at various depths in the backbone networks. The results were aggregated with respect to the similarity of representations of within domain images versus out of domain images, and averaged over the three shuffles.

Results

To test within and out-of-domain performance we created a new dataset of 30 different Thoroughbreds that are led by different humans, resulting in a dataset of 8114 images with 22 labeled body parts each. These videos differ strongly in horse appearance, context, and background (Figure 5.2). Thus, this dataset is ideal for testing robustness and out-of-sample generalization. We created 3 splits containing 10 random horses each, and then varied the amount of training data from these 10 horses (referred to as Horse-10, see Methods). As the horses could vary dramatically in size across frames, due to the "in-the-wild" variation in distance from the camera, we used a normalized pixel error; i.e. we normalized the raw pixel errors by the eye-to-nose distance and report the fraction within this distance (see Methods).

Table 5.1: average PCK@0.3 (%)

MODELS	WITHIN DOMAIN	OUT-OF-D.
MOBILENETV2-0.35	95.2	63.5
MOBILENETV2-0.5	97.1	70.4
MOBILENETV2-0.75	97.8	73.0
MOBILENETV2-1	98.8	77.6
RESNET-50	99.8	81.3
RESNET-101	99.9	84.3
EFFICIENTNET-B0	99.9	81.6
EFFICIENTNET-B1	99.9	84.5
EFFICIENTNET-B2	99.9	84.3
EFFICIENTNET-B3	99.9	86.6
EFFICIENTNET-B4	99.9	86.9
EFFICIENTNET-B5	99.9	87.7
EFFICIENTNET-B6	99.9	88.4

Table 5.2: PCK@0.3 (%) for several bodyparts and architectures on within domain horses (FF=front foot; HF = Hind foot; HH = Hind Hock).

	Nose	Eye	Shoulder	Wither	Elbow	NearFF	OffFF	Hip	NearHH	NearHF	OffHF
MobileNetV2 0.35	90.7	94.1	97.6	96.9	96.7	92.3	93.7	96.4	94.1	94.2	92.5
MobileNetV2 0.5	94.1	96.1	99.2	98.3	98.0	93.8	95.4	96.7	97.2	97.2	97.0
MobileNetV2 0.75	96.0	97.5	99.2	98.0	99.0	96.6	96.8	98.8	97.6	98.0	97.4
MobileNetV2 1.0	97.7	98.8	99.7	99.1	99.0	97.6	97.3	99.4	98.4	98.5	98.9
ResNet 50	99.9	100.0	99.8	99.9	99.8	99.8	99.6	99.9	99.9	99.6	99.8
ResNet 101	99.9	100.0	99.9	99.8	99.9	99.8	99.7	99.8	99.9	99.7	99.9
EfficientNet-B0	99.7	99.9	100.0	99.9	100.0	99.6	99.5	100.0	99.9	99.7	99.7
EfficientNet-B1	99.8	99.9	100.0	99.8	99.9	99.5	99.8	100.0	99.8	99.8	99.8
EfficientNet-B2	99.9	99.9	100.0	99.9	100.0	99.8	99.7	99.9	99.8	99.7	99.7
EfficientNet-B3	99.9	99.9	99.9	99.9	99.9	99.7	99.6	99.7	99.8	99.6	99.9
EfficientNet-B4	100.0	100.0	99.9	99.8	99.9	99.6	99.7	99.9	99.7	99.8	99.8
EfficientNet-B5	99.9	99.9	100.0	99.9	100.0	99.7	99.8	99.6	99.8	99.8	99.9
EfficientNet-B6	99.9	99.9	99.9	99.8	100.0	99.8	99.9	99.8	99.8	99.7	99.8

ImageNet performance vs task performance

To probe the impact of ImageNet performance on pose estimation robustness, we selected modern convolutional architectures as backbones with a wide range of ImageNet performance (see Methods; 13 models spanning from 60% to 84% ImageNet performance). To fairly compare the MobileNetV2, ResNet and EfficientNet backbones, we cross validated the learning schedules for each model (see Methods). In total, we found that all ImageNet-pretrained architectures exhibited strong performance on Horse-10 within domain, i.e. low average errors, and high average percent correct key points (aPCK; Figure 5.3, Table 5.1). Performance on Horse-10 within domain also closely matched performance on Horse-30 (see Supplementary Material). Next, we directly compared the ImageNet performance to their respective performance on this pose estimation task. We found Top-1% ImageNet accuracy correlates with pose estimation test error (linear fit for test: slope -0.33%, $R^2 = 0.93$, $p = 1.4 \times 10^{-7}$; Figure 5.3). Results for different bodyparts are displayed in Table 5.2.

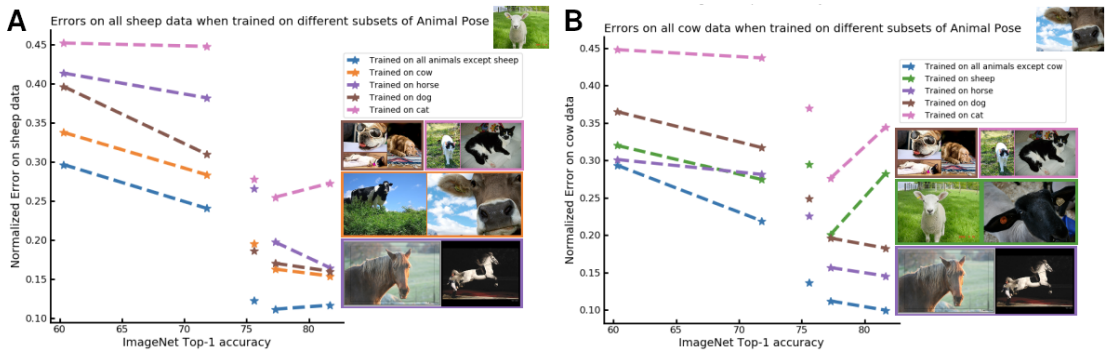


Figure 5.4: Generalization Across Species. Normalized pose estimation error vs. ImageNet Top 1% accuracy with different backbones (as in Figure 5.1), but for 10 additional out-of-domain tests. Training on either a single species or four species while holding one species (either cow or sheep) out.

Generalization to novel horses

Next, we evaluated the performance of the networks on different horses in different contexts, i.e. out-of-domain horses (Figures 5.3A-C). Most strikingly, on out-of-domain horses, the relationship between ImageNet performance and performance on Horse-10 was even stronger. This can be quantified by comparing the linear regression slope for out-of-domain test data: -0.93% pose-estimation improvement per percentage point of ImageNet performance, $R^2 = 0.93$, $p = 9 \times 10^{-8}$ vs. within-domain test data -0.33%, $R^2 = 0.93$, $p = 1.4 \times 10^{-7}$ (for 50% training data). Results for several different bodyparts of the full 22 are displayed in Table 5.3, highlighting that better models also generalized better in a bodypart specific way. In other words, *less* powerful models overfit more on the training data.

Generalization across species

Does the improved generalization to novel individuals also hold for a more difficult out-of-domain generalization, namely, across species? Thus, we turned to a pose-estimation

Table 5.3: PCK@0.3 (%) for several bodyparts and architectures on out-of-domain horses (FF=front foot; HF = Hind foot; HH = Hind Hock).

	Nose	Eye	Shoulder	Wither	Elbow	NearFF	OffFF	Hip	NearHH	NearHF	OffHF
MobileNetV2 0.35	45.6	53.1	65.5	68.0	69.1	56.4	57.6	65.9	65.9	60.5	62.5
MobileNetV2 0.5	52.7	61.0	76.7	69.7	78.3	62.9	65.4	73.6	70.8	68.1	69.7
MobileNetV2 0.75	54.2	65.6	78.3	73.2	80.5	67.3	68.9	80.0	74.1	70.5	70.2
MobileNetV2 1.0	59.0	67.2	83.8	79.7	84.0	70.1	72.1	82.0	79.9	76.0	76.7
ResNet 50	68.2	73.6	85.4	85.8	88.1	72.6	70.2	89.2	85.7	77.0	74.1
ResNet 101	67.7	72.4	87.6	86.0	89.0	79.9	78.0	92.6	87.2	83.4	80.0
EfficientNet-B0	60.3	62.5	84.9	84.6	87.2	77.0	75.4	86.7	86.7	79.6	79.4
EfficientNet-B1	67.4	71.5	85.9	85.7	89.6	80.0	81.1	86.7	88.4	81.8	81.6
EfficientNet-B2	68.7	74.8	84.5	85.2	89.2	79.7	80.9	88.1	88.0	82.3	81.7
EfficientNet-B3	71.7	76.6	88.6	88.7	92.0	80.4	81.8	90.6	90.8	85.0	83.6
EfficientNet-B4	71.1	75.8	88.1	87.4	91.8	83.3	82.9	90.8	90.3	86.7	85.5
EfficientNet-B5	74.8	79.5	89.6	89.5	93.5	82.2	84.1	91.8	90.9	86.6	85.2
EfficientNet-B6	74.7	79.7	90.3	89.8	92.8	83.6	84.4	92.1	92.1	87.8	85.3

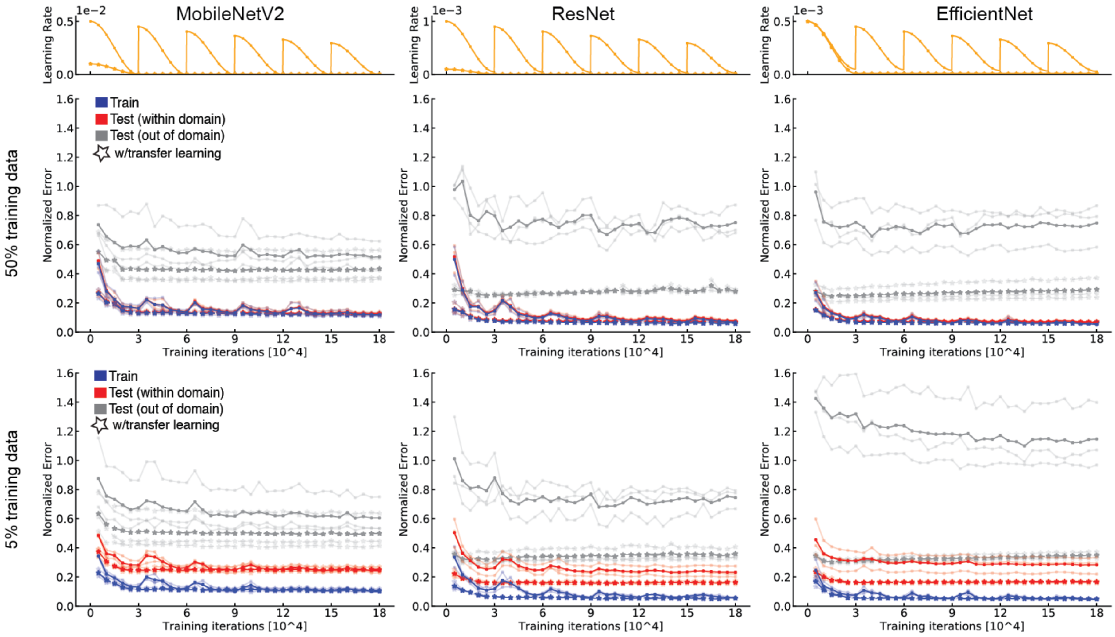


Figure 5.5: Training randomly initialized networks longer cannot rescue out-of-domain performance. **Top Row:** Best performing (cross-validated) learning schedules used for training. **Middle:** Normalized error vs. training iterations for MobileNetV2-0.35, ResNet-50 and EfficientNet-B0 *using* 50% *of the training data*. Test errors when training from scratch (solid lines) closely match the transfer learning (dashed lines) performance after many iterations. Crucially, out-of-domain testing does not approach performance for pretrained network (stars). **Bottom Row:** Same as Middle but *using* 5% *of the training data*; note, however, for just 5% training data, the test errors do not approach the test error of pretrained models for larger models.

dataset comprising multiple species. We evaluated the performance of the various architectures on the Animal Pose dataset from Cao et. al (Cao et al., 2019). Here, images and poses of horses, dogs, sheep, cats, and cows allow us to test performance across animal classes. Using ImageNet pretraining and the cross validated schedules from our Horse-10 experiments, we trained on individual animal classes or multiple animal classes (holding out sheep/ cows) and examined how the architectures generalized to sheep/cows, respectively (Figure 5.4). For both cows and sheep, better ImageNet architectures, trained on the pose data of other animal classes performed better, in most cases. We mused that this improved generalization could be a consequence of the ImageNet pretraining or the architectures themselves. Therefore, we turned to Horse-10 and trained the different architectures directly on horse pose estimation from scratch.

Task-based training from scratch

To assess the impact of ImageNet pretraining we also trained several architectures from scratch. Thereby we could directly test if the increased slope for out-of-domain performance across networks was merely a result of more powerful network architectures. He

Figure 5.6: Measuring the impact of common image corruptions on pose estimation (Horse-C): We adapt the image corruptions considered by Hendrycks et al. and contrast the impact of common image corruptions with that of out of domain evaluation.

et al. demonstrated that training Mask R-CNN with ResNet backbones directly on the COCO object detection, instance segmentation and key point detection tasks, *catches-up* with the performance of ImageNet-pretrained variants if training for substantially more iterations than typical training schedules (He et al., 2018). However, due to the nature of these tasks, they could not test this relationship on out-of-domain data. For fine-tuning from ImageNet pretrained models, we trained for $30k$ iterations (as the loss had flattened; see Figure 5.5). First, we searched for best performing schedules for training from scratch while substantially increasing the training time ($6X$ longer). We found that cosine decay with restart was the best for out-of-domain performance (see Methods; Figure 5.5).

Using this schedule, and consistent with He et al. (He et al., 2018), we found that randomly initialized networks could closely match the performance of pretrained networks, given enough data and time (Figure 5.5). As expected, for smaller training sets (5 % training data; 160 images), this was not the case (Figure 5.5). While task-training could therefore match the performance of pretrained networks given enough training data, this was not the case for novel horses (out-of-domain data). The trained from-scratch networks never caught up and indeed plateaued early (Figure 5.5; Figure 5.1). Quantitatively, we also found that for stronger networks (ResNets and EfficientNets) generalization was worse if trained from scratch (Figure 5.1). Interestingly that was not the case for the lightweight models, i.e. MobileNetV2s (cf. (Raghu et al., 2019)).

Network similarity analysis

We hypothesized that the differences in generalization are due to more invariant representations in networks with higher ImageNet-performance using Centered Kernel Alignment (CKA) (Kornblith et al., 2019a). We first verified that the representations change with task training (Supplementary Material Figure 1). We compared the representations of within-domain and out-of-domain images across networks trained from ImageNet vs. from scratch. We found that early blocks are similar for from scratch

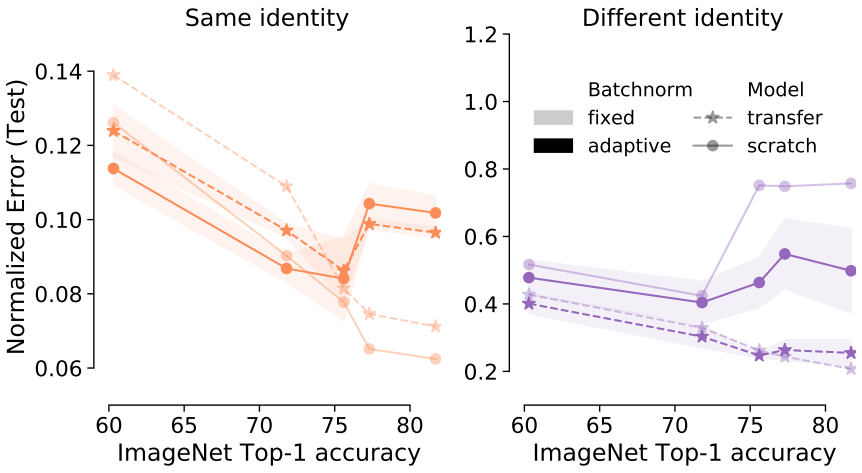


Figure 5.7: Impact of test time normalization. Models trained with adaptive BN layers slightly outperform our baseline models for the MobileNetV2 and ResNet architecture out-of-domain evaluation. Lines with alpha transparency represent fixed models (vs. adapted). We show mean ± SEM computed across 3 data splits.

vs transfer learning for both sets of horses. In later layers, the representations diverge, but comparisons between within-domain and out of domain trends were inconclusive as to why e.g., EfficientNets generalize better (Supplementary Material Figure 2).

Horse-C: Robustness to image corruptions

To elucidate the difficulty of the Horse-10 benchmark, we more broadly evaluate pose estimation performance under different forms of domain shift (Figure 5.6). Recently, Schneider, Rusak et al. demonstrated that simple unsupervised domain adaptation methods can greatly enhance the performance on corruption robustness benchmarks (Schneider et al., 2020b). We therefore settled on a full adaptation evaluation protocol: We re-trained MobileNetV2 0.35 and 1.0, ResNet50, as well as EfficientNet B0 and B3 with batch normalization enabled. During evaluation we then re-computed separate batch norm statistics for each horse and corruption type.

We use batch norm adaptation (Schneider et al., 2020b) during our evaluation on Horse-C. On clean out-of-domain data, we see improvements for MobileNetV2s and ResNets when using pre-trained networks, and for all models when training models from scratch (Figure 5.7). On common corruptions, utilizing adaptation is crucial to final performance (see full results in Supplementary Material). In the batch norm adapted models, we compared four test conditions comprised of within-domain and out-of domain for both original (clean) and corrupted images (Figure 5.6). First, we find that even with batch norm adapted models, Horse-C is as hard as Horse-10; namely performance is significantly affected on corrupted data (Figure 5.8). Secondly, we find the corruption plus "out-of-domain" identity, is even harder—the performance degradation induced by different horse identities is on the same order of magnitude

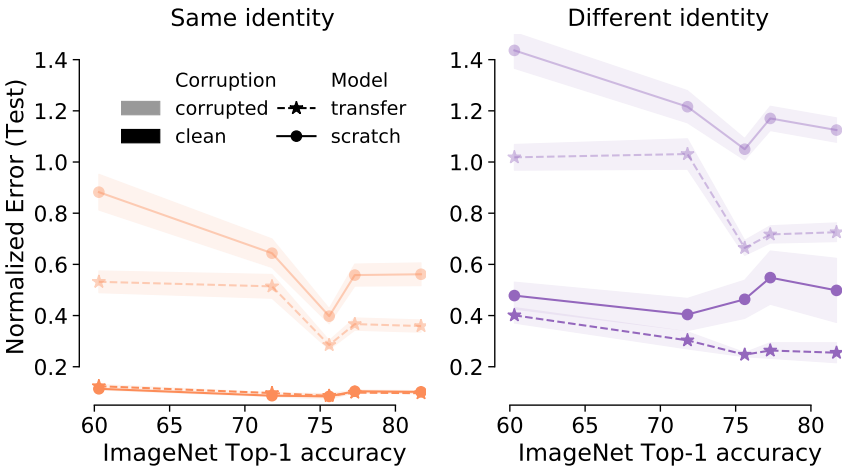


Figure 5.8: Impact of distribution shift introduced by horse identities and common corruptions. We tested within identity (i.e., equivalent to within-domain in Horse-10 (left), or out-of-domain identity (right). Lines with alpha transparency represent corrupted images, whereas solid is the original (clean) image. We show mean ± SEM across 3 splits for clean images, and across 3 splits, 15 corruptions and 5 severities for corrupted images.

as the mean error induced on the corrupted dataset. Finally, and consistent with our other results, we found a performance gain by using pretrained networks (Figure 5.8).

Discussion and conclusions

We developed a novel pose estimation benchmark for out-of-domain robustness (Horse-10), and for testing common image corruptions on pose estimation (Horse-C). The data and benchmarks are available at http://horse10.deeplabcut.org. Furthermore, we report two key findings: (1) pretrained-ImageNet networks offer known advantages: shorter training times, and less data requirements, as well as a novel advantage: robustness on out-of-domain data, and (2) pretained networks that have higher ImageNet performance lead to better generalization. Collectively, this sheds a new light on the inductive biases of "better ImageNet architectures" for visual tasks to be particularly beneficial for robustness.

We introduced novel DeepLabCut model variants, part of https://github.com/DeepLabCut/DeepLabCut, that can achieve high accuracy, but with higher inference speed (up to double) than the original ResNet backbone (see Supplementary Material for an inference speed benchmark).

In summary, transfer learning offers multiple advantages. Not only does pretraining networks on ImageNet allow for using smaller datasets and shorter training time (Figure 5.5), it also strongly improves robustness and generalization, especially for more powerful, over-parameterized models. In fact, we found a strong correlation between generalization and ImageNet accuracy (Figure 5.3). While we found a significant advantage (>2X boost) of using pretrained networks vs. from scratch for out-of-

domain robustness, there is still a 3-fold difference in performance between within domain and out of domain (Figure 5.1). We believe that this work demonstrates that transfer learning approaches are powerful to build robust architectures, and are particularly important for further developing performance improvements on real-world datasets, such as Horse-10 and derived benchmarks such as Horse-C. Furthermore, by sharing our animal pose robustness benchmark dataset, we also believe that the community can collectively work towards closing the gap.

Acknowledgements

Funding was provided by a Rowland Fellowship (MWM), CZI EOSS Grant (MWM & AM), the Bertarelli Foundation (MWM) and the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A; StS & MB). St.S. thanks the International Max Planck Research School for Intelligent Systems and acknowledges his membership in the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program. We thank Maxime Vidal for ground truth corrections to the Animal Pose dataset.

6

Contrastive Learning Inverts the Data Generating Process

The following pages contain the postprint version of the published paper

> Roland S. Zimmermann*, Yash Sharma*, Steffen Schneider*, Matthias Bethge†, and Wieland Brendel†. "Contrastive learning inverts the data generating process." *In International Conference on Machine Learning, pp. 12979-12990. PMLR, 2021.*

A short version of the paper was presented in the NeurIPS 2020 workshop "Self-Supervised Learning – Theory and Practice".

Author Contributions The project was initiated by WB. RSZ, StS and WB jointly derived the theory. RSZ and YS implemented and executed the experiments. The 3DIdent dataset was created by RSZ with feedback from StS, YS, WB and MB. RSZ, YS, StS and WB contributed to the final version of the manuscript. *RSZ, YS and StS contributed equally to the project. †WB and MB contributed equally to advising the project.

Summary

Contrastive learning has recently seen tremendous success in self-supervised learning. So far, however, it is largely unclear why the learned representations generalize so effectively to a large variety of downstream tasks. We here prove that feedforward models trained with objectives belonging to the commonly used InfoNCE family learn to implicitly invert the underlying generative model of the observed data. While the proofs make certain statistical assumptions about the generative model, we observe empirically that our findings hold even if these assumptions are severely violated. Our theory highlights a fundamental connection between contrastive learning, generative modeling, and nonlinear independent component analysis, thereby furthering our understanding of the learned representations as well as providing a theoretical foundation to derive more effective contrastive losses.

Introduction

With the availability of large collections of unlabeled data, recent work has led to significant advances in self-supervised learning. In particular, contrastive methods have been tremendously successful in learning representations for visual and sequential data (Bachman et al., 2019; Baevski et al., 2020a, 2020c; Chen et al., 2020a; He et al., 2020a; Hénaff, 2020; Hjelm et al., 2019; Logeswaran & Lee, 2018; Oord et al., 2018; Ravanelli et al., 2020; Schneider et al., 2019; Tian et al., 2019; Wu et al., 2018). While a number of explanations have been provided as to why contrastive learning leads to such informative representations, existing theoretical predictions and empirical observations appear to be at odds with each other (Bachman et al., 2019; Saunshi et al., 2019; Tian et al., 2019; Wu et al., 2020).

In a nutshell, contrastive methods aim to learn representations where related samples are aligned (positive pairs, e.g. augmentations of the same image), while unrelated samples are separated (negative pairs) (Chen et al., 2020a). Intuitively, this leads to invariance to irrelevant details or transformations (by decreasing the distance between positive pairs), while preserving a sufficient amount of information about the input for solving downstream tasks (by increasing the distance between negative pairs) (Tian et al., 2020). This intuition has recently been made more precise by (Wang & Isola, 2020), showing that a commonly used contrastive loss from the InfoNCE family (Chen et al., 2020a; Gutmann & Hyvärinen, 2012; Oord et al., 2018) asymptotically converges to a sum of two losses: an *alignment* loss that pulls together the representations of positive pairs, and a *uniformity* loss that maximizes the entropy of the learned latent distribution.

We show that an encoder learned with a contrastive loss from the InfoNCE family can recover the true generative factors of variation (up to rotations) if the process that generated the data fulfills a few weak statistical assumptions. This theory bridges the gap between contrastive learning, nonlinear independent component analysis (ICA) and

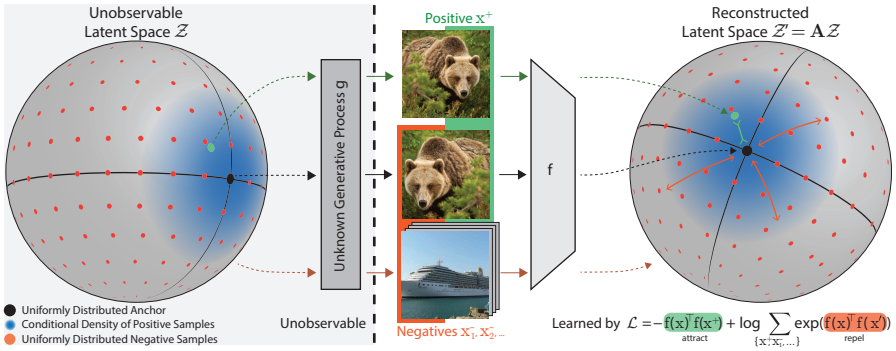


Figure 6.1: We analyze the setup of contrastive learning, in which a feature encoder f is trained with the InfoNCE objective (Chen et al., 2020a; Gutmann & Hyvärinen, 2012; Oord et al., 2018) using positive samples (green) and negative samples (orange). We assume the observations are generated by an (unknown) injective generative model g that maps unobservable latent variables from a hypersphere to observations in another manifold. Under these assumptions, the feature encoder f implictly learns to invert the ground-truth generative process g up to linear transformations, i.e., $f = \mathbf{A}g^{-1}$ with an orthogonal matrix $\mathbf{A}$, if f minimizes the InfoNCE objective.

generative modeling (see Fig. 6.1). Our theory reveals implicit assumptions encoded in the InfoNCE objective about the generative process underlying the data. If these assumptions are violated, we show a principled way of deriving alternative contrastive objectives based on assumptions regarding the positive pair distribution. We verify our theoretical findings with controlled experiments, providing evidence that our theory holds true in practice, even if the assumptions on the ground-truth generative model are partially violated.

To the best of our knowledge, our work is the first to analyze under what circumstances representation learning methods used in practice provably represent the data in terms of its underlying factors of variation. Our theoretical and empirical results suggest that the success of contrastive learning in many practical applications is due to an implicit and approximate inversion of the data generating process, which explains why the learned representations are useful in a wide range of downstream tasks.

In summary, our contributions are:

- We establish a theoretical connection between the InfoNCE family of objectives, which is commonly used in self-supervised learning, and nonlinear ICA. We show that training with InfoNCE inverts the data generating process if certain statistical assumptions on the data generating process hold.

- We empirically verify our predictions when the assumed theoretical conditions are fulfilled. In addition, we show successful inversion of the data generating process even if these theoretical assumptions are partially violated.

- We build on top of the CLEVR rendering pipeline (Johnson et al., 2017b) to generate a more visually complex disentanglement benchmark, called *3DIdent*,

that contains hallmarks of natural environments (shadows, different lighting conditions, a 3D object, etc.). We demonstrate that a contrastive loss derived from our theoretical framework can identify the ground-truth factors of such complex, high-resolution images.

Related Work

Contrastive Learning Despite the success of contrastive learning (CL), our understanding of the learned representations remains limited, as existing theoretical explanations yield partially contradictory predictions. One way to theoretically motivate CL is to refer to the InfoMax principle (Linsker, 1988), which corresponds to maximizing the mutual information (MI) between different views (Bachman et al., 2019; Chen et al., 2020a; Hjelm et al., 2019; Oord et al., 2018; Tian et al., 2020). However, as optimizing a tighter bound on the MI can produce worse representations (Tschannen et al., 2020), it is not clear how accurate this motivation describes the behavior of CL.

Another approach aims to explain the success by introducing latent classes (Saunshi et al., 2019). While this theory has some appeal, there exists a gap between empirical observations and its predictions, e.g. the prediction that an excessive number of negative samples decreases performance does not corroborate with empirical results (Chen et al., 2020a; He et al., 2020a; Tian et al., 2019; Wu et al., 2018). However, recent work has suggested some empirical evidence for said theoretical prediction, namely, issues with the commonly used sampling strategy for negative samples, and have proposed ways to mitigate said issues as well (Chuang et al., 2020; Robinson et al., 2020).

More recently, the behavior of CL has been analyzed from the perspective of *alignment* and *uniformity* properties of representations, demonstrating that these two properties are correlated with downstream performance (Wang & Isola, 2020). We build on these results to make a connection to cross-entropy minimization from which we can derive identifiability results.

Nonlinear ICA Independent Components Analysis (ICA) attempts to find the underlying sources for multidimensional data. In the nonlinear case, said sources correspond to a well-defined nonlinear generative model g, which is assumed to be invertible (i.e., injective) (Hyvärinen et al., 2001; Jutten et al., 2010). In other words, nonlinear ICA solves a demixing problem: Given observed data $\mathbf{x} = g(\mathbf{z})$, it aims to find a model f that equals the inverse generative model g^{-1}, which allows for the original sources $\mathbf{z}$ to be recovered.

Hyvärinen et al. (2019a) show that the nonlinear demixing problem can be solved as long as the independent components are conditionally mutually independent with respect to some auxiliary variable. The authors further provide practical estimation methods for solving the nonlinear ICA problem (Hyvärinen & Morioka, 2016, 2017), similar in spirit to noise contrastive estimation (NCE; Gutmann and Hyvärinen, 2012). Recent work has generalized this contribution to VAEs (Khemakhem et al., 2020b;

Klindt et al., 2021a; Locatello et al., 2020), as well as (invertible-by-construction) energy-based models (Khemakhem et al., 2020c). We here extend this line of work to more general feed-forward networks trained using InfoNCE (Oord et al., 2018).

In a similar vein, Roeder et al. (2020b) build on the work of Hyvärinen et al. (2019a) to show that for a model family which includes InfoNCE, distribution matching implies parameter matching. In contrast, we associate the learned latent representation with the ground-truth generative factors, showing under what conditions the data generating process is inverted, and thus, the true latent factors are recovered.

Theory

We will show a connection between contrastive learning and identifiability in the form of nonlinear ICA. For this, we introduce a feature encoder f that maps observations $\mathbf{x}$ to representations. We consider the widely used *InfoNCE* loss, which often assumes L^2 normalized representations (Bachman et al., 2019; Chen et al., 2020a; He et al., 2020b; Tian et al., 2019; Wu et al., 2018),

$$\mathcal{L}_{\mathsf{contr}}(f;\tau,M) \quad := \underset{\substack{(\mathbf{x},\tilde{\mathbf{x}})\sim p_{\mathsf{pos}} \\ \{\mathbf{x}_i^-\}_{i=1}^M \overset{\text{i.i.d.}}{\sim} p_{\mathsf{data}}}}{\mathbb{E}} \left[-\log \frac{e^{f(\mathbf{x})^\top f(\tilde{\mathbf{x}})/\tau}}{e^{f(\mathbf{x})^\top f(\tilde{\mathbf{x}})/\tau} + \sum_{i=1}^{M} e^{f(\mathbf{x})^\top f(\mathbf{x}_i^-)/\tau}} \right]. \tag{6.1}$$

Here $M \in \mathbb{Z}_+$ is a fixed number of negative samples, p_{data} is the distribution of all observations and p_{pos} is the distribution of positive pairs. This loss was motivated by the InfoMax principle (Linsker, 1988), and has been shown to be effective by many recent representation learning methods (Bachman et al., 2019; Baevski et al., 2020c; Chen et al., 2020a; He et al., 2020a; Hjelm et al., 2019; Logeswaran & Lee, 2018; Tian et al., 2019; Wu et al., 2018). Our theoretical results also hold for a loss function whose denominator only consists of the second summand across the negative samples (e.g., the SimCLR loss (Chen et al., 2020a)).

In the spirit of existing literature on nonlinear ICA (Gutmann & Hyvärinen, 2012; Harmeling et al., 2003; Hyvärinen & Morioka, 2016, 2017; Hyvärinen & Pajunen, 1999; Hyvärinen et al., 2019a; Khemakhem et al., 2020b; Sprekeler et al., 2014), we assume that the observations $\mathbf{x} \in \mathcal{X}$ are generated by an invertible (i.e., injective) generative process $g : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^K$ is the space of observations and $\mathcal{Z} \subseteq \mathbb{R}^N$ with $N \leq K$ denotes the space of latent factors. Influenced by the commonly used feature normalization in InfoNCE, we further assume that $\mathcal{Z}$ is the unit hypersphere $\mathbb{S}^{N-1}$ (see Appx. E). Additionally, we assume that the ground-truth marginal distribution of the latents of the generative process is uniform and that the conditional distribution (under

which positive pairs have high density) is a von Mises-Fisher (vMF) distribution:

$$p(\mathbf{z}) = |\mathcal{Z}|^{-1}, \qquad p(\mathbf{z}|\tilde{\mathbf{z}}) = C_p^{-1} e^{\kappa \mathbf{z}^\top \tilde{\mathbf{z}}} \quad \text{with} \tag{6.2}$$
$$C_p := \int e^{\kappa \mathbf{z}^\top \tilde{\mathbf{z}}} \, \mathrm{d}\tilde{\mathbf{z}} = \text{const.}, \quad \mathbf{x} = g(\mathbf{z}), \quad \tilde{\mathbf{x}} = g(\tilde{\mathbf{z}}).$$

Given these assumptions, we will show that if f minimizes the contrastive loss $\mathcal{L}_{\text{contr}}$, then f solves the demixing problem, i.e., inverts g up to orthogonal linear transformations.

Our theoretical approach consists of three steps: (1) We demonstrate that $\mathcal{L}_{\text{contr}}$ can be interpreted as the cross-entropy between the (conditional) ground-truth and inferred latent distribution. (2) Next, we show that encoders minimizing $\mathcal{L}_{\text{contr}}$ maintain distance, i.e., two latent vectors with distance α in the ground-truth generative model are mapped to points with the same distance α in the inferred representation. (3) Finally, we leverage distance preservation to show that minimizers of $\mathcal{L}_{\text{contr}}$ invert the generative process up to orthogonal transformations. Detailed proofs are given in Appx. E.

Additionally, we will present similar results for general convex bodies in $\mathbb{R}^N$ and more general similarity measures, see Sec. 6. For this, the detailed proofs are given in Appx. E.

Contrastive learning is related to cross-entropy minimization

From the perspective of nonlinear ICA, we are interested in understanding how the representations $f(\mathbf{x})$ which minimize the contrastive loss $\mathcal{L}_{\text{contr}}$ (defined in Eq. (6.1)) are related to the ground-truth source signals $\mathbf{z}$. To study this relationship, we focus on the map $h = f \circ g$ between the recovered source signals $h(\mathbf{z})$ and the true source signals $\mathbf{z}$. Note that this is merely for mathematical convenience; it does not necessitate knowledge regarding neither g nor the ground-truth factors during learning (beyond the assumptions stated in the theorems).

A core insight is a connection between the contrastive loss and the cross-entropy between the ground-truth latent distribution and a certain model distribution. For this, we expand the theoretical results obtained by Wang and Isola (2020):

Theorem 3 ($\mathcal{L}_{\text{contr}}$ converges to the cross-entropy between latent distributions)**.** *If the ground-truth marginal distribution p is uniform, then for fixed $\tau > 0$, as the number of negative samples $M \to \infty$, the (normalized) contrastive loss converges to*

$$\lim_{M \to \infty} \mathcal{L}_{\text{contr}}(f; \tau, M) - \log M + \log |\mathcal{Z}| = \underset{\mathbf{z} \sim p(\mathbf{z})}{\mathbb{E}} \left[H(p(\cdot|\mathbf{z}), q_h(\cdot|\mathbf{z})) \right] \tag{6.3}$$

where H is the cross-entropy between the ground-truth conditional distribution p over positive

pairs and a conditional distribution q_{h} parameterized by the model f,

$$\begin{aligned} q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_h(\mathbf{z})^{-1} e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \\ \text{with} \quad C_h(\mathbf{z}) &:= \int e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \, \mathrm{d}\tilde{\mathbf{z}}, \end{aligned} \tag{6.4}$$

where $C_h(\mathbf{z}) \in \mathbb{R}^+$ is the partition function of q_{h} (see Appx. E).

Next, we show that the minimizers h^* of the cross-entropy (6.4) are isometries in the sense that $\kappa \mathbf{z}^\top \tilde{\mathbf{z}} = h^*(\mathbf{z})^\top h^*(\tilde{\mathbf{z}})$ for all $\mathbf{z}$ and $\tilde{\mathbf{z}}$. In other words, they preserve the dot product between $\mathbf{z}$ and $\tilde{\mathbf{z}}$.

Proposition 4 (Minimizers of the cross-entropy maintain the dot product)**.** *Let $\mathcal{Z} = \mathbb{S}^{N-1}$, $\tau > 0$ and consider the ground-truth conditional distribution of the form $p(\tilde{\mathbf{z}}|\mathbf{z}) = C_p^{-1} \exp(\kappa \tilde{\mathbf{z}}^\top \mathbf{z})$. Let h map onto a hypersphere with radius $\sqrt{\tau\kappa}$.[1] Consider the conditional distribution q_h parameterized by the model, as defined above in Theorem 3, where the hypothesis class for h (and thus f) is assumed to be sufficiently flexible such that $p(\tilde{\mathbf{z}}|\mathbf{z})$ and $q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ can match. If h is a minimizer of the cross-entropy $\mathbb{E}_{p(\tilde{\mathbf{z}}|\mathbf{z})}[-\log q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})]$, then $p(\tilde{\mathbf{z}}|\mathbf{z}) = q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ and $\forall \mathbf{z}, \tilde{\mathbf{z}} : \kappa \mathbf{z}^\top \tilde{\mathbf{z}} = h(\mathbf{z})^\top h(\tilde{\mathbf{z}})$.*

Contrastive learning identifies ground-truth factors on the hypersphere

From the strong geometric property of isometry, we can now deduce a key property of the minimizers h^*:

Proposition 5 (Extension of the Mazur-Ulam theorem to hyperspheres and the dot product)**.** *Let $\mathcal{Z} = \mathbb{S}^{N-1}$ and $\mathcal{Z}' = \mathbb{S}_r^{N-1}$ be the hyperspheres with radius 1 and $r > 0$, respectively. If $h : \mathbb{R}^N \to \mathcal{Z}'$ is differentiable in the vicinity of $\mathcal{Z}$ and its restriction to $\mathcal{Z}$ maintains the dot product up to a constant factor, i.e., $\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z} : r^2 \mathbf{z}^\top \tilde{\mathbf{z}} = h(\mathbf{z})^\top h(\tilde{\mathbf{z}})$, then h is an orthogonal linear transformation scaled by r for all $\mathbf{z} \in \mathcal{Z}$.*

In the last step, we combine the previous propositions to derive our main result: the minimizers of the contrastive loss $\mathcal{L}_{\mathsf{contr}}$ solve the demixing problem of nonlinear ICA up to linear transformations, i.e., they identify the original sources $\mathbf{z}$ for observations $g(\mathbf{z})$ up to orthogonal linear transformations. For a hyperspherical space $\mathcal{Z}$ these correspond to combinations of permutations, rotations and sign flips.

Theorem 4. *Let $\mathcal{Z} = \mathbb{S}^{N-1}$, the ground-truth marginal be uniform, and the conditional a vMF distribution (cf. Eq. 6.2). Let the restriction of the mixing function g to $\mathcal{Z}$ be injective and h be differentiable in a vicinity of $\mathcal{Z}$. If the assumed form of q_{h}, as defined above, matches that of p, and if f is differentiable and minimizes the CL loss as defined in Eq. (6.1), then for fixed $\tau > 0$ and $M \to \infty$, $h = f \circ g$ is linear, i.e., f recovers the latent sources up to an orthogonal linear transformation and a constant scaling factor.*

[1] Note that in practice this can be implemented as a learnable rescaling operation as the last operation of the network f.

Note that we do not assume knowledge of the ground-truth generative model g; we only make assumptions about the conditional and marginal distribution of the latents. On real data, it is unlikely that the assumed model distribution q_{h} can exactly match the ground-truth conditional. We do, however, provide empirical evidence that h is still an affine transformation even if there is a severe mismatch, see Sec. 6.

Contrastive learning identifies ground-truth factors on convex bodies in $\mathbb{R}^N$

While the previous theoretical results require $\mathcal{Z}$ to be a hypersphere, we will now show a similar theorem for the more general case of $\mathcal{Z}$ being a convex body in $\mathbb{R}^N$. Note that the hyperrectangle $[a_1, b_1] \times \ldots \times [a_N, b_N]$ is an example of such a convex body.

We follow a similar three step proof strategy as for the hyperspherical case before: (1) We begin again by showing that a properly chosen contrastive loss on convex bodies corresponds to the cross-entropy between the ground-truth conditional and a distribution parametrized by the encoder. For this step, we additionally extend the results of Wang and Isola (2020) to this latent space and loss function. (2) Next, we derive that minimizers of the loss function are isometries of the latent space. Importantly, we do not limit ourselves to a specific metric, thus the result is applicable to a family of contrastive objectives. (3) Finally, we show that these minimizers must be affine transformations. For a special family of conditional distributions (rotationally asymmetric generalized normal distributions (Subbotin, 1923)), we can further narrow the class of solutions to permutations and sign-flips. For the detailed proofs, see Appx. E.

As earlier, we assume that the ground-truth marginal distribution of the latents is uniform. However, we now assume that the conditional distribution is exponential:

$$\begin{aligned} p(\mathbf{z}) = |\mathcal{Z}|^{-1}, \qquad p(\mathbf{z}|\tilde{\mathbf{z}}) = C_p^{-1} e^{-\delta(\mathbf{z},\tilde{\mathbf{z}})} \quad \text{with} \\ C_p(\mathbf{z}) := \int e^{-\delta(\mathbf{z},\tilde{\mathbf{z}})} \, \mathrm{d}\tilde{\mathbf{z}}, \quad \mathbf{x} = g(\mathbf{z}), \quad \tilde{\mathbf{x}} = g(\tilde{\mathbf{z}}), \end{aligned} \tag{6.5}$$

where δ is a metric induced by a norm (see Appx. E).

To reflect the differences between this conditional distribution and the one assumed for the hyperspherical case, we need to introduce an adjusted version of the contrastive loss in (6.1):

Definition 1 ($\mathcal{L}_{\delta\text{-contr}}$ objective)**.** *Let $\delta : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ be a metric on $\mathcal{Z}$. We define the general InfoNCE loss, which uses δ as a similarity measure, as*

$$\mathcal{L}_{\delta\text{-contr}}(f; \tau, M) \quad := \quad \mathop{\mathbb{E}}_{\substack{(\mathbf{x},\tilde{\mathbf{x}}) \sim p_{\mathsf{pos}} \\ \{\mathbf{x}_i^-\}_{i=1}^M \overset{i.i.d.}{\sim} p_{\mathsf{data}}}} \left[-\log \frac{e^{-\delta(f(\mathbf{x}), f(\tilde{\mathbf{x}}))/\tau}}{e^{-\delta(f(\mathbf{x}), f(\tilde{\mathbf{x}}))/\tau} + \sum_{i=1}^{M} e^{-\delta(f(\mathbf{x}), f(\mathbf{x}_i^-))/\tau}} \right]. \tag{6.6}$$

Note that this is a generalization of the InfoNCE criterion in Eq. (6.1). In contrast to

the objective above, the representations are no longer assumed to be L^2 normalized, and the dot-product is replaced with a more general similarity measure δ.

Analogous to the previously demonstrated case for the hypersphere, for convex bodies $\mathcal{Z}$, minimizers of the adjusted $\mathcal{L}_{\delta\text{-contr}}$ objective solve the demixing problem of nonlinear ICA up to invertible linear transformations:

Theorem 7. *Let $\mathcal{Z}$ be a convex body in $\mathbb{R}^N$, $h = f \circ g : \mathcal{Z} \to \mathcal{Z}$, and δ be a metric or a semi-metric (cf. Lemma 4 in Appx. E), induced by a norm. Further, let the ground-truth marginal distribution be uniform and the conditional distribution be as Eq. (6.5). Let the mixing function g be differentiable and injective. If the assumed form of q_{h} matches that of p, i.e.,*

$$\begin{aligned} q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_q^{-1}(\mathbf{z}) e^{-\delta(h(\tilde{\mathbf{z}}),h(\mathbf{z}))/\tau} \\ \text{with} \quad C_q(\mathbf{z}) &:= \int e^{-\delta(h(\tilde{\mathbf{z}}),h(\mathbf{z}))/\tau} \, \mathrm{d}\tilde{\mathbf{z}}, \end{aligned} \tag{6.7}$$

and if f is differentiable and minimizes the $\mathcal{L}_{\delta\text{-contr}}$ objective in Eq. (6.6) for $M \to \infty$, we find that $h = f \circ g$ is invertible and affine, i.e., we recover the latent sources up to affine transformations.

Note that the model distribution q_{h}, which is implicitly described by the choice of the objective, must be of the same form as the ground-truth distribution p, i.e., both must be based on the same metric. Thus, identifying different ground-truth conditional distributions requires different contrastive $\mathcal{L}_{\delta\text{-contr}}$ objectives. This result can be seen as a generalized version of Theorem 4, as it is valid for any convex body $\mathcal{Z} \subseteq \mathbb{R}^N$, allowing for a larger variety of conditional distributions.

Finally, under the mild restriction that the ground-truth conditional distribution is based on an L^p similarity measure for $p \geq 1, p \neq 2$, h identifies the ground-truth generative factors up to generalized permutations. A generalized permutation matrix $\mathbf{A}$ is a combination of a permutation and element-wise sign-flips, i.e., $\forall \mathbf{z} : (\mathbf{A}\mathbf{z})_{\mathbf{i}} = \mathbf{f}\mathbf{f}_{\mathbf{i}}\mathbf{z}_{\boldsymbol{\sigma}(\mathbf{i})}$ with $\alpha_i = \pm 1$ and σ being a permutation.

Theorem 8. *Let $\mathcal{Z}$ be a convex body in $\mathbb{R}^N$, $h : \mathcal{Z} \to \mathcal{Z}$, and δ be an L^α metric or semi-metric (cf. Lemma 4 in Appx. E) for $\alpha \geq 1, \alpha \neq 2$. Further, let the ground-truth marginal distribution be uniform and the conditional distribution be as Eq. (6.5), and let the mixing function g be differentiable and invertible. If the assumed form of $q_{\mathrm{h}}(\cdot|\mathbf{z})$ matches that of $p(\cdot|\mathbf{z})$, i.e., both use the same metric δ up to a constant scaling factor, and if f is differentiable and minimizes the $\mathcal{L}_{\delta\text{-contr}}$ objective in Eq. (6.6) for $M \to \infty$, we find that $h = f \circ g$ is a composition of input independent permutations, sign flips and rescaling.*

Experiments

Validation of theoretical claim

We validate our theoretical claims under both perfectly matching and violated conditions regarding the ground-truth marginal and conditional distributions. We consider

Table 6.1: Identifiability up to affine transformations. Mean ± standard deviation over 5 random seeds. Note that only the first row corresponds to a setting that matches (✓) our theoretical assumptions, while the others show results for violated assumptions (✗; see column *M.*). Note that the identity score only depends on the ground-truth space and the marginal distribution defined for the generative process, while the supervised score additionally depends on the space assumed by the model.

Generative process g			Model f			R^2 Score [%]		
Space	$p(\cdot)$	$p(\cdot\|\cdot)$	Space	$q_h(\cdot\|\cdot)$	M.	Identity	Supervised	Unsupervised
Sphere	Uniform	vMF(κ=1)	Sphere	vMF(κ=1)	✓	66.98 ± 2.79	99.71 ± 0.05	99.42 ± 0.05
Sphere	Uniform	vMF(κ=10)	Sphere	vMF(κ=1)	✗	—"—	—"—	99.86 ± 0.01
Sphere	Uniform	Laplace(λ=0.05)	Sphere	vMF(κ=1)	✗	—"—	—"—	99.91 ± 0.01
Sphere	Uniform	Normal(σ=0.05)	Sphere	vMF(κ=1)	✗	—"—	—"—	99.86 ± 0.00
Box	Uniform	Normal(σ=0.05)	Unbounded	Normal	✗	67.93 ± 7.40	99.78 ± 0.06	99.60 ± 0.02
Box	Uniform	Laplace(λ=0.05)	Unbounded	Normal	✗	—"—	—"—	99.64 ± 0.02
Box	Uniform	Laplace(λ=0.05)	Unbounded	GenNorm(β=3)	✗	—"—	—"—	99.70 ± 0.02
Box	Uniform	Normal(σ=0.05)	Unbounded	GenNorm(β=3)	✗	—"—	—"—	99.69 ± 0.02
Sphere	Normal(σ=1)	Laplace(λ=0.05)	Sphere	vMF(κ=1)	✗	63.37 ± 2.41	99.70 ± 0.07	99.02 ± 0.01
Sphere	Normal(σ=1)	Normal(σ=0.05)	Sphere	vMF(κ=1)	✗	—"—	—"—	99.02 ± 0.02
Unbounded	Laplace(λ=1)	Normal(σ=1)	Unbounded	Normal	✗	62.49 ± 1.65	99.65 ± 0.04	98.13 ± 0.14
Unbounded	Normal(σ=1)	Normal(σ=1)	Unbounded	Normal	✗	63.57 ± 2.30	99.61 ± 0.17	98.76 ± 0.03

source signals of dimensionality $N = 10$, and sample pairs of source signals in two steps: First, we sample from the marginal $p(\mathbf{z})$. For this, we consider both uniform distributions which match our assumptions and non-uniform distributions (e.g., a normal distribution) which violate them. Second, we generate the positive pair by sampling from a conditional distribution $p(\tilde{\mathbf{z}}|\mathbf{z})$. Here, we consider matches with our assumptions on the conditional distribution (von Mises-Fisher for $\mathcal{Z} = \mathbb{S}^{N-1}$) as well as violations (e.g. normal, Laplace or generalized normal distribution for $\mathcal{Z} = \mathbb{S}^{N-1}$). Further, we consider spaces beyond the hypersphere, such as the bounded box (which is a convex body) and the unbounded $\mathbb{R}^N$.

We generate the observations with a multi-layer perceptron (MLP), following previous work (Hyvärinen & Morioka, 2016, 2017). Specifically, we use three hidden layers with leaky ReLU units and random weights; to ensure that the MLP g is invertible, we control the condition number of the weight matrices. For our feature encoder f, we also use an MLP with leaky ReLU units, where the assumed space is denoted by the normalization, or lack thereof, of the encoding. Namely, for the hypersphere (denoted as *Sphere*) and the hyperrectangle (denoted as *Box*) we apply an L^2 and L^∞ normalization, respectively. For flexibility in practice, we parameterize the normalization magnitude of the *Box*, including it as part of the encoder's learnable parameters. On the hypersphere we optimize $\mathcal{L}_{\mathsf{contr}}$ and on the hyperrectangle as well as the unbounded space we optimize $\mathcal{L}_{\delta\text{-}\mathsf{contr}}$. For further details, see Appx. E.

To test for identifiability up to affine transformations, we fit a linear regression between the ground-truth and recovered sources and report the coefficient of determination (R^2). To test for identifiability up to generalized permutations, we leverage the mean correlation coefficient (MCC), as used in previous work (Hyvärinen & Morioka, 2016, 2017). For further details, see Appx. E.

We evaluate both identifiability metrics for three different model types. First,

Table 6.2: Identifiability up to generalized permutations, averaged over 5 runs. Note that while Theorem 8 requires the model latent space to be a convex body and $p(\cdot|\cdot) = q_{\mathrm{h}}(\cdot|\cdot)$, we find that empirically either is sufficient. The results are grouped in four blocks corresponding to different types and degrees of violation of assumptions of our theory showing identifiability up to permutations: (1) no violation, violation of the assumptions on either the (2) space or (3) the conditional distribution, or (4) both.

Generative process g			Model f			MCC Score [%]		
Space	$p(\cdot)$	$p(\cdot\|\cdot)$	Space	$q_{\mathrm{h}}(\cdot\|\cdot)$	M.	Identity	Supervised	Unsupervised
Box	Uniform	Laplace(λ=0.05)	Box	Laplace	✓	46.55 ± 1.34	99.93 ± 0.03	98.62 ± 0.05
Box	Uniform	GenNorm(β=3; λ=0.05)	Box	GenNorm(β=3)	✓	—"—	—"—	99.90 ± 0.06
Box	Uniform	Normal(σ=0.05)	Box	Normal	✗	—"—	—"—	99.77 ± 0.01
Box	Uniform	Laplace(λ=0.05)	Box	Normal	✗	—"—	—"—	99.76 ± 0.02
Box	Uniform	GenNorm(β=3; λ=0.05)	Box	Laplace	✗	—"—	—"—	98.80 ± 0.02
Box	Uniform	Laplace(λ=0.05)	Unbounded	Laplace	✗	—"—	99.97 ± 0.03	98.57 ± 0.02
Box	Uniform	GenNorm(β=3; λ=0.05)	Unbounded	GenNorm(β=3)	✗	—"—	—"—	99.85 ± 0.01
Box	Uniform	Normal(σ=0.05)	Unbounded	Normal	✗	—"—	—"—	58.26 ± 3.00
Box	Uniform	Laplace(λ=0.05)	Unbounded	Normal	✗	—"—	—"—	59.67 ± 2.33
Box	Uniform	Normal(σ=0.05)	Unbounded	GenNorm(β=3)	✗	—"—	—"—	43.80 ± 2.15

we ensure that the problem requires nonlinear demixing by considering the identity function for model f, which amounts to scoring the observations against the sources (**Identity Model**). Second, we ensure that the problem is solvable within our model class by training our model f with supervision, minimizing the mean-squared error between $f(g(\mathbf{z}))$ and $\mathbf{z}$ (**Supervised Model**). Third, we fit our model without supervision using a contrastive loss (**Unsupervised Model**).

Tables 6.1 and 6.2 show results evaluating identifiability up to affine transformations and generalized permutations, respectively. When assumptions match (see column M.), CL recovers a score close to the empirical upper bound. Mismatches in assumptions on the marginal and conditional do not lead to a significant drop in performance with respect to affine identifiability, but do for permutation identifiability compared to the empirical upper bound. In many practical scenarios, we use the learned representations to solve a downstream task, thus, identifiability up to affine transformations is often sufficient. However, for applications where identification of the individual generative factors is desirable, some knowledge of the underlying generative process is required to choose an appropriate loss function and feature normalization. Interestingly, we find that for convex bodies, we obtain identifiability up to permutation even in the case of a normal conditional, which likely is due to the axis-aligned box geometry of the latent domain. Finally, note that the drop in performance for identifiability up to permutations in the last group of Tab. 6.2 is a natural consequence of either the ground-truth or the assumed conditional being rotationally symmetric, e.g., a normal distribution, in an unbounded space. Here, rotated versions of the latent space are indistinguishable and, thus, the model cannot align the axes of the reconstruction with that of the ground-truth latent space, resulting in a lower score.

To zoom in on how violations of the uniform marginal assumption influence the identifiability achieved by a model in practice, we perform an ablation on the marginal distribution by interpolating between the theoretically assumed uniform distribution

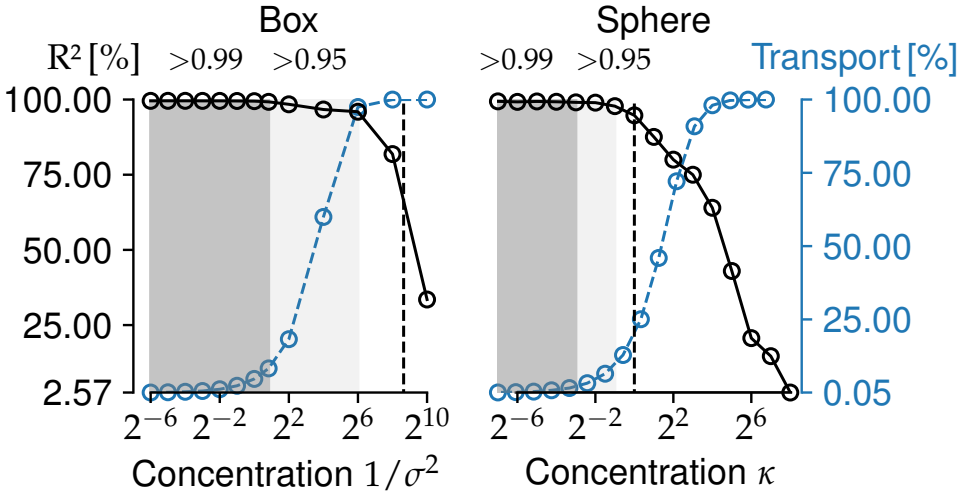


Figure 6.2: Varying degrees of violation of the uniformity assumption for the marginal distribution. The figure shows the R^2 score measuring identifiability up to linear transformations (black) as well as the difference between the used marginal and assumed uniform distribution in terms of probability mass (blue) as a function of the marginal's concentration. The black dotted line indicates the concentration of the used conditional distribution.

and highly locally concentrated distributions. In particular, we consider two cases: (1) a sphere ($\mathcal{S}^9$) with a vMF marginal around its north pole for different concentration parameters κ; (2) a box ($[0,1]^{10}$) with a normal marginal around the box's center for different standard deviations σ. For both cases, Fig. 6.2 shows the R^2 score as a function of the concentration κ and $1/\sigma^2$ respectively (black). As a reference, the concentration of the used conditional distribution is highlighted as a dashed line. In addition, we also display the probability mass (0–100%) that needs to be moved for converting the used marginal distribution (i.e., vMF or normal) into the assumed uniform marginal distribution (blue) as an intuitive measure of the mismatch (i.e., $\frac{1}{2}\int |p(\mathbf{z}) - p_{\text{uni}}|\, \mathrm{d}\mathbf{z}$). While, we observe significant robustness to mismatch, in both cases, we see performance drop drastically once the marginal distribution is more concentrated than the conditional distribution of positive pairs. In such scenarios, positive pairs are indistinguishable from negative pairs.

Extensions to image data

Previous studies have demonstrated that representation learning using contrastive learning scales well to complex natural image data (Chen et al., 2020c, 2020a; Hénaff, 2020). Unfortunately, the true generative factors of natural images are inaccessible, thus we cannot evaluate identifiability scores.

We consider two alternatives. First, we evaluate on the recently proposed benchmark *KITTI Masks* (Klindt et al., 2021a), which is composed of segmentation masks of natural videos. Second, we contribute a novel benchmark (*3DIdent*; cf. Fig. 6.3) which features aspects of natural scenes, e.g. a complex 3D object and different lighting conditions,

Table 6.3: KITTI Masks. Mean $\pm$ standard deviation over 10 random seeds. $\overline{\Delta t}$ indicates the average temporal distance of frames used.

	Model	Model Space	MCC [%]
$\overline{\Delta t} = 0.05s$	SlowVAE	Unbounded	66.1 ± 4.5
	Laplace	Unbounded	77.1 ± 1.0
	Laplace	Box	74.1 ± 4.4
	Normal	Unbounded	58.3 ± 5.4
	Normal	Box	59.9 ± 5.5
$\overline{\Delta t} = 0.15s$	SlowVAE	Unbounded	79.6 ± 5.8
	Laplace	Unbounded	79.4 ± 1.9
	Laplace	Box	80.9 ± 3.8
	Normal	Unbounded	60.2 ± 8.7
	Normal	Box	68.4 ± 6.7

while still providing access to the continuous ground-truth factors. For further details, see Appx. E. *3DIdent* is available at zenodo.org/record/4502485.

KITTI Masks KITTI Masks (Klindt et al., 2021a) is composed of pedestrian segmentation masks extracted from an autonomous driving vision benchmark KITTI-MOTS (Geiger et al., 2012), with natural shapes and continuous natural transitions. We compare to SlowVAE (Klindt et al., 2021a), the state-of-the-art on the considered dataset. In our experiments, we use the same training hyperparameters (for details see Appx. E) and (encoder) architecture as Klindt et al. (2021a). The positive pairs consist of nearby frames with a time separation $\overline{\Delta t}$.

As argued and shown in Klindt et al. (2021a), the transitions in the ground-truth latents between nearby frames is sparse. Unsurprisingly then, Table 6.3 shows that assuming a Laplace conditional as opposed to a normal conditional in the contrastive loss leads to better identification of the underlying factors of variation. SlowVAE also assumes a Laplace conditional (Klindt et al., 2021a) but appears to struggle if the frames of a positive pair are too similar ($\overline{\Delta t} = 0.05s$). This degradation in performance is likely due to the limited expressiveness of the decoder deployed in SlowVAE.

3DIdent

Dataset description We build on (Johnson et al., 2017b) and use the Blender rendering engine (Blender Online Community, 2021) to create visually complex 3D images (see Fig. 6.3). Each image in the dataset shows a colored 3D object which is located and rotated above a colored ground in a 3D space. Additionally, each scene contains a colored spotlight focused on the object and located on a half-circle around the scene. The observations are encoded with an RGB color space, and the spatial resolution is 224×224 pixels.

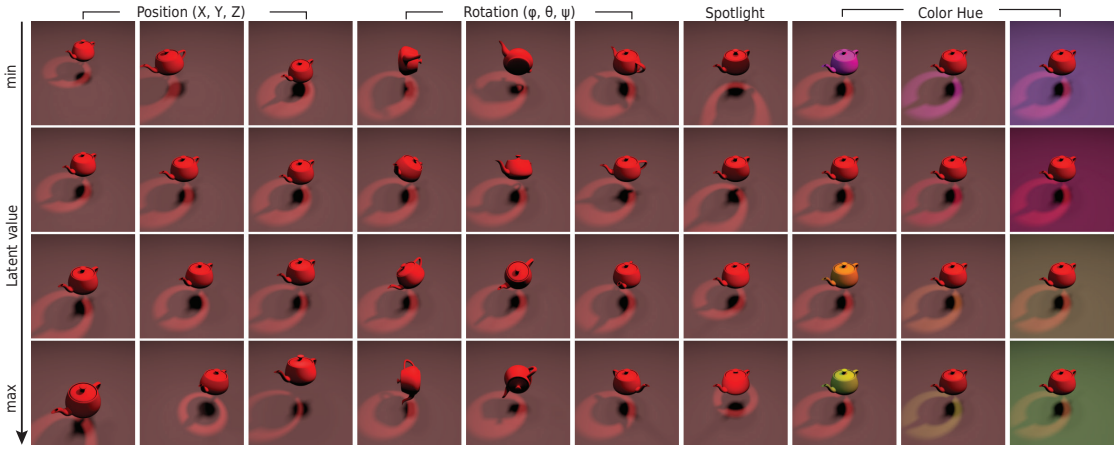


Figure 6.3: 3DIdent. Influence of the latent factors $\mathbf{z}$ on the renderings $\mathbf{x}$. Each column corresponds to a traversal in one of the ten latent dimensions while the other dimensions are kept fixed.

The images are rendered based on a 10-dimensional latent, where: (1) three dimensions describe the XYZ position, (2) three dimensions describe the rotation of the object in Euler angles, (3) two dimensions describe the color of the object and the ground of the scene, respectively, and (4) two dimensions describe the position and color of the spotlight. We use the HSV color space to describe the color of the object and the ground with only one latent each by having the latent factor control the hue value. For more details on the dataset see Sec. E.

The dataset contains 250 000 observation-latent pairs where the latents are uniformly sampled from the hyperrectangle $\mathcal{Z}$. To sample positive pairs $(\mathbf{z}, \tilde{\mathbf{z}})$ we first sample a value $\tilde{\mathbf{z}}'$ from the data conditional $p(\tilde{\mathbf{z}}'|\mathbf{z})$, and then use nearest-neighbor matching[2] implemented by FAISS (Johnson et al., 2017a) to find the latent $\tilde{\mathbf{z}}$ closest to $\tilde{\mathbf{z}}'$ (in L^2 distance) for which there exists an image rendering. In addition, unlike previous work (Locatello et al., 2019b), we create a hold-out test set with 25 000 distinct observation-latent pairs.

Experiments and Results We train a convolutional feature encoder f composed of a ResNet18 architecture (He et al., 2016b) and an additional fully-connected layer, with a LeakyReLU nonlinearity as the hidden activation. For more details, see Appx. E. Following the same methodology as in Sec. 6, i) depending on the assumed space, the output of the feature encoder is normalized accordingly and ii) in addition to the CL models, we also train a supervised model to serve as an upper bound on performance. We consider normal and Laplace distributions for positive pairs. Note, that due to the finite dataset size we only sample from an approximation of these distributions.

As in Tables 6.1 and 6.2, the results in Table 6.4 demonstrate that CL reaches

[2]We used an Inverted File Index (IVF) with Hierarchical Navigable Small World (HNSW) graph exploration for fast indexing.

Table 6.4: Identifiability up to affine transformations on the test set of 3DIdent. Mean ± standard deviation over 3 random seeds. As earlier, only the first row corresponds to a setting that matches the theoretical assumptions for linear identifiability; the others show distinct violations. Supervised training with unbounded space achieves scores of $R^2 = (98.67 \pm 0.03)\%$ and MCC $= (99.33 \pm 0.01)\%$. The last row refers to using the image augmentations suggested by Chen et al. (2020a) to generate positive image pairs. For performance on the training set, see Appx. Table E.1.

Dataset	Model f			Identity [%]	Unsupervised [%]	
$p(\cdot\|\cdot)$	Space	$q_{\mathrm{h}}(\cdot\|\cdot)$	M.	R^2	R^2	MCC
Normal	Box	Normal	✓	5.25 ± 1.20	96.73 ± 0.10	98.31 ± 0.04
Normal	Unbounded	Normal	✗	—"—	96.43 ± 0.03	54.94 ± 0.02
Laplace	Box	Normal	✗	—"—	96.87 ± 0.08	98.38 ± 0.03
Normal	Sphere	vMF	✗	—"—	65.74 ± 0.01	42.44 ± 3.27
Augm.	Sphere	vMF	✗	—"—	45.51 ± 1.43	46.34 ± 1.59

scores close to the topline (supervised) performance, and mismatches between the assumed and ground-truth conditional distribution do not harm the performance significantly. However, if the hypothesis class of the encoder is too restrictive to model the ground-truth conditional distribution, we observe a clear drop in performance, i.e., mapping a box onto a sphere. Note, that this corresponds to the InfoNCE objective for L^2-normalized representations, commonly used for self-supervised representation learning (Bachman et al., 2019; Chen et al., 2020a; He et al., 2020b; Tian et al., 2019; Wu et al., 2018). Finally, the last result shows that leveraging image augmentations (Chen et al., 2020a) as opposed to sampling from a specified conditional distribution of positive pairs $p(\cdot|\cdot)$ results in a performance drop. For details on the experiment, see Appx. Sec. E. We explain this with the greater mismatch between the conditional distribution assumed by the model and the conditional distribution induced by the augmentations. In all, we demonstrate validation of our theoretical claims even for generative processes with higher visual complexity than those considered in Sec. 6.

Conclusion

We showed that objectives belonging to the InfoNCE family, the basis for a number of state-of-the-art techniques in self-supervised representation learning, can uncover the true generative factors of variation underlying the observational data. To succeed, these objectives implicitly encode a few weak assumptions about the statistical nature of the underlying generative factors. While these assumptions will likely not be exactly matched in practice, we showed empirically that the underlying factors of variation are identified even if theoretical assumptions are severely violated.

Our theoretical and empirical results suggest that the representations found with contrastive learning implicitly (and approximately) invert the generative process of the data. This could explain why the learned representations are so useful in many downstream tasks. It is known that a decisive aspect of contrastive learning is the right

choice of augmentations that form a positive pair. We hope that our framework might prove useful for clarifying the ways in which certain augmentations affect the learned representations, and for finding improved augmentation schemes.

Furthermore, our work opens avenues for constructing more effective contrastive losses. As we demonstrate, imposing a contrastive loss informed by characteristics of the latent space can considerably facilitate inferring the correct semantic descriptors, and thus boost performance in downstream tasks. While our framework already allows for a variety of *conditional* distributions, it is an interesting open question how to adapt it to *marginal* distributions beyond the uniform implicitly encoded in InfoNCE. Also, future work may extend our theoretical framework by incorporating additional assumptions about our visual world, such as compositionality, hierarchy or objectness. Accounting for such inductive biases holds enormous promise in forming the basis for the next generation of self-supervised learning algorithms.

Taken together, we lay a strong theoretical foundation for not only understanding but extending the success of state-of-the-art self-supervised learning techniques.

Acknowledgements

We thank Muhammad Waleed Gondal, Ivan Ustyuzhaninov, David Klindt, Lukas Schott, Luisa Eck, and Kartik Ahuja for helpful discussions. We thank Bozidar Antic, Shubham Krishna and Jugoslav Stojcheski for ideas regarding the design of 3DIdent. We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting RSZ, YS and StS. StS acknowledges his membership in the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program. We acknowledge support from the German Federal Ministry of Education and Research (BMBF) through the Competence Center for Machine Learning (TUE.AI, FKZ 01IS18039A) and the Bernstein Computational Neuroscience Program Tübingen (FKZ: 01GQ1002). WB acknowledges support via his Emmy Noether Research Group funded by the German Science Foundation (DFG) under grant no. BR 6382/1-1 as well as support by Open Philantropy and the Good Ventures Foundation. MB and WB acknowledge funding from the MICrONS program of the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00003.

7

Learnable latent embeddings for joint behavioral and neural analysis

The following pages contain the postprint version of the published paper

> Steffen Schneider*, Jin Hwa Lee*, and Mackenzie Weygandt Mathis. "Learnable latent embeddings for joint behavioural and neural analysis." *Nature (2023): 1-9.*

An extended abstract version of the work was also presented at COSYNE 2023, Montreal.

Author Contributions Conceptualization: MWM, StS; Methodology: StS, JHL, MWM; Software: StS, JHL; Theory: StS; Formal analysis: StS, JHL; Investigation: StS, JHL; Data Curation: JHL, StS; Writing-Original Draft: MWM; Writing-Editing: MWM, StS, JHL. *StS and JHL contributed equally to the project.

Summary

Mapping behavioural actions to neural activity is a fundamental goal of neuroscience. As our ability to record large neural and behavioural data increases, there is growing interest in modelling neural dynamics during adaptive behaviours to probe neural representations. In particular, although neural latent embeddings can reveal underlying correlates of behaviour, we lack nonlinear techniques that can explicitly and flexibly leverage joint behaviour and neural data to uncover neural dynamics. Here, we fill this gap with a new encoding method, CEBRA, that jointly uses behavioural and neural data in a (supervised) hypothesis- or (self-supervised) discovery-driven manner to produce both consistent and high-performance latent spaces. We show that consistency can be used as a metric for uncovering meaningful differences, and the inferred latents can be used for decoding. We validate its accuracy and demonstrate our tool's utility for both calcium and electrophysiology datasets, across sensory and motor tasks and in simple or complex behaviours across species. It allows leverage of single- and multi-session datasets for hypothesis testing or can be used label free. Lastly, we show that CEBRA can be used for the mapping of space, uncovering complex kinematic features, for the production of consistent latent spaces across two-photon and Neuropixels data, and can provide rapid, high-accuracy decoding of natural videos from visual cortex.

Introduction

A central quest in neuroscience is the neural origin of behavior (Krakauer et al., 2017b; Urai et al., 2022b). Yet, we are still limited in both the number of neurons and length of time we can record from behaving animals in a session. Therefore, we need new methods that can combine data across animals and sessions with minimal assumptions, and generate interpretable neural embedding spaces (Jazayeri & Ostojic, 2021; Urai et al., 2022b). Current tools for representation learning are either linear, or if non-linear they typically rely on generative models, and they do not yield consistent embeddings across animals (or repeated runs of the algorithm). Here, we combine recent advances in non-linear disentangled representation learning and self-supervised learning to develop a new dimensionality reduction method that can be applied jointly to behavioral and neural recordings to reveal meaningful lower dimensional neural population dynamics (Humphries, 2021; Jazayeri & Ostojic, 2021; Zhou & Wei, 2020).

From data visualization (clustering) to discovering latent spaces that explain neural variance, dimensionality reduction of behavior or neural data has been impactful in neuroscience. For example, complex 3D forelimb reaching can be reduced to only 8–12 dimensions (Okorokova et al., 2020; Vargas-Irwin et al., 2010), and the low dimensional embeddings reveal some robust aspects of movements (e.g., PCA-based manifolds where the neural state space can easily be constrained and is stable across time (Churchland et al., 2012b; Gallego et al., 2018; Yu et al., 2008b)). Linear methods such as PCA are often used to increase interpretability, but this comes at the cost

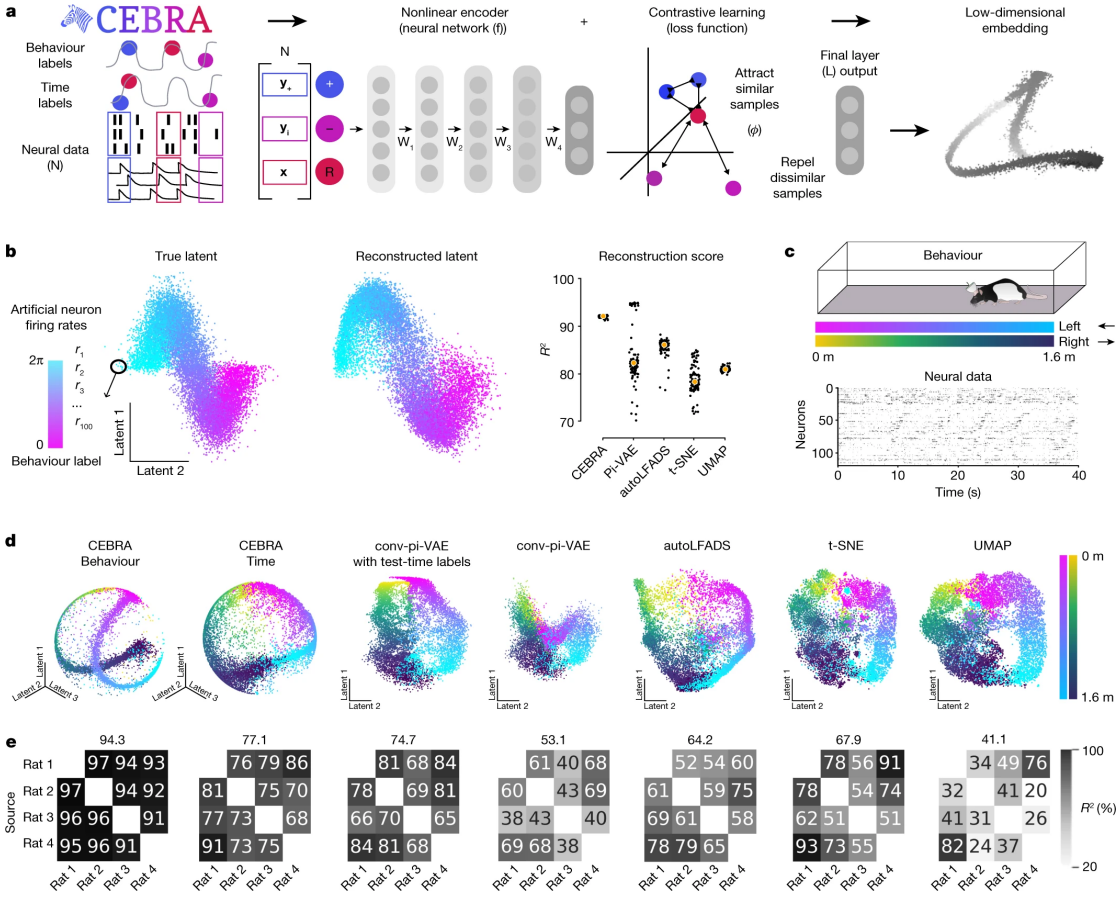


Figure 7.1: **Use of CEBRA for consistent and interpretable embeddings. (a)**: CEBRA allows for self-supervised, supervised, and hybrid approaches for both hypothesis-driven and discovery-driven analysis. Overview of pipeline: collect data (e.g., pairs of behavior (or time) and neural data (x,y)), determine positive and negative pairs, train CEBRA, and produce embeddings. $W_1, ...W_4$ represent the neural network weights. **(b)**: Left: True 2D latent, where each point is mapped to spiking rate of 100 neurons. Middle: CEBRA embedding after linear regression to the true latent. Right: Reconstruction score is R^2 of linear regression between the true latent and resulting embedding from each method. The "behavior label" is a 1D random variable sampled from uniform distribution of [0, 2π] that is assigned to each time bin of synthetic neural data, visualized by the color map. The orange line is the median, and each black dot is an individual run (n=100). CEBRA-Behavior shows significantly higher reconstruction score compare to pi-VAE, tSNE and UMAP (one-way ANOVA, $F(4, 495)=251$, $p=1.12 \times 10^{-117}$ with post hoc Tukey's honest significant difference $p<0.001$). **(c)**: Rat hippocampus data derived from Grosmark and Buzsáki (2016). Electrophysiology data were collected while a rat traversed a 1.6m linear track "leftwards" or "rightwards". **(d)**: We benchmarked CEBRA against conv-pi-VAE (both with labels and without), autoLFADS, tSNE, and unsupervised UMAP. Note, for performance against the original pi-VAE see Extended Data Fig. 7.3. We plot the three latents (all CEBRA embedding figures show the first three latents).The dimensionality of the latent space is set to the minimum and equivalent dimension per method (3D for CEBRA and 2D for others) for fair comparison. Note, higher dimensions for CEBRA can yield higher consistency values (see Extended Data Fig. 7.9). **(e)**: Correlation matrices show R^2 values after fitting a linear model between behavior-aligned embeddings of pairs of rats, one as the target and the other as the source (mean, n=10 runs). Parameters were picked by optimization of average run consistency across rats.

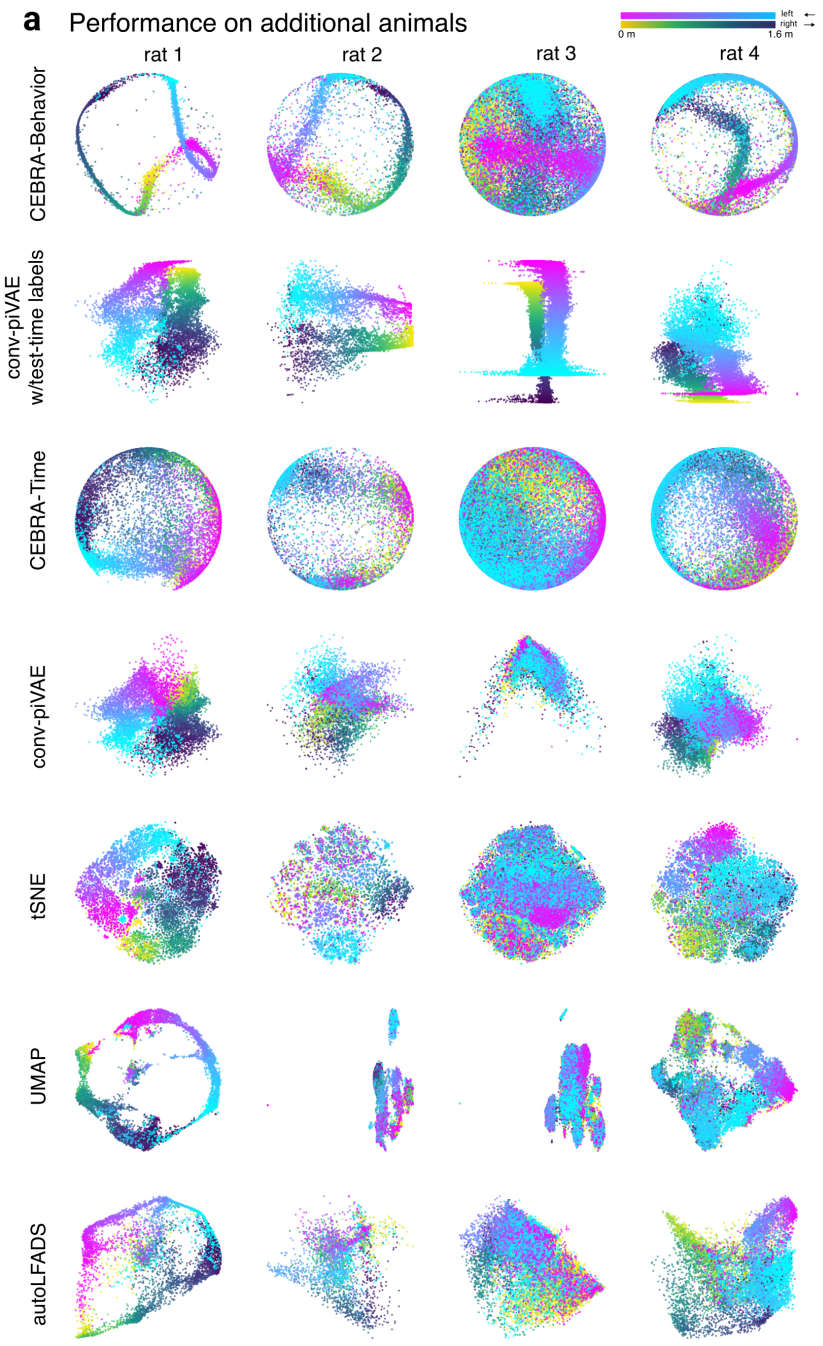


Figure 7.2: **CEBRA produced consistent, highly decodable embeddings (a)**: Additional rat data shown for all algorithms we benchmarked (see Methods). For CEBRA-Behavior, we used temperature 1, time offset 10, batch size 512 and 10k training steps. For CEBRA-Time, we used temperature 2.25, time offset 10, batch size 512 and 4k training steps. For UMAP, we used the cosine metric and *min_dist* of 0.99 and *n_neighbors* of 31. For tSNE we used cosine metric and *perplexity* of 29. For conv-pi-VAE, we trained 1000 epochs with learning rate 2.5×10^{-4}. For autoLFADS we used the in-built ray-tune framework for finding optimal hyperparameters. CEBRA was trained with output latent 3D (the minimum) and all other methods were trained with a 2D latent.

of performance (Urai et al., 2022b). UMAP (McInnes et al., 2018) and tSNE (Van Der Maaten et al., 2009) are excellent non-linear methods, but they lack the ability to explicitly use time information, which is always available in neural recordings, and they are not directly as interpretable as PCA. Non-linear methods are desirable to use for high performance decoding, but often lack identifiability: the desirable property that true model parameters can be determined, up to a known indeterminacy (Hyvärinen et al., 2019b; Roeder et al., 2020a). This is critical as it ensures that the learned representations are uniquely determined and thus facilitates consistency across animals and/or sessions.

There is recent evidence that label-guided VAEs could improve interpretability (Klindt et al., 2021b; Sani et al., 2020; Zhou & Wei, 2020). Namely, by using behavioral variables, such algorithms can learn to project future behavior onto past neural activity (Sani et al., 2020), or explicitly use label-priors to shape the embedding (Zhou & Wei, 2020). However, these methods still have restrictive explicit assumptions on the underlying statistics of the data, and they do not guarantee consistent neural embeddings across animals (Pandarinath et al., 2018b; Prince et al., 2021; Zhou & Wei, 2020), which limits their generalizability as well as interpretability (and thereby affects accurate decoding across animals).

We address these open challenges with CEBRA, a new self-supervised learning algorithm for obtaining interpretable, **C**onsistent **E**m**B**eddings of high-dimensional **R**ecordings using **A**uxiliary variables. Our method combines ideas from non-linear independent component analysis (ICA) with contrastive learning (Gutmann & Hyvärinen, 2012; Hyvärinen et al., 2019b; Khosla et al., 2020; Oord et al., 2018), a powerful self-supervised learning scheme, to generate latent embeddings conditioned on behavior (auxiliary variables) and/or time. CEBRA uses a novel data sampling scheme to train a neural network encoder with a contrastive optimization objective to shape the embedding space. It also can generate embeddings across multiple subjects, and cope with distribution shifts between experimental sessions, subjects, and recording modalities. Importantly, our method neither relies on data augmentation (as does SimCLR (Chen et al., 2020b)), nor on a specific generative model that would limit its range of use.

Results

Joint behavioral and neural embeddings

We propose a framework for jointly trained latent embeddings. CEBRA leverages user-defined labels (supervised, hypothesis-driven), or time-only labels (self-supervised, discovery-driven; Fig. 7.1a, Suppl. Note 1) to obtain consistent embeddings of neural activity that can be used for both visualization of data and downstream tasks like decoding. Specifically, it is an instantiation of non-linear ICA based on contrastive learning (Hyvärinen et al., 2019b). Contrastive learning is a technique that leverages

contrasting samples (positive and negative) against each other to find attributes in common and those that separate them. We can use discrete and continuous variables and/or time to shape the distribution of positive and negative pairs, and then use a non-linear encoder (here, a convolutional neural network (CNN), but can be another type of model) trained with a novel contrastive learning objective. The encoder features form a low-dimensional embedding of the data (Fig. 7.1a). Generating consistent embeddings is highly desirable and closely linked to identifiability in non-linear ICA (Hälvä et al., 2021; Hyvärinen et al., 2019b). Theoretical work has shown that using contrastive learning with auxiliary variables is identifiable for bijective neural networks using a noise contrastive estimation (NCE) loss (Hyvärinen et al., 2019b), and that with an InfoNCE loss this bijectivity assumption can sometimes be removed (Zimmermann et al., 2021a) (see also Suppl. Note 2). InfoNCE minimization can be viewed as a classification problem where given a reference sample, the correct positive pair needs to be distinguished from multiple negative pairs.

CEBRA optimizes neural networks $\mathbf{f}$, $\mathbf{f}'$ that map neural activity to an embedding space of a defined dimension (Fig. 7.1a). Pairs of data $(\mathbf{x}, \mathbf{y})$ are mapped to this embedding space, and then compared with a similarity measure $\phi(\cdot, \cdot)$. Abbreviating this process with $\psi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}'(\mathbf{y}))/\tau$ with a temperature hyperparameter τ, the full criterion to optimize is

$$\mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p(\mathbf{x}),\ \mathbf{y}_+\sim p(\mathbf{y}|\mathbf{x}) \\ \mathbf{y}_1,\ldots,\mathbf{y}_n\sim q(\mathbf{y}|\mathbf{x})}} \left[-\psi(\mathbf{x}, \mathbf{y}_+) + \log \sum_{i=1}^{n} e^{\psi(\mathbf{x},\mathbf{y}_i)} \right],$$

which, depending on the dataset size, can be optimized with algorithms for either batch or stochastic gradient descent.

In contrast to other contrastive learning algorithms, the positive pair distribution p and the negative pair distribution q can be systematically designed and allows the use of time, behavior, and other auxiliary information to shape the geometry of the embedding space. If only discrete labels are used, this training scheme is conceptually similar to supervised contrastive learning (Khosla et al., 2020).

CEBRA can leverage continuous behavioral (kinematics, actions) as well as other discrete variables (trial ID, rewards, brain-area ID, etc.). Additionally, user-defined information about desired invariances in the embedding is used (across animals, sessions, etc.), allowing flexible ways of analyzing data. We group this information into task-irrelevant and task-relevant variables, and these can be leveraged in different contexts. For example, to investigate trial-to-trial variability or learning across trials, information like a trial ID would be considered a task-relevant variable. On the contrary, if we aim to build a robust brain machine interface that should be invariant to such short-term changes, we would include trial information as a task-irrelevant variable and obtain an embedding space which no longer carries this information. Crucially, this allows for inferring latent embeddings without explicitly modeling the data generating process (as done in pi-VAE (Zhou & Wei, 2020) and LFADS (Pandarinath et al., 2018b)).

Omitting the generative model and replacing it by a contrastive learning algorithm facilitates broader applicability without modifications.

Robust and decodable latent embeddings

We first demonstrate that CEBRA significantly outperforms tSNE, UMAP, autoLFADS (Keshtkaran et al., 2022) and pi-VAE (the latter was shown to outperform PCA, autoLFADS, demixed-PCA, and pfLDS (Zhou & Wei, 2020) on some tasks) at reconstructing ground truth synthetic data (one-way ANOVA, F(4, 495)=251, p=1.12 $\times$ 10^{-117}; Fig. 7.1b, Extended Data Fig. 7.3a, b).

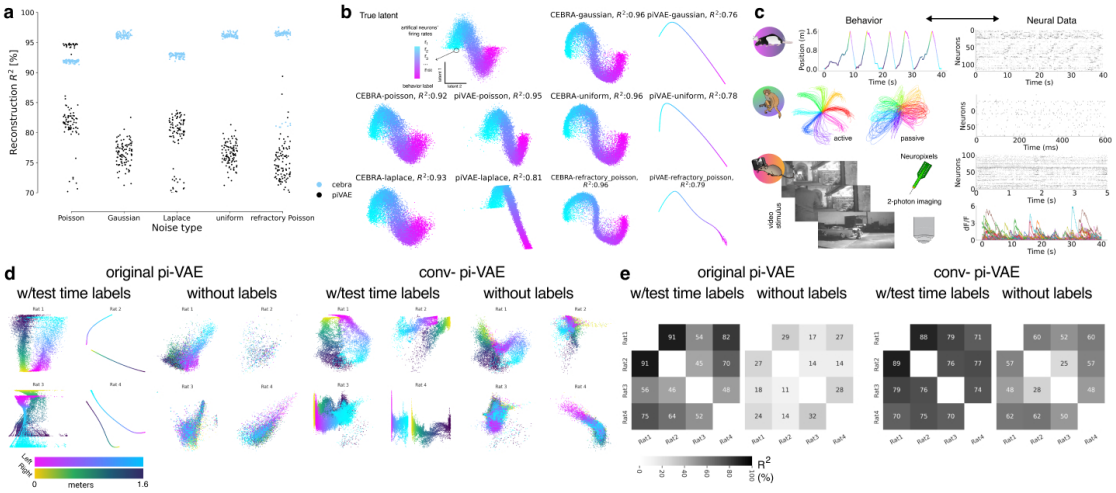


Figure 7.3: **Overview of datasets, synthetic data, & original pi-VAE implementation vs. modified conv-pi-VAE. (ab)**: We generated synthetic datasets similar to Fig. 7.1b with additional variations in the noise distributions in the generative process. We benchmarked the reconstruction score of the true latent using CEBRA and pi-VAE (100 seeds) on the generated synthetic datasets. CEBRA showed higher and less variable reconstruction scores than pi-VAE in all noise types. **(b)** Example visualization of the reconstructed latents from CEBRA and pi-VAE on different synthetic dataset types. **(c)**: we benchmarked and demonstrate the abilities of CEBRA on four datasets. Rat-based electrophysiology data (Grosmark & Buzsáki, 2016), where the animal transversed a 1.6m linear track "leftwards" or "rightwards". Two mouse-based datasets: one 2-photon calcium imaging passively viewing dataset (de Vries et al., 2020), and one with the same stimulus but recorded with Neuropixels (Siegle et al., 2021b). A monkey-based electrophysiology dataset of center out reaching from Chowdhury et al. (Chowdhury et al., 2020), and processed to trial data as in (Pei et al., 2021a). **(d)**: Conv-pi-VAE showed improved performance, both with labels (Wilcoxon signed-rank test, p=0.0341) and without labels Wilcoxon signed-rank test, p=0.0005). Example runs/embeddings the consistency across rats, with **(e)**: consistency across rats, from target to source, as computed in Fig. 7.1.

We then turned to a hippocampus dataset that was used to benchmark neural embedding algorithms (Grosmark & Buzsáki, 2016; Zhou & Wei, 2020) (Extended Data Fig. 7.3c, Suppl. Note 1). To note, we first significantly improved pi-VAE by adding a CNN thereby allowing this model to leverage multiple time steps, and

used this for further benchmarking (Extended Data Fig. 7.3d-e). To test our methods, we first consider the correlation of the resulting embedding space across subjects (does it produce similar latent spaces?), and the correlation across repeated runs of the algorithm (how consistent are the results?). We found that CEBRA significantly outperformed other algorithms at producing consistent embeddings, and it produced visually informative embeddings (Fig. 7.1c-e, Extended Data Figs. 7.4, 7.2; for each embedding a single point represents the neural population activity over a specified time bin).

When using CEBRA-Behavior, the consistency of the resulting embedding space across subjects is significantly higher compared to autoLFADS and conv-pi-VAE with, or without test-time labels (one-way ANOVA F(25.4) p=1.92×10^{-16}, Table S1; Fig. 7.1d, e). Qualitatively, it can be appreciated that both CEBRA-Behavior and -Time have similar output embeddings, while the latents from conv-pi-VAE with label priors or without labels are not consistent (CEBRA does not need test-time labels), suggesting that the label prior strongly shapes the output embedding structure of conv-pi-VAE. We also considered correlations across repeated runs of the algorithm and found higher consistency and lower variability with CEBRA (Extended Data Fig. 7.5).

Hypothesis- and discovery-driven analyses

One of the advantages of CEBRA is its collective flexibility, limited assumptions, and ability to test hypotheses. For the hippocampus, one can hypothesize that these neurons represent space (Huxter et al., 2003; Moser et al., 2008), and therefore the behavioral label could be position, or velocity (Fig.7.6a). Also considering structure in the behavioral data could help refine which behavioral labels to use jointly with neural data (Fig. 7.6b). Conversely, for the sake of argument, we could have an alternative hypothesis; i.e., hippocampus does not map space, just the direction of travel, or some other feature. Using the same model, but hypothesis-free and using time for selecting the contrastive pairs is also possible, and/or a hybrid thereof (Fig. 7.6a, b).

We trained hypothesis-guided (supervised), time-only (self-supervised), or hybrid models across a range of input dimensions and embedded the neural latents into a 3D space for visualization. Qualitatively, we find that position-based model produces a highly smooth embedding that reveals the position of the animal—namely, there is a continuous "loop" of latent dynamics around the track (Fig. 7.6b). This is consistent with what is known about the hippocampus (Grosmark & Buzsáki, 2016) and in particular reveals the topology of the linear track with direction specificity. Whereas shuffling the labels, which breaks the correlation between neural activity and direction and position, produces an unstructured embedding (Fig. 7.6b).

CEBRA-Time produces an embedding that more closely resembles that of position (Fig. 7.6b). This also suggests that time contrastive learning captured the major latent space structure, independent of any label input, reinforcing that CEBRA can serve both discovery and hypothesis-driven questions (and running both variants can be

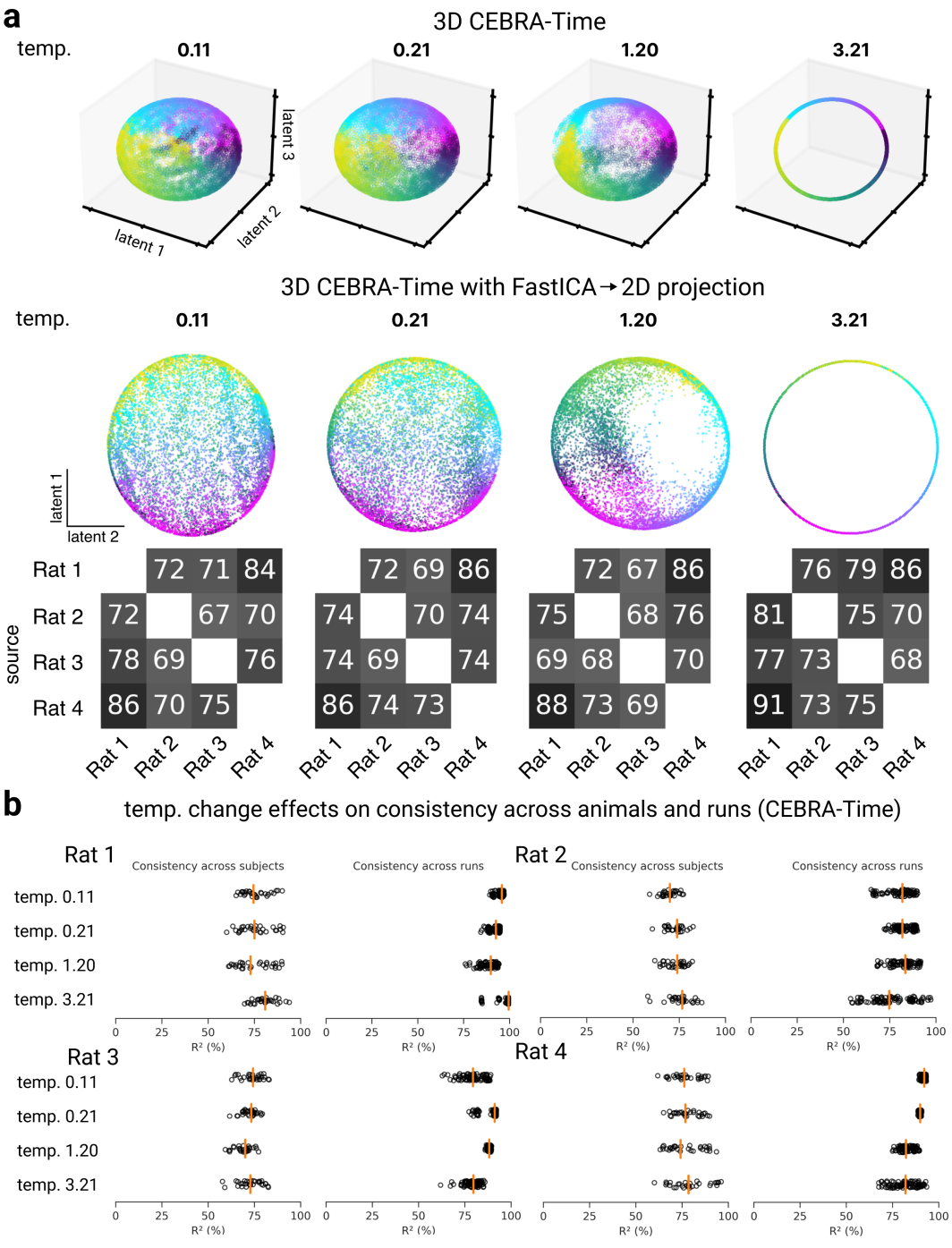


Figure 7.4: **Hyperparameter changes on visualization and consistency. (a)**: Temperature has the largest effect on visualization (vs. consistency) of the embedding as shown by a range from 0.1 to 3.21 (highest consistency for Rat 1), as can be appreciated in 3D (top) and post FastICA into a 2D embedding (middle). Bottom row shows the corresponding change on mean consistency, and in **b,** the variance can be noted. Orange line denotes the median and black dots are individual runs (subject consistency: 10 runs with 3 comparisons per rat; run consistency: 10 runs, each compared to 9 remaining runs).

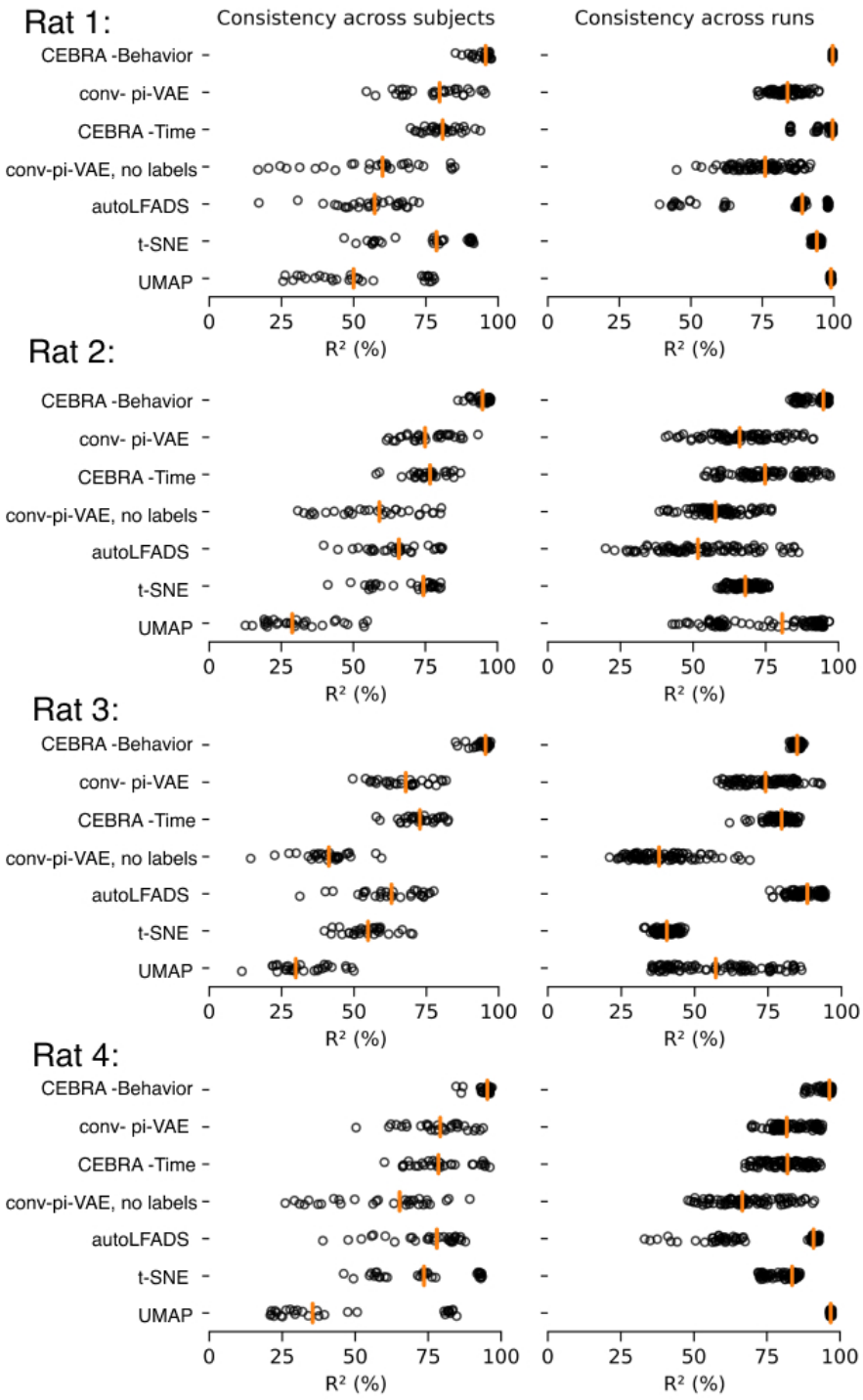


Figure 7.5: **Additional metrics used for benchmarking consistency.** Comparisons of all algorithms along different metrics for Rats 1, 2, 3, 4. The orange line is median across n=10 runs, black circles denote individual runs. Each run is the average over three non-overlapping test splits.

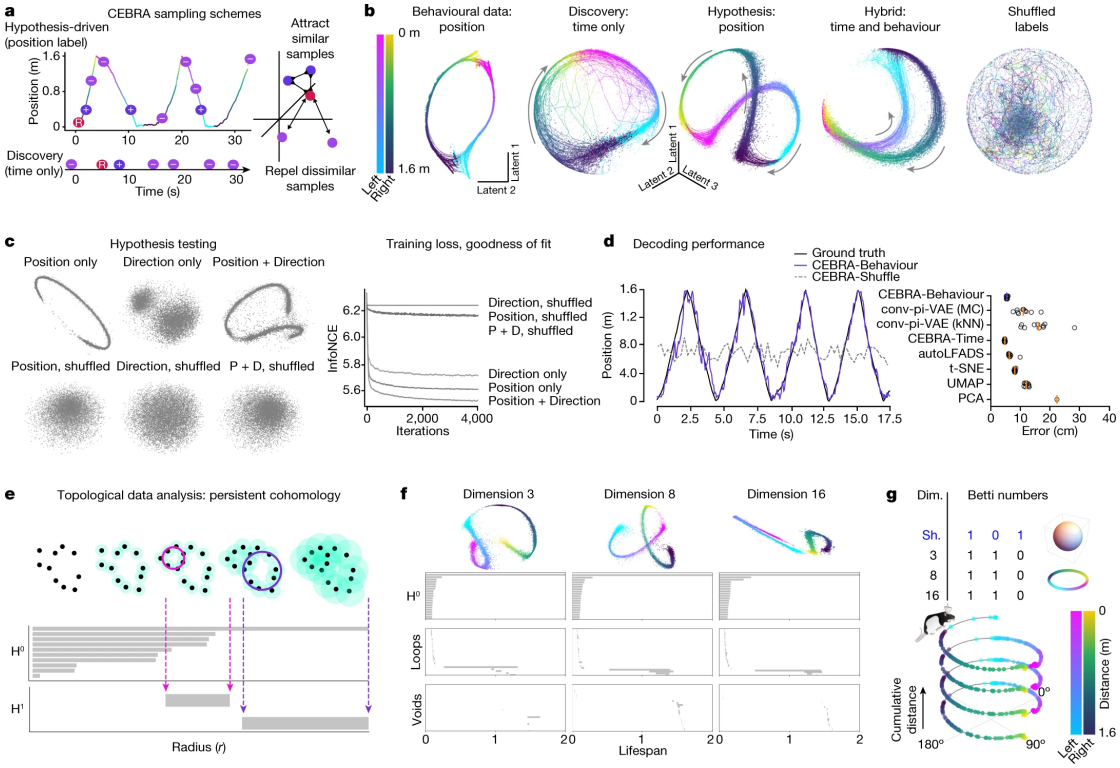


Figure 7.6: **Hypothesis- and discovery-driven analysis with CEBRA. (a)**: CEBRA can be used in any of three modes: hypothesis-driven mode, discovery-driven mode, or hybrid mode, which allows for weaker priors on the latent embedding. **(b)**: Left to right, CEBRA on behavioral data using position as a label, CEBRA-Time, CEBRA-Behavior (on neural data) with position hypothesis, CEBRA-Hybrid (a five-dimensional space was used, in which 3D is first guided by both behavior+time and the final 2D is guided by time) and shuffled (erroneous). **(c)**: Embeddings with position (P) only, direction (D) only and P+D only, and shuffled position only, direction only and P+D only, for hypothesis testing. The loss function can be used as a metric for embedding quality. **(d)**: Left, we utilized either hypothesis-driven P+D or shuffle (erroneous) to decode the position of the rat, which yielded a large difference in decoding performance: P+D R^2=73.35% versus -49.90% for shuffled, and median absolute error 5.8 versus 44.7 cm. Purple line represents decoding from the 3D hypothesis-based latent space; dashed line is shuffled. Right, performance across additional methods (orange bars indicate the median of individual runs (n=10), indicated by black circles. Each run is averaged over three splits of the dataset). MC, Monte Carlo. **(e)**: Schematic showing how persistent cohomology is computed. Each data point is thickened to a ball of gradually expanding radius (r) while tracking the birth and death of "cycles" in each dimension. Prominent lifespans, indicated by pink and purple arrows, are considered to determine Betti numbers. **(f)**: Top, visualization of neural embeddings computed with different input dimensions. Bottom, related persistent cohomology lifespan diagrams. **(g)**: Betti numbers from shuffled embeddings (sh.) and across increasing dimensions (dim.) of CEBRA, and the topology-preserving circular coordinates using the first cocycle from persistent cohomology analysis (Methods).

informative). The hybrid design, whose goal is to disentangle the latent to subspaces that are relevant to the given behavioral and the residual temporal variance and noise, showed a similarly structured embedding space as behavior (Fig. 7.6b).

To quantify how CEBRA can disentangle which variable had the largest influence on the embedding, we tested for encoding position, direction, and combinations thereof (Fig. 7.6c). We find that position plus direction is the most informative label (Dombeck et al., 2010) (Fig. 7.6c, and Extended Data Fig. 7.7a-d). This is evident in the embedding and the value of the loss function upon convergence, which serves as an additional "goodness of fit" metric to select the best labels; i.e., which label(s) produce the lowest loss at the same point in training (Extended Data Fig. 7.7e). Note that erroneous (shuffled) labels converge to considerably higher loss values.

To measure performance we consider how well can we decode behavior from the embeddings. As an additional baseline we performed linear dimensionality reduction with PCA. We used a k-nearest-neighbor (kNN) decoder for position and direction and measured the reconstruction error. We find CEBRA-Behavior has significantly better decoding performance (Fig. 7.6d, and Suppl. Video 1), compared to pi-VAE and our conv-pi-VAE (one-way ANOVA F=131, p=3.6×10^{-24}), and CEBRA-Time compared to unsupervised methods (autoLFADS, tSNE, UMAP, PCA; one-way ANOVA F=1983, p=6×10^{-50}; see also Table S2). Zhou & Wei (Zhou & Wei, 2020) reported a median absolute decoding error of 12 cm error, while we can achieve ≈5 cm (Fig. 7.6d). CEBRA therefore allows for high performance decoding while ensuring consistent embeddings.

Co-homology as a metric for robust embeddings

CEBRA can be trained across a range of dimensions and models can be selected based on decoding, goodness of fit, and consistency. Yet, we also sought to find a principled approach to verify the robustness of embeddings, which might yield insight into neural computations (Chaudhuri et al., 2019; Curto, 2016) (Fig. 7.6e). We used algebraic topology to measure the persistent co-homology, for comparing if learned latent spaces are equivalent. While it is not required to project embeddings onto a sphere, this has the advantage that there are default Betti numbers (for a d-dimensional uniform embedding, $H^0 = 1, H^1 = 0, \cdots, H^{d-1} = 1$, i.e., 1,0,1 for the 2-sphere). We used the distance from the unity line (and thresholded based on a computed null shuffled distribution in Births vs. Deaths to compute Betti numbers; Extended Data Fig. 7.8). Using CEBRA-Behavior or -Time we find a ring topology (1,1,0; Fig. 7.6f), as one would expect from a linear track for place cells. We then computed the Eilenberg-MacLane coordinates for the identified co-cycle (H1) for each model (de Silva et al., 2009; Gardner et al., 2022)—this allowed us to map each time-point to topology-preserving coordinates—and indeed we find that the ring topology for the CEBRA models matches space (position) across dimensions (Fig. 7.6g, Extended Data Fig. 7.8). Note, this topology differs from (1,0,1); i.e., Betti numbers for a uniformly covered sphere, which in our setting would indicate a random embedding as found by shuffling (Fig. 7.6g).

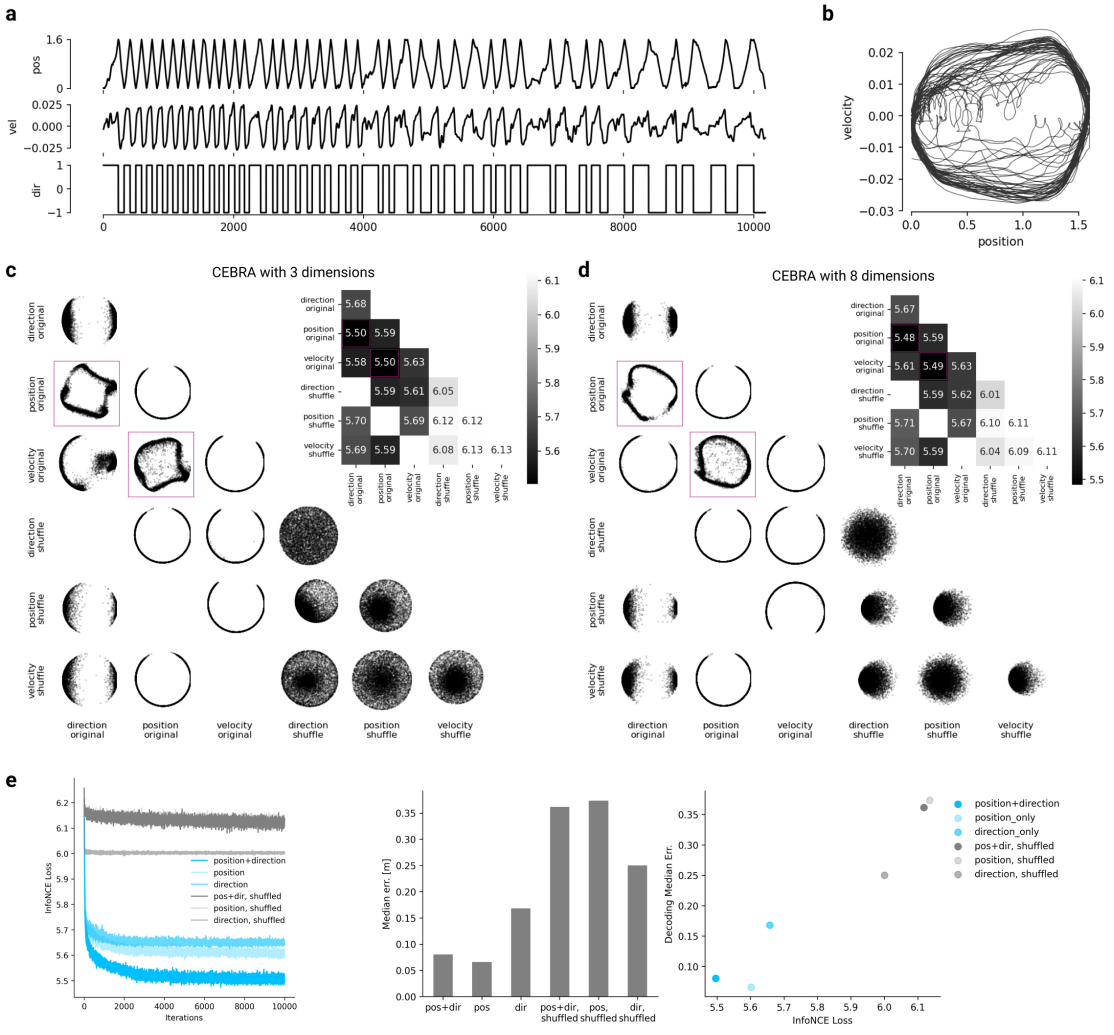


Figure 7.7: **Hypothesis testing with CEBRA (a)**: Example data from a hippocampus recording session (Rat 1). We tested possible relationships between three experimental variables (rat location, velocity, movement direction) and the neural recordings (120 neurons, not shown). **(b)**: Relationship between velocity and position. **(c)**: We trained CEBRA with three-dimensional outputs on every single experimental variable (main diagonal) and every combination of two variables. All variables are treated as "continuous" in this experiment. We compared original to shuffled variables (shuffling is done by permuting all samples over the time dimension) as a control. We projected the original three dimensional space onto the first principal components. We show the minimum value of the InfoNCE loss on the trained embedding for all combinations in the confusion matrix (lower number is better). Either velocity or direction, paired with position information is needed for maximum structure in the embedding (highlighted, colored), yielding lowest InfoNCE error. **(d)**: Using an eight-dimensional CEBRA embedding did not qualitatively alter the results. We again report the first two principal components as well as InfoNCE training error upon convergence, and find non-trivial embeddings with lowest training error for combinations of direction/velocity and position. **(e)**: The InfoNCE metric can serve as the goodness of fit metric, both for hypothesis testing and identifying decodable embeddings. We trained CEBRA in discovery-driven mode with 32 latent dimensions. We compared the InfoNCE loss (left, middle) between various hypotheses. Low InfoNCE was correlated with low decoding error (right).

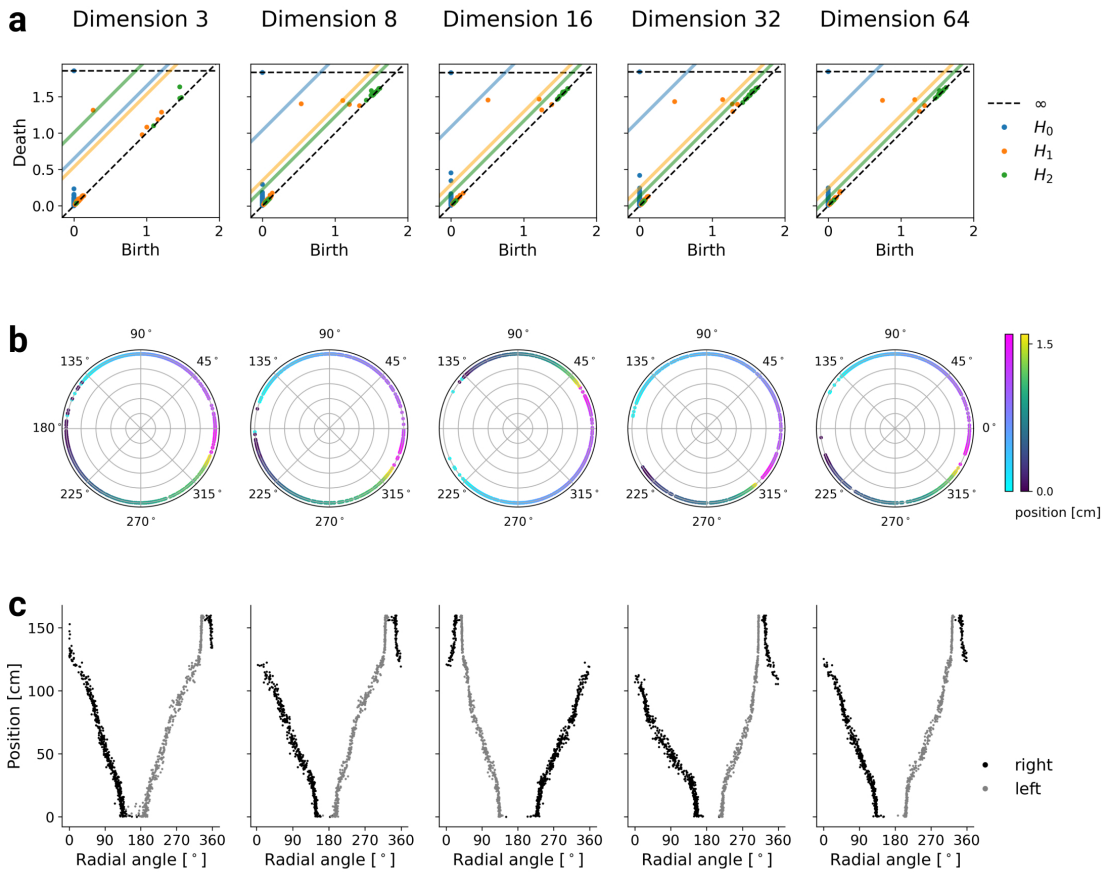


Figure 7.8: **Persistence across dimensions (a)**: For each dimension of CEBRA-Behavior embedding from the rat hippocampus dataset Betti numbers were computed by applying persistent co-homology. The colored dots are lifespans observed in hypothesis based CEBRA-Behavior. To rule out noisy lifespans, we set a threshold (colored diagonal lines) as maximum lifespan based on 500 seeds of shuffled-CEBRA embedding for each dimension. **(b)**: The topology preserving circular coordinates using the first co-cycle from persistent co-homology analysis on the CEBRA embedding of each dimension is shown (see Methods). The colors indicate position and direction of the rat at the corresponding CEBRA embedding points. **(c)**: The radial angle of each embedding point obtained from **(b)** and the corresponding position and direction of the rat.

Multi-session, multi-animal CEBRA

CEBRA can also be used to jointly train across sessions and different animals, which can be highly advantageous when there is limited access to simultaneously recorded neurons, or when looking for animal-invariant features in the neural data. We trained CEBRA across animals within each multi-animal dataset and find this joint embedding allows for even more consistent embeddings across subjects (Extended Data Fig. 7.9a-c; one-sided, paired T-tests, Allen data: (-5.80), p=5.99 $\times$ 10^{-5}; Hippocampus: (-2.22), p=0.024).

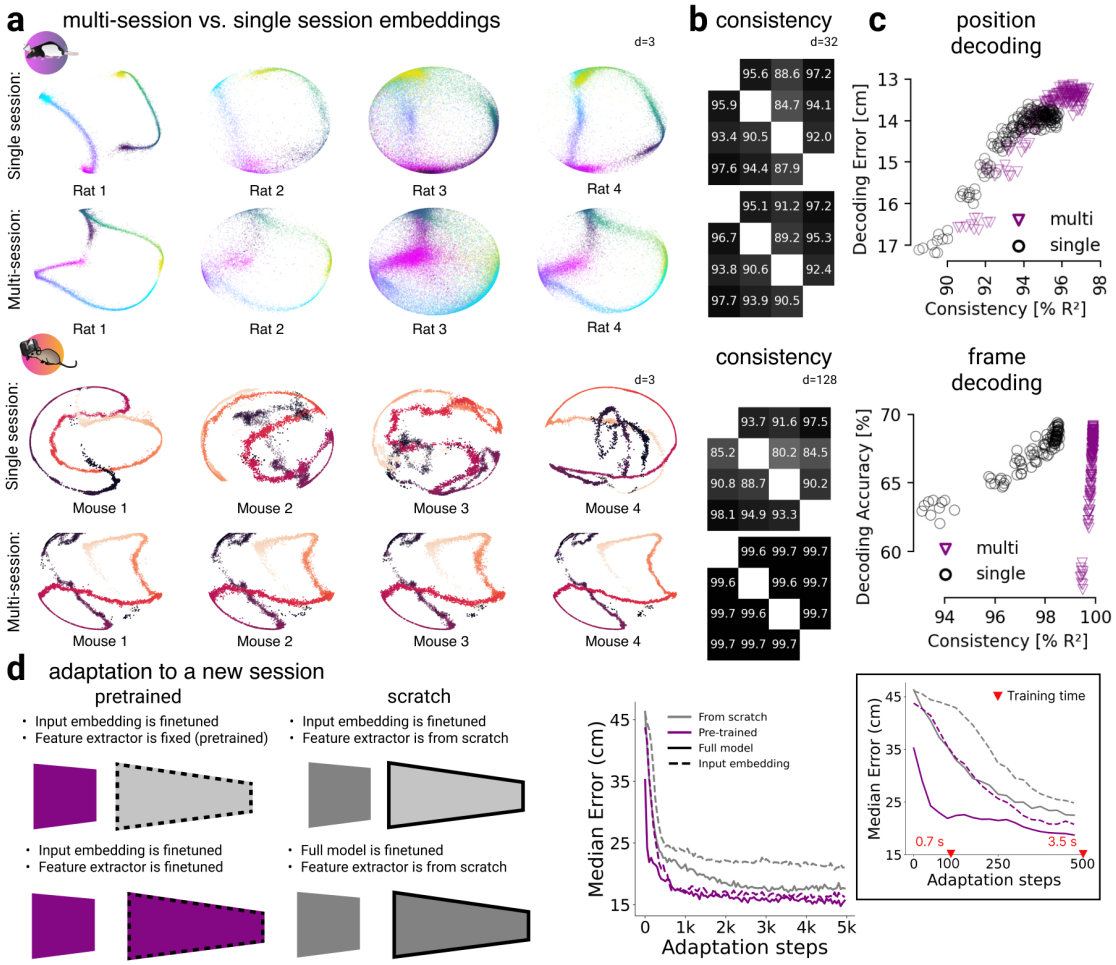


Figure 7.9: **Multi-session training and rapid decoding (a)**: Top: hippocampus dataset, single animal vs. multi-animal training shows an increase in consistency across animals. Bottom: same for Allen dataset, 4 mice. **(b)**: consistency matrix single vs. multi-session training for hippocampus (32D embedding) and Allen data (128D embedding) respectively. Consistency is reported at the point in training where the average position decoding error is less than 14 cm (corresponds to 7 cm error for rat 1), and a decoding accuracy of 60% on the Allen dataset. **(c)**: Comparison of decoding metrics for single or multi-session training at various consistency levels (averaged across all 12 comparisons). Models were trained for 5,000 (single) or 10,000 (multi-session) steps with a 0.003 learning rate; batch size was 7,200 samples per session. Multi-session training requires longer training or higher learning rates to obtain the same accuracy due to the 4-fold larger batch size, but converges to same decoding accuracy. We plot points at intervals of 500 steps (n=10 seeds); training progresses from lower right to upper left corner within both plots. **(d)**: We demonstrate that we could also adapt to an unseen dataset; here, 3 rats were used for pretraining, and rat #4 was used as a held-out test. The grey lines indicate models trained from scratch (random initialization). We also tested fine-tuning only the input embedding (first layer) or the full model, as the diagram, left, describes. We measured the average time (mean $\pm$ STD) to adapt 100 steps (0.65 $\pm$ 0.13 sec) and 500 steps (3.07 $\pm$ 0.61 sec) on 40 repeated experiments.

While consistency increased, it is not *a priori* clear that decoding from "pseudo-subjects" would be equally good, as there could be session or animal specific information that is lost in pseudo-decoding (as decoding is usually performed within session). Alternatively, if this joint latent space was as high-performance as the single subject, this would suggest that CEBRA is able to produce robust latent spaces across subjects. Indeed, we find no loss in decoding performance (Extended Data Fig. 7.9c).

It is also possible to rapidly decode from a new session that is *unseen* during training, which is an attractive setting for brain machine interface deployment. We show that by pretraining on a subset of the subjects, we can apply and rapidly adapt CEBRA-Behavior on unseen data (i.e., it runs at 50–100 steps/second, and positional decoding error already decreased by 10 cm after adapting the pretrained network for one step). Lastly, we can achieve a lower error more rapidly compared to training fully on the unseen individual (Extended Data Fig. 7.9d). Collectively, this shows that CEBRA can rapidly produce high-performance, consistent and robust latent spaces.

Discovering latent dynamics during a motor task

We next consider an eight direction "center-out" reaching task paired with electrophysiology recordings in somatosensory cortex (S1) of a primate (Chowdhury et al., 2020) (Fig. 7.10a). The monkey performed active movements and in a subset of trials experienced randomized bumps that caused a passive limb movement. CEBRA produced highly informative visualisations of the data compared to other methods (Fig. 7.10b), and CEBRA-Behavior can be used in order to test encoding properties of S1. Using position or time information showed embeddings with clear positional encoding (Fig. 7.10c, d, and Extended Data Fig. 7.11a-c).

To then test how directional information and active vs. passive movements influence population dynamics in S1 (Chowdhury et al., 2020; London & Miller, 2013; Prud'homme & Kalaska, 1994), we trained embedding spaces with directional information and then either separated the trials into active and passive for training (Fig. 7.10e), or trained jointly and post-hoc plotted separately (Fig. 7.10f). We find striking similarities that suggest active vs. passive strongly influences the neural latent space: the embeddings for active trials show a clear start and stop, while for passive trials it shows a continuous trajectory through the embedding, independently of how they are trained. This finding is confirmed in embeddings that used only the *continuous* position of the end-effector as the behavioral label (Fig. 7.10g). Notably, direction is a less prominent feature (Fig. 7.10g), although they are entangled parameters in this task.

Next, since position and active or passive trial type appear robust in the embeddings, we further explored the decodability of the embeddings. Both position and trial type were readily decodable from 8D+ embeddings with a kNN decoder trained on position-only, but directional information was not as decodable (Fig. 7.10h). Here too the loss function is informative for hypothesis testing (Extended Data Fig. 7.11d-f). Notably, we could recover the hand trajectory with an R^2 of 88% (concatenated across 26 held-

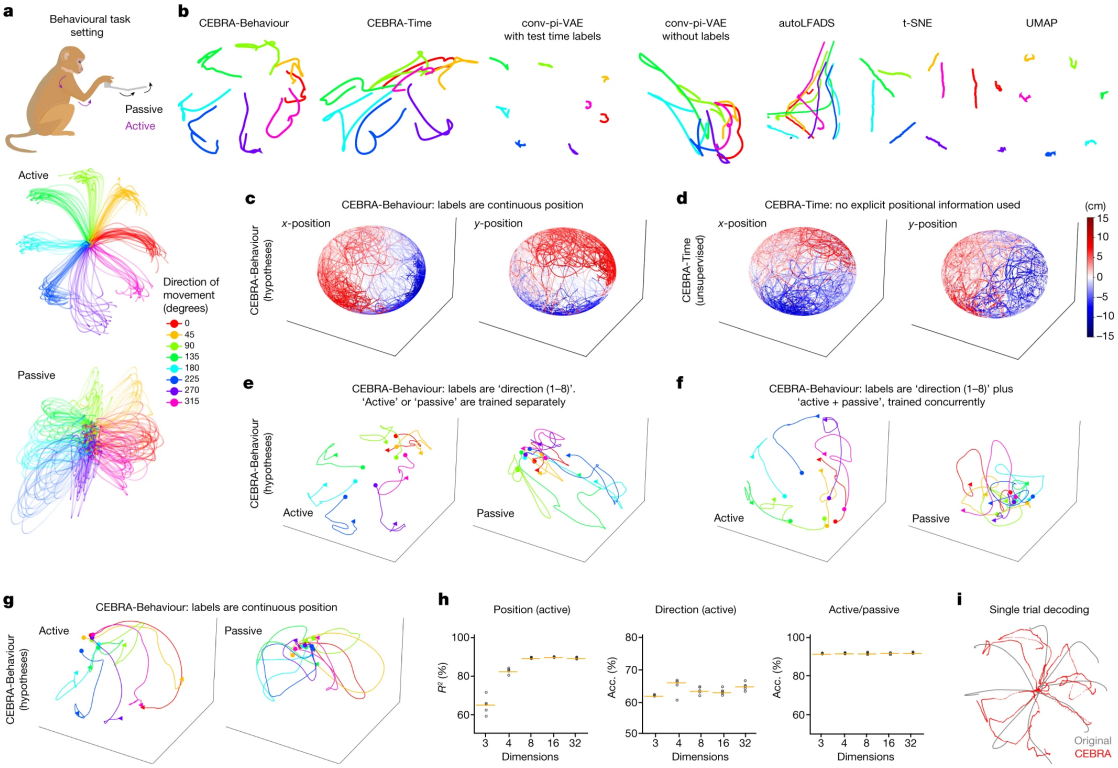
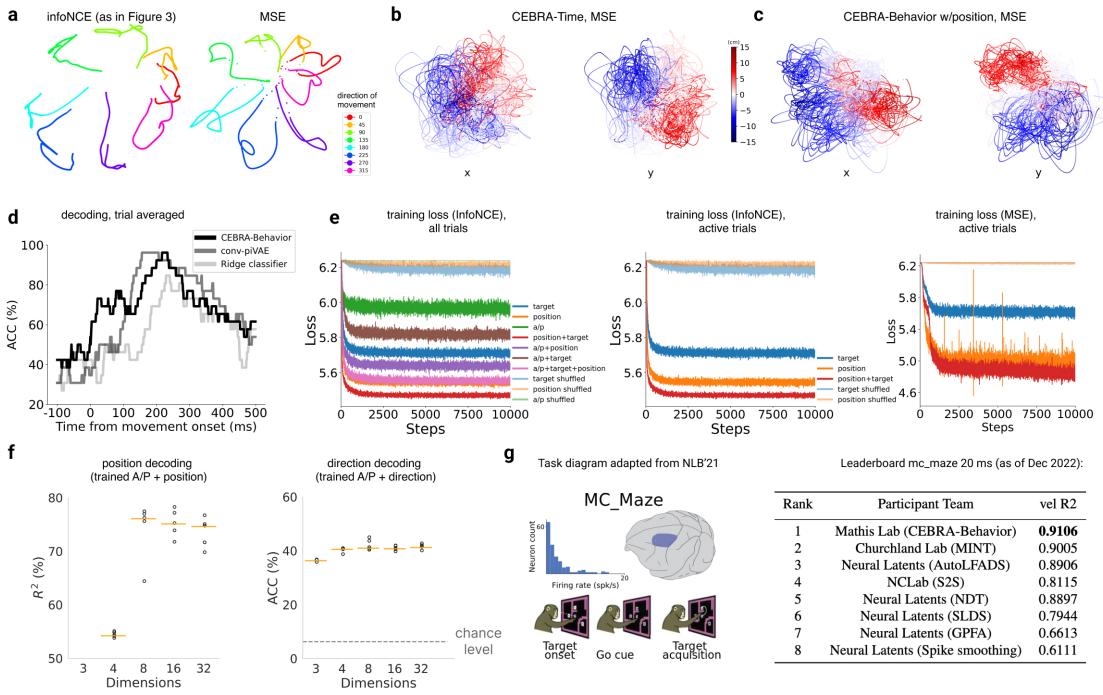


Figure 7.10: **Forelimb movement behavior in a primate. (a)**: The monkey makes either active or passive movements in eight directions. Data derived from area 2 of primary S1 from Chowdhury et al. (2020). Cartoon from https://scidraw.io. **(b)**: Embeddings of active trials generated with 4D CEBRA-Behavior, 4D CEBRA-Time, 4D autoLFADS, 4D conv-pi-VAE variants, 2D tSNE and 2D UMAP. The embeddings of trials (n=364) for each direction are post hoc averaged. **(c)**: CEBRA-Behavior trained with (x,y) position of the hand. Left, color coded to x position; right, color coded to y position. **(d)**: CEBRA-Time with no external behavior variables. Color coded as in c. **(e)**: CEBRA-Behavior embedding trained using a 4D latent space, with discrete target direction as behavior labels, trained and plotted separately for active and passive trials. **(f)**: CEBRA-Behavior embedding trained using a 4D latent space, with discrete target direction and active and passive trials as auxiliary labels plotted separately, active versus passive trials. **(g)**: CEBRA-Behavior embedding trained with a 4D latent space using active and passive trials with continuous (x,y) position as auxiliary labels plotted separately, active versus passive trials. The trajectory of each direction is averaged across trials (n=18–30 each, per direction) over time. Each trajectory represents 600 ms from -100 ms before the start of the movement. **(h)**: Left to right, decoding performance of: position using CEBRA-Behavior trained with (x,y) position (active trials); target direction using CEBRA-Behavior trained with target direction (active trials); and active versus passive accuracy (Acc.) using CEBRA-Behavior trained with both active and passive movements. In each case we trained and evaluated five seeds, represented by black dots; orange line represents the median. **(i)**, Decoded trajectory of hand position using CEBRA-Behavior trained on active trial with (x,y) position of the hand. The grey line represents a true trajectory and the red line represents a decoded trajectory.

out test trials, Fig. 7.10i) using a 16D CEBRA-Behavior model trained on position (Fig. 7.10i). For comparison, a L1 regression using all neurons achieved R^2 74%, and 16D conv-pi-VAE achieved R^2 82%. We also tested CEBRA on an additional monkey dataset (mc_maze) presented in the Neural Latent Benchmark (Pei et al., 2021c), where it achieves state-of-the-art decoding performance (Extended Data Fig. 7.11).



Rank	Participant Team	vel R2
1	Mathis Lab (CEBRA-Behavior)	**0.9106**
2	Churchland Lab (MINT)	0.9005
3	Neural Latents (AutoLFADS)	0.8906
4	NCLab (S2S)	0.8115
5	Neural Latents (NDT)	0.8897
6	Neural Latents (SLDS)	0.7944
7	Neural Latents (GPFA)	0.6613
8	Neural Latents (Spike smoothing)	0.6111

Figure 7.11: **Somatosensory cortex decoding from primate recordings (a)**: We compare CEBRA-Behavior with the cosine similarity and embeddings on the sphere reproduced from Fig. 7.10b (left) against CEBRA-Behavior trained with the MSE loss and unnormalized embeddings. The embeddings of trials (n=364) of each direction were post-hoc averaged. **(b)**: CEBRA-Behavior trained with x,y position of the hand. Left panel is color-coded to changes in x position and right panel is color-coded to changes in y position. **(c)**: CEBRA-Time without any external behavior variables. As in **b**, left and right are color-coded to x and y position, respectively. **(d)**: Decoding performance of on target direction using CEBRA-Behavior, conv-pi-VAE and a linear classifier. CEBRA-Behavior shows significantly higher decoding performance than the linear classifier (one-way ANOVA, $F(2,75)=3.37$, $p<0.05$ with Post Hoc Tukey HSD $p<0.05$). **(e)**: Loss (InfoNCE) vs. training iteration for CEBRA-Behavior with position, direction, active or passive, and position+direction labels (and shuffled labels) for all trials (left) or only active trials (right), or active trials with a MSE loss. **(f)**: Additional decoding performance results on position and direction-trained CEBRA models with all trial types. For each case, we trained and evaluated 5 seeds represented by black dots and the orange line represents the median. **(g)**: Results on the mc-maze benchmark.

Consistent embeddings across recording modalities

CEBRA is agnostic to the recording modality of neural data. But do different modalities produce similar latent embeddings? Understanding the relationship of calcium signaling and electrophysiology is a debated topic, yet an underlying assumption is that they inherently encode related, yet not identical, information. Although there are a wealth of excellent tools aimed at inferring spike trains from calcium data, currently the pseudo-R^2 of algorithms on paired spiking and calcium data tops out at around 0.6 (Berens et al., 2018). Nonetheless, it is clear that recording with either modality has lead to similar global conclusions—for example, grid cells can be uncovered in spiking or calcium signals (Gardner et al., 2022; Hafting et al., 2005b), reward prediction errors can be found in dopamine neurons across species and recording modalities (Cohen et al., 2012; Menegas et al., 2015; Schultz et al., 1997), and visual cortex shows orientation tuning across species and modalities (Hubel & Wiesel, 1977; Niell et al., 2008; Ringach et al., 2016).

We aimed to formally study whether CEBRA could capture the same neural population dynamics either from spikes or calcium imaging. We utilized a dataset from the Allen Brain Observatory where mice passively watched three movies repeatedly. We focused on paired data from 10 repeats of "Natural Movie 1" where neural data were recorded with either Neuropixels probes or via calcium imaging with a 2-photon (2P) microscope (from separate mice) (de Vries et al., 2020; Siegle et al., 2021b). Note, the data we considered thus far have goal-driven actions of the animals (such as running down a linear track or reaching to targets), yet this visual cortex dataset is collected during passive viewing (Fig. 7.12a).

We used the movie features as "behavior" labels by extracting the high-level visual features from the movie on a frame-by-frame basis using DINO, a powerful vision transformer (Caron et al., 2021a). Those were then used to sample the neural data with feature-labels (Fig. 7.12b). Next, we used Neuropixels data or calcium (2P) data (each with multi-session training) in order to generate (from 8D to 128D) latent spaces from varying number of neurons recorded from V1 (Fig. 7.12c, d). The visualization of CEBRA-Behavior showed trajectories that smoothly capture the video with either modality with an increasing number of neurons. This is reflected quantitatively in the consistency metric (Fig. 7.12e). Strikingly, CEBRA-Time nicely captured the 10 repeats of the movie (Extended Data Fig. 7.13). This result demonstrates that there is a highly consistent latent space independent of the recording method.

Next, we stacked the neurons from different mice and modalities and then sampled random subsets of V1 neurons to construct a pseudo-mouse. We did not find that joint training lowered consistency within modality (Extended Data Fig. 7.14a, b), and overall we found considerable improvement in consistency with joint training (Fig. 7.12f-h).

Using CEBRA-Behavior or -Time we trained models on four higher visual areas (HVAs) and one posterior parietal cortex (PPC) area and measured the consistency with and without joint training, and within or across areas. Our results show that with

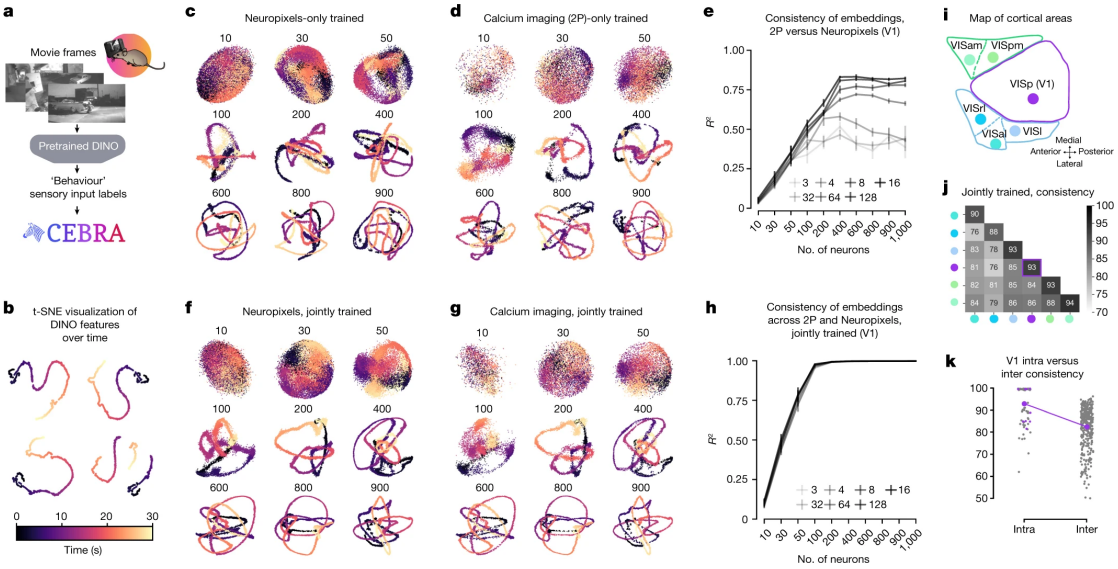


Figure 7.12: **Spikes and calcium signalling show similar CEBRA embeddings. (a)**: CEBRA-Behavior can use frame-by-frame video features as a label of sensory input to extract the neural latent space of the visual cortex of mice watching a video. Cartoon from https://scidraw.io. **(b)**: tSNE visualization of the DINO features of video frames from four different DINO configurations (latent size, model size), all showing continuous evolution of video frames over time. **(c,d)**: Visualization of trained 8D latent CEBRA-Behavior embeddings with Neuropixels (NP) data (c) or calcium imaging (2P) (d). Numbers above each embedding indicate neurons subsampled from the multi-session concatenated dataset. Color map as in b. **(e)**: Linear consistency between embeddings trained with either calcium imaging or Neuropixels data (n=10–1,000 neurons, across n=5 shuffles of neural data; mean values ± s.e.m.). **(f,g)**: Visualization of CEBRA-Behavior embedding (8D) trained jointly with Neuropixels (f) and calcium imaging (g). Color map as in (b). **(h)**: Linear consistency between embeddings of calcium imaging and Neuropixels trained jointly using a multi-session CEBRA model (n=10–1000 neurons, across n=5 shuffles of neural data; mean values ± s.e.m.). **(i)**: Diagram of mouse primary visual cortex (V1, VISp) and higher visual areas. **(j)**: CEBRA-Behavior 32D model jointly trained with 2P+NP incorporating 400 neurons, followed by measurement of consistency within or across areas (2P versus NP) across two unique sets of disjoint neurons for three seeds and averaged. **(k)**: Models trained as in h, with intra-V1 consistency measurement versus all interarea versus V1 comparison. Purple dots indicate mean of V1 intra-V1 consistency (across n=120 runs) and inter-V1 consistency (n=120 runs). Intra-V1 consistency is significantly higher than interarea consistency (one-sided Welch's t-test, t(12.30)=4.55, p=0.00019).

joint training intra-area consistency is higher vs. other areas (Fig. 7.12i-k), suggesting that with CEBRA we are not removing biological differences across areas (that have known differences in decodability and feature representations (Esfahany et al., 2018; Jin & Glickfeld, 2020)). Moreover, we test within modality and find a similar effect with CEBRA-Behavior or -Time within recording modality (Extended Data Fig. 7.14c-f).

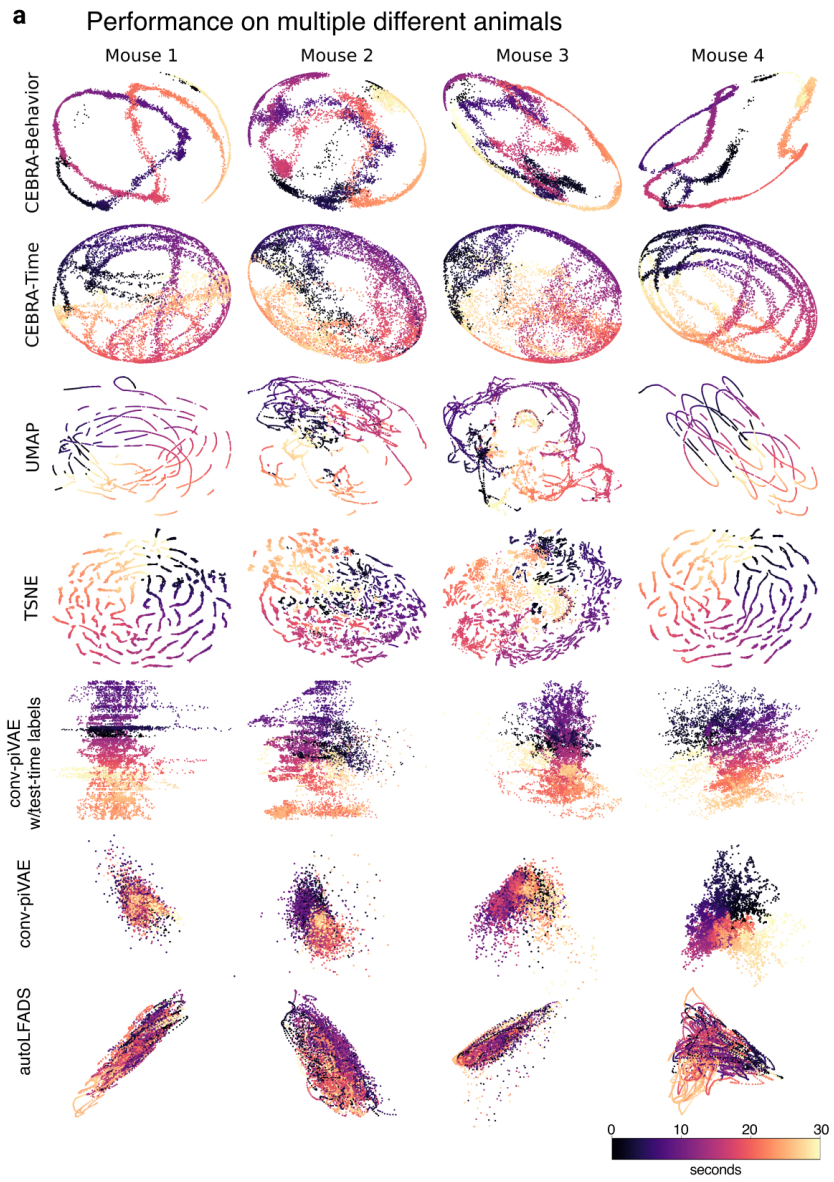


Figure 7.13: **CEBRA produces consistent, highly decodable embeddings (a)**: Additional 4 sessions with the most neurons in the Allen visual dataset calcium recording shown for all algorithms we benchmarked (see Methods). For CEBRA-Behavior and CEBRA-Time, we used temperature 1, time offset 10, batch size 128 and 10k training steps. For UMAP, we used a cosine metric and *n_neighbors* 15 and *min_dist* 0.1. For tSNE, we used a cosine metric and *perplexity* 30. For conv-pi-VAE, we trained with 600 epochs, a batch size of 200 and a learning rate 5×10^{-4}. LFADS was trained with ray-tune parameter selection and the resulting factors were transformed with PCA to generate the visualization. All methods used 10 time bins input. CEBRA was trained with 3D latent and all other methods were obtained with an equivalent 2D latent dimension. To align for visualization, we aligned to mouse 1, except for conv-pi-VAE without labels and for autoLFADS, which visually looked best when aligned to mouse 4.

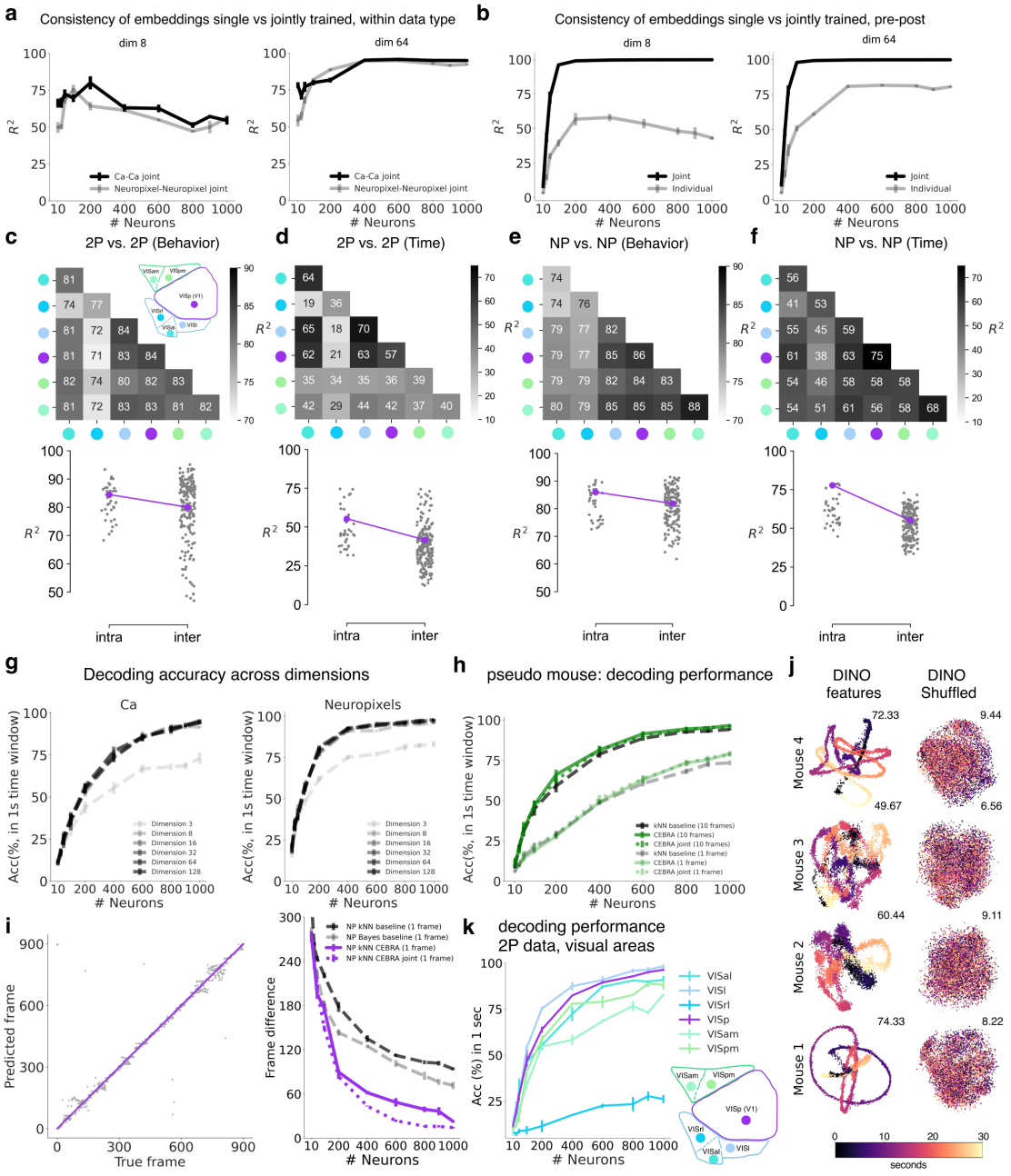


Figure 7.14: **Spikes and calcium signaling reveal similar embeddings (a)**: Consistency between the single modality embedding and jointly trained embedding from CEBRA. In higher dimensions, the embedding from single recording modality and the jointly trained embedding became highly consistent with more neurons. **(b)**: Consistency of embeddings from two recording modalities, when a single modality was trained independently and/or jointly trained. The consistency significantly improved with joint training. In higher dimensions, the consistency between single modality embeddings improved as well, which shows that CEBRA can find 'common latents' in two different recording methods (that is theoretically meant to have same information) even without joint training (yet, joint training improves consistency). This data is also presented in Fig. 7.12e, h, but here plotted together to show improvement with joint training.

Figure 7.14 (continued): **(c-f)**: Consistency across modalities and areas for CEBRA-Behavior and -Time (as computed in Fig. 7.12i-k). The purple dots indicate mean of intra-V1 scores and inter-V1 scores (inter-V1 vs intra-V1 Welch's t-test; 2P (Behavior): T(10.6)=1.52, p=0.081, 2P (Time): T(44.3)=4.26 ,p=0.0005, NP (Behavior): T(11.6)=2.83, p=0.0085, NP (Time): T(8.9)=15.51, p<0.00001) **(g)**: CEBRA + kNN decoding performance (see Methods) of CEBRA embeddings of different output embedding dimensions, from calcium (2P) data or Neuropixels, as denoted.**(h)**: Decoding accuracy measured by considering predicted frame being within 1 sec difference to true frame as correct prediction using CEBRA (2P only), jointly trained (2P+NP), or a baseline population vector kNN decoder (using the time window 33 ms (single frame), or 330 ms (10 frame receptive field)). **(i)**: Single frame performance and quantification using CEBRA 1 frame receptive field (NP data), or baseline models. **(j)**: As a control experiment we shuffled DINO features: CEBRA-Behavior used the DINO features as behavior labels and CEBRA-Shuffled used the shuffled DINO features. We shuffled the frame order of DINO features within a repeat. Same shuffled order was use for all repeats. Color code is frame number from the movie. The prediction is considered as true if the predicted frame is within 1 sec from the true frame, and the accuracy (%) is noted next to the embedding. For Mice ID 1-4: 337, 353, 397, 475 neurons were recorded, respectively. **(k)**: Decoding performance from 2P data from different visual cortical areas from different layers (2/3, 4, 5/6), as denoted, using a 10 frame window CEBRA-Behavior model using DINO features with 128 output dimension.

Decoding of natural movies from visual cortex

We performed V1 decoding analysis using CEBRA models that are either joint-modality trained, single-modality trained, or with a baseline population vector then paired with a simple kNN or naive Bayes decoder. We aimed to see if we could decode on a frame-by-frame basis the natural movie the mice watched. We used the last movie repeat as a held-out test set and nine repeats as the training set. We could achieve greater than 95% decoding accuracy, which is significantly better than the baseline decoding methods (naive Bayes or kNN) for Neuropixels recordings, and joint training CEBRA outperformed Neuropixels-only CEBRA based training (single frame: one-way ANOVA, F(3,197)=5.88, p=0.0007, Tables S3, S4, S5, Fig. 7.15a-d, Extended Data Fig. 7.14g, h). Accuracy was defined as the fraction of correct frames within a 1-second window or by the correct scene being identified. Frame-by-frame results also showed reduced frame ID errors (one-way ANOVA, F(3,16)=20.22, p=1.09×10^{-5}, n=1000 neurons, Table S6) which can be appreciated in Fig. 7.15e, f, Extended Data Fig. 7.14i, and Suppl. Video 2. The DINO features themselves did not drive performance, as shuffling the features showed poor decoding (Extended Data Fig. 7.14j).

Lastly, we tested decoding from other HVAs and PPC using DINO features. Overall, decoding from V1 had the highest performance and PPC (VISrl) the lowest (Fig. 7.15g, Extended Data Fig. 7.14k). Given the high decoding performance of CEBRA, we tested if there was a particular V1 layer that was most informative. We leveraged CEBRA-Behavior by training models on each category and find that layer 2/3 and layer 5/6 show significantly higher decoding performance compared to layer 4 (one-way ANOVA, F(2,12)=9.88, p=0.003; Fig. 7.15h). Given the known cortical connectivity, this suggests that the non-thalamic input layers make frame information more explicit, perhaps via

feedback or predictive processing.

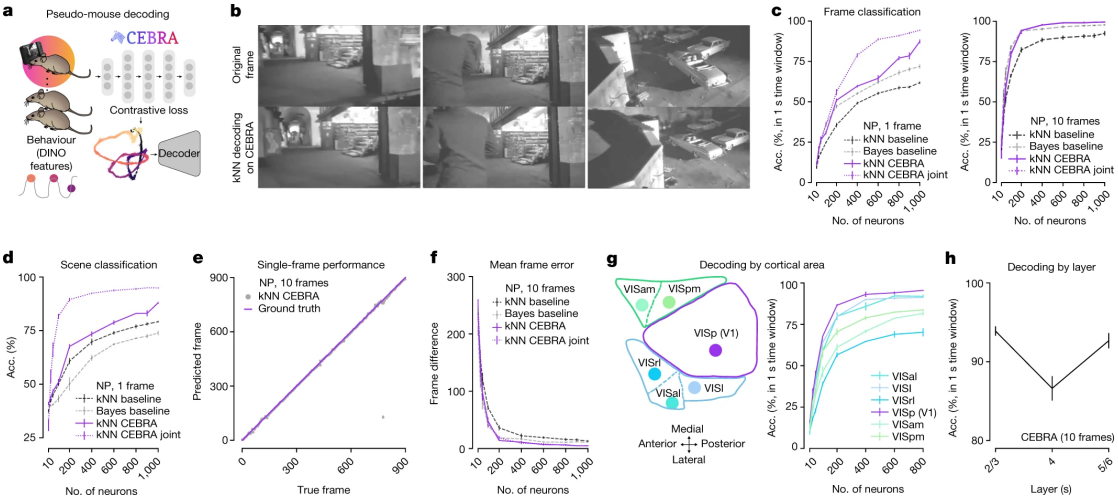


Figure 7.15: **Decoding of natural video features from mouse visual cortical areas. (a)**: Schematic of the CEBRA encoder and kNN (or naive Bayes) decoder. **(b)**: Examples of original frames (top row) and frames decoded from CEBRA embedding of V1 calcium recording using kNN decoding (bottom row). The last repeat among ten was used as the held-out test. **(c)**: Decoding accuracy measured by considering a predicted frame being within 1s of the true frame as a correct prediction using CEBRA (NP only), jointly trained (2P+NP) or a baseline population vector plus kNN or naive Bayes decoder using either a one-frame (33ms) receptive field (left) or ten frames (330ms) (right); results shown for Neuropixels dataset (V1 data); for each neuron number we have n=5 shuffles, mean ± s.e.m. **(d)**: Decoding accuracy measured by correct scene prediction using either CEBRA (NP only), jointly trained (2P+NP) or baseline population vector plus kNN or Bayes decoder using a one-frame (33ms) receptive field (V1 data); n=5 shuffles per neuron number, mean ± s.e.m. **(e)**: Single-frame ground truth frame ID versus predicted frame ID for Neuropixels using a CEBRA-Behaviour model trained with a 330ms receptive field (1,000 V1 neurons across mice used). **(f)**: Mean absolute error of the correct frame index; shown for baseline and CEBRA models as computed in (c)–(e). **(g)**: Diagram of the cortical areas considered and decoding performance from CEBRA (NP only), ten-frame receptive field; n=3 shuffles for each area and number of neurons, mean ± s.e.m. **(h)**: V1 decoding performance versus layer category using 900 neurons with a 330ms receptive field CEBRA-Behavior model; n=5 shuffles for each layer, mean ± s.e.m.

Discussion

CEBRA is a new non-linear dimensionality reduction method to explicitly leverage behavior labels and/or time in order to discover latent neural embeddings. The unique property of CEBRA is the extension and generalization of the standard InfoNCE objective by introducing a variety of different *sampling strategies* tuned for usage of the algorithm in the experimental sciences and for analysis of time series datasets, and it can be used for supervised and self-supervised analysis and thereby directly allows for hypothesis- and discovery-driven science. It produces both consistent embeddings across subjects (thus revealing common structure) and can find the dimensionality of

neural spaces that are topologically robust. While there remains a gap in understanding how these latent spaces map to neural-level computations, we believe this tool provides an advance in our ability to map behavior to neural populations. Moreover, pretrained CEBRA models can be used for decoding in new animals within tens of steps (milliseconds); we can thereby get equal or better performance compared to training on the unseen animal alone.

Dimensionality reduction is often tightly linked to data visualization, and here we make an empirical argument that ultimately this is only useful when you are getting consistent results, and discovering robust features. Unsupervised tSNE and UMAP are examples of algorithms widely used in life sciences for discovery-based analysis. However, they do not leverage time, and for neural recordings, this is always available and can be used. Even more critical is that concatenating data from different animals can lead to shifted clusters with tSNE or UMAP due to inherent small changes across animals or in how the data was collected. CEBRA allows the user to remove this unwanted variance and discover robust latents that are invariant to animal ID, sessions, or any-other-user-defined nuisance variable. Collectively, we believe CEBRA will become a complement to (or replacement for) these methods such that, at minimum, the structure of time in the neural code is leveraged, and robustness is prioritized.

Methods

Datasets

Artificial Spiking Dataset Synthetic spiking data for benchmarking in Fig. 7.1 was adopted from Zhou and Wei (2020). The continuous 1D behavior variable $c \in [0, 2\pi)$ was sampled uniformly in the interval $[0, 2\pi)$. The true 2D latent variable $\mathbf{z} \in \mathbb{R}^2$ was then sampled from a Gaussian distribution $\mathcal{N}(\mu(c), \Sigma(c))$ with mean $\mu(c) = (c, 2\sin c)^\top$ and covariance $\Sigma(c) = \mathrm{diag}(0.6 - 0.3|\sin c|, 0.3|\sin c|)$. After sampling, the 2D latent variable $\mathbf{z}$ was mapped to spiking rates of 100 neurons by applying four randomly initialized RealNVP (Dinh et al., 2016) blocks. Poisson noise was then applied (Zhou & Wei, 2020) to map firing rates onto spike counts. The final dataset consisted of 1.5×10^4 data points for 100 neurons ([number of samples, number of neurons]), and was split into train (80%) and validation (20%) sets. We quantified consistency across the entire dataset for all methods. The additional synthetic data, presented in Extended Data Fig. 1, was generated by varying the noise distribution in the above generative process. Beside Poisson noise, we used additive truncated ($[0, 1000]$) Gaussian noise with standard deviation 1 and additive uniform noise defined in $[0, 2)$ which was applied to the spiking rate. We also adapted the Poisson spiking by simulating neurons with a refractory period. For this, we scaled the spiking rates to an average rate of 110Hz. We sample inter-spike intervals from an exponential distribution with the given rate and add a refractory period of 10ms.

Rat Hippocampus Dataset We used the dataset presented in Grosmark and Buzsáki (2016). In brief, bilaterally implanted silicon-probes recorded multi-cellular electrophysiological data from the CA1 hippocampus areas from each of four male Long-Evans rats. During a given session, each rat independently ran on a 1.6 meter long linear track, where they were rewarded with water at each end of the track. The numbers of recorded putative pyramidal neurons for each rat ranged between 48 to 120. Here, we processed the data like Zhou and Wei (2020). Specifically, the spikes were binned into 25ms time windows. The position and running direction (left or right) of the rat was encoded into a 3D vector, which consisted of the continuous position value and two binary values indicating right or left direction. Recordings from each rat was parsed into trials (a round trip from one end of the track as a trial) and then split into a train, validation, and test set with a k=3 nested cross-validation scheme for the decoding task.

Macaque Dataset We used the dataset presented in Chowdhury et al. (2020). In brief, electrophysiological recordings were performed in Area 2 of somatosensory cortex (S1) in a rhesus macaque (monkey) during a center-out reaching task with a manipulandum. Specifically, the monkey performed an eight direction reaching task where on 50% of trials they actively made center-out movements to a presented target. The remaining trials were "passive" trials, where an unexpected 2N force bump was given to the manipulandum towards one of the eight target directions during a holding period. The trials were aligned as by Pei et al. (2021a), and we used the data from -100ms and 500ms from the movement onset. We used 1ms time bins and convolved the data with a Gaussian kernel with standard deviation of 40ms.

Mouse Visual Cortex Datasets We utilized the Allen Institute 2-photon calcium imaging and Neuropixels data recorded from five mouse visual cortical areas (VISp, VISl, VISal, VISam, VISpm) and one posterior parietal cortex (PPC)-like area (VISrl) during presentation of a black-and-white movie with 30 Hz frame rate, as presented previously (Deitch et al., 2021; de Vries et al., 2020; Siegle et al., 2021b). For calcium imaging (2P), we used the processed dataset by de Vries et al. (2020) with a sampling rate of 30 Hz, aligned to the video frames. We considered the recordings from excitatory neurons (Emx1-IRES-Cre, Slc17a7-IRES2-Cre, Cux2-CreERT2, Rorb-IRES2-Cre, Scnn1a-Tg3-Cre, Nr5a1-Cre, Rbp4-Cre_KL100, Fezf2-CreER, Tlx3-Cre_PL56) in the "Visual Coding-2P" dataset. Ten repeats of the first movie (Movie 1) were shown in all session types (A,B,C) for each mouse and we used the neurons that were recorded in all three session types, found by using the cell registration (de Vries et al., 2020). The Neuropixels recordings were obtained from the "Brain Observatory 1.1" dataset (Siegle et al., 2021b). We used the pre-processed spike-timings and binned them to a sampling frequency of 120 Hz, aligned with the movie timestamps (i.e., exactly 4 bins are aligned with each frame). The dataset contains recordings for 10 repeats, and we used the same Movie 1 that was used for the 2P recordings. For the analysis of consistency across the visual and

PPC cortical areas, we used a disjoint set of neurons for each seed to avoid higher intra-consistency due to overlapping neuron identities. We made 3 disjoint set of neurons by only considering neurons from session A (for 2P data) and non-overlapping random sampling for each seed.

CEBRA Model Framework

Notation We will use $\mathbf{x}, \mathbf{y}$ as general placeholder variables, and denote the multidimensional, time-varying signal as $\mathbf{ss}_t$, parameterized by the time t. The multidimensional, continuous context variable $\mathbf{c}_t$ contains additional information about the experimental condition and additional recordings, similar to the discrete categorical variable k_t.

The exact composition of $\mathbf{ss}$, $\mathbf{c}$ and k depends on the experimental context. CEBRA is agnostic to the exact signal types; with the default parameterizations, $\mathbf{ss}_t$ and $\mathbf{c}_t$ can have up to an order of hundred or thousand dimensions. For even higher dimensional datasets (e.g. raw video, audio, ...) other optimized deep learning tools can be used for feature extraction prior to the application of CEBRA.

Applicable problem setup We refer to $\mathbf{x} \in X$ as the *reference* sample, and to $\mathbf{y} \in Y$ as a corresponding *positive* or *negative* sample. Together, $(\mathbf{x}, \mathbf{y})$ form a positive or negative pair, based on the distribution $\mathbf{y}$ is sampled from. We denote the distribution and density function of $\mathbf{x}$ as $p(\mathbf{x})$, the conditional distribution and density of the positive sample $\mathbf{y}$ given $\mathbf{x}$ as $p(\mathbf{x}|\mathbf{y})$ and the conditional distribution and density of the negative sample $\mathbf{y}$ given $\mathbf{x}$ as $q(\mathbf{y}|\mathbf{x})$.

After sampling—and no matter whether we are considering a positive or negative pair—both samples $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{y} \in \mathbb{R}^{D'}$ are encoded by feature extractors $\mathbf{f} : X \mapsto Z$ and $\mathbf{f}' : Y \mapsto Z$. The feature extractors map both samples from signal space $X \subseteq \mathbb{R}^D, Y \subseteq \mathbb{R}^{D'}$ into a common embedding space $Z \subseteq \mathbb{R}^E$. The design and parameterization of the feature extractor is chosen by the user of the algorithm. Note that the spaces X and Y and their corresponding feature extractors can be the same (which is the case for single-session experiments in this work), but that this is not a strict requirement within the CEBRA framework (e.g., in multi-session training across animals or modalities, X and Y are selected to be signals from different mice or modalities, respectively). It is also possible to include the context variable (e.g., behavior) into X, or it is possible to set $\mathbf{x}$ to the context variable, and $\mathbf{y}$ to the signal variable.

Given two encoded samples, a similarity measure $\phi : Z \times Z \mapsto \mathbb{R}$ assigns a score to a pair of embeddings. The similarity measure needs to assign a higher score to more similar pairs of points, and have an upper bound. For this work, we consider the dot product between normalized feature vectors, $\phi(\mathbf{z}, \mathbf{z}') = \mathbf{z}^\top \mathbf{z}'/\tau$, in most analyses (latents on a hypersphere), or the negative mean squared error, $\phi(\mathbf{z}, \mathbf{z}') = -\|\mathbf{z} - \mathbf{z}'\|^2/\tau$ (latents in Euclidean space). Both metrics can be scaled by a temperature parameter τ which is either fixed, or jointly learned with the network. Other L_p norms and other similarity metrics, or even a trainable neural network (a so-called projection

head commonly used in contrastive learning algorithms, cf. Chen et al. (2020b) and Hyvärinen et al. (2019b)), are possible choices within the CEBRA software package. The exact choice of ϕ shapes the properties of the embedding space, and encodes assumptions about the distributions p and q.

The technique requires paired data recordings, e.g. as common in aligned time-series. The signal $\mathbf{ss}_t$, continuous context $\mathbf{c}_t$ and discrete context k_t are synced in their time-point t. How the reference, positive and negative samples are constructed from these available signals is a configuration choice made by the algorithm user, and depends on the scientific question to investigate.

Optimization Given the feature encoders $\mathbf{f}$ and $\mathbf{f}'$ for the different sample types, as well as the similarity measure ϕ, we introduce the shorthand $\psi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}'(\mathbf{y}))$. The objective function can then be compactly written as:

$$\int_{\mathbf{x}\in X} \mathrm{d}\mathbf{x} p(\mathbf{x}) \left[- \int_{\mathbf{y}\in Y} \mathrm{d}\mathbf{y} p(\mathbf{y}|\mathbf{x}) \psi(\mathbf{x}, \mathbf{y}) + \log \int_{\mathbf{y}\in Y} \mathrm{d}\mathbf{y} q(\mathbf{y}|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y})} \right] . \tag{7.1}$$

We approximate this objective (Chapter 6, Wang and Isola, 2020; Zimmermann et al., 2021a) by drawing a single positive example $\mathbf{y}_+$, and multiple *negative examples* $\mathbf{y}_i$ from the distributions outlined above, and minimize the loss function

$$\mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p(\mathbf{x}),\ \mathbf{y}_+\sim p(\mathbf{y}|\mathbf{x}) \\ \mathbf{y}_1,\ldots,\mathbf{y}_n\sim q(\mathbf{y}|\mathbf{x})}} \left[-\psi(\mathbf{x}, \mathbf{y}_+) + \log \sum_{i=1}^{n} e^{\psi(\mathbf{x},\mathbf{y}_i)} \right] , \tag{7.2}$$

with a gradient-based optimization algorithm. The number of negative samples is a hyperparameter of the algorithm and larger batch sizes are generally preferable.

For sufficiently small datasets as used in this paper, both positive and negative samples are drawn from all available samples in the dataset. This is in contrast to the common practice in many contrastive learning frameworks, where a mini-batch of samples is drawn first, which are then grouped into positive and negative pairs. Allowing to access the whole dataset to form the pairs gives a better approximation of the respective distributions $p(\mathbf{y}|\mathbf{x})$ and $q(\mathbf{y}|\mathbf{x})$, and considerably improves the quality of the obtained embeddings. If the dataset is small enough to fit into the GPU memory, CEBRA can be optimized with batch gradient descent, i.e., use the whole dataset at each optimizer step.

Goodness of fit Comparing the loss value—at both the absolute value and relative value across models at the same point in training time—can be used to determine the goodness of fit. Practically, this means one can find which hypothesis best fits one's data, in the case of using CEBRA-Behavior. Specifically, let us denote the objective in Eq. 7.1 as L_{asympt} and its approximation in Eq. 7.2 with a batch size of n as L_n. In the limit of

many samples, the objective converge up to a constant, $L_{\text{asympt}} = \lim_{n\to\infty}[L_n - \log n]$ (cf. Suppl. Note 2, and Wang and Isola, 2020).

The objective has also two trivial solutions: The first one is obtained for a constant $\psi(\mathbf{x}, \mathbf{y}) = \psi$, which yields a value of $L_n = \log n$. This solution can be obtained when the labels are not related to the signal (e.g., with shuffled labels). It is typically not obtained during regular training because the network is initialized randomly, causing the initial embedding points to be randomly distributed in space.

If the embedding points are distributed uniformly in space, and ϕ is selected such that $\mathbb{E}[\phi(\mathbf{x}, \mathbf{y})] = 0$, we will also get a value that is *approximately* $L_n = \log n$. The value can be readily estimated by computing $\phi(\mathbf{u}, \mathbf{v})$ for randomly distributed points.

The minimizer of Eq. 7.1 is also clearly defined as $-D_{\text{KL}}(p\|q)$ and depends on the positive and negative distribution. For discovery-driven (time contrastive) learning, this value is impossible to estimate because it would require access to the underlying conditional distribution of the latents. However, for training with predefined positive and negative distributions, this quantity can be again numerically estimated.

Interesting values of the loss function when fitting a CEBRA model are therefore

$$-D_{\text{KL}}(p\|q) \leq L_n - \log n \leq 0 \tag{7.3}$$

where $L_n - \log n$ is the goodness of fit (lower is better) of the CEBRA model. Note that the metric is independent of the batch size used for training.

Sampling Selection of the sampling scheme is CEBRA's key feature to adapt embedding spaces to different datasets and recording setups. The conditional distributions $p(\mathbf{y}|\mathbf{x})$ for positive samples and $q(\mathbf{y}|\mathbf{x})$ for negative samples as well as the marginal distribution $p(\mathbf{x})$ for reference samples are specified by the user. CEBRA offers a set of pre-defined sampling techniques, but customized variants can be specified to implement additional, domain specific distributions. This form of training allows to use the context variables to shape the properties of the embedding space, as outlined in the graphical model in Suppl. Note 1.

Through the choice of sampling technique, various use cases can be built into the algorithm: For instance, by forcing the positive and negative distributions to sample uniform across a factor, the model will become invariant to this factor, as including it would yield in a sub-optimal value of the objective function.

When considering different sampling mechanisms, we distinguish between *single-session* and *multi-session* datasets: A single-session dataset consists of samples $\mathbf{ss}_t$, which are associated to one or more context variables $\mathbf{c}_t$ and/or k_t. These context variables allow to impose structure on the marginal and conditional distribution used for obtaining the embedding. Multi-session datasets consist of multiple single-session datasets. The dimension of context variables $\mathbf{c}_t$ and/or k_t must be shared across all sessions, while the dimension of the signal $\mathbf{ss}_t$ can vary. In such a setting, CEBRA allows to learn a shared embedding space for signals from all sessions.

For single-session datasets, sampling is done in two steps: First, based on a specified "index" (the user-defined context variable $\mathbf{c}_t$ and/or k_t), locations t are sampled for reference, positive and negative samples. The algorithm differentiates between categorical (k) and continuous ($\mathbf{c}$) variables for this purpose.

In the simplest case, negative sampling (q) returns a random sample from the empirical distribution, by returning a randomly chosen index from the dataset. Optionally, with a categorical context variable $k_t \in [K]$, negative sampling can be performed to approximate a uniform distribution of samples over this context variable. If this is performed for both the negative and positive samples, the resulting embedding will become invariant with respect to the variable k_t. Sampling is performed in this case by computing the cumulative histogram of k_t, and sampling uniformly over k using the transformation theory for probability densities.

For positive pairs, different options exist based on the availability of continuous and discrete context variables. For a discrete context variable $k_t \in [K]$ with K possible values, sampling from the conditional distribution is done by filtering the whole dataset for the value k_t of the reference sample, and uniformly selecting a positive sample with the same value. For a continuous context variable $\mathbf{c}_t$, we use a set of time offsets Δ to specify the distribution. Given the time offsets, the empirical distribution $P(\mathbf{c}_{t+\tau}|\mathbf{c}_t)$ for a particular choice of $\tau \in \Delta$ can be computed from the dataset: We build up a set $D = \{t \in [T], \tau \in \Delta : \mathbf{c}_t - \mathbf{c}_{t+\tau}\}$, sample a $\mathbf{d}$ uniformly from D, and obtain the sample that is closest to the reference sample modified by this distance $\mathbf{d}$ from the dataset ($\mathbf{x} + \mathbf{d}$). It is possible to combine a continuous variable $\mathbf{c}_t$ with a categorical variable k_t for mixed sampling. On top of the continual sampling step above, it is ensured that both samples in the positive pair share the same value k_t.

It is crucial that the context samples $\mathbf{c}$ and the norm used in the algorithm match in some way; for simple context variables with predictable conditional distributions (e.g., a one or two-dimensional position of a moving animal, which can be most likely well described by a Gaussian conditional distribution based on the previous sample). An additional alternative is to use CEBRA also to *pre-process* the original context samples $\mathbf{c}$ and use the embedded context samples with the metric used for CEBRA training. This scheme is especially useful for higher dimensional behavioral data, or even complex inputs like video.

We next consider the multi-session case, where signals $\mathbf{ss}_t^{(i)} \in \mathbb{R}^{n_i}$ come from N different sessions $i \in [N]$ with session-dependent dimensionality n_i. Importantly, the corresponding continuous context variables $\mathbf{c}_t^{(i)} \in \mathbb{R}^m$ share the same dimensionality m, which makes it possible to relate samples across sessions. The multi-session setup is similar to mixed session sampling (if we treat the session ID as a categorical variable $k_t^{(i)} := i$ for all time steps t in session i). The conditional distribution for both negative and positive pairs is uniformly sampled across sessions, irrespective of session length. Multi session mixed sampling or multi session discrete sampling can be implemented analogously.

Besides the outlined sampling scheme, CEBRA is flexible to incorporate more

specialized sampling schemes. For instance, mixed single session sampling could be extended to additionally incorporate a dimension the algorithm should become invariant to. This would add an additional step of uniform sampling with regard to to this desired discrete variable (e.g., via ancestral sampling).

Choice of reference and positive and negative samples Depending on the exact application, the contrastive learning step can be performed by explicitly including or excluding the context variable: The reference sample $\mathbf{x}$ can contain information from the signal $\mathbf{ss}_t$, but also from the experimental conditions, behavioral recordings, or other context variables. The positive and negative samples $\mathbf{y}$ are set to the signal variable $\mathbf{ss}_t$.

Theoretical guarantees for linear identifiability of CEBRA models Identifiability describes the property of an algorithm to give a consistent estimate for the model parameters given that the data distributions match. We here apply the relaxed notion of *linear identifiability* that was previously discussed and used by Hyvärinen et al. (2019b) and Roeder et al. (2020a): After training two encoder models $\mathbf{f}$ and $\mathbf{f}'$, the models are linear identifiable if $\mathbf{f}(\mathbf{x}) = \mathbf{L}\mathbf{f}(\mathbf{x})$ where $\mathbf{L}$ is a linear projection.

When applying CEBRA, three cases are of potential interest. First, when applying discovery-driven CEBRA, will two models estimated on comparable experimental data agree in their inferred representation? Second, under which assumptions about the data will we be able to discover the *true* latent distribution? Third, in the hypothesis-driven or hybrid application of CEBRA, is the algorithm guaranteed to give a meaningful (non-standard) latent space when we can find signal within the data?

For the first case, we note that the CEBRA objective with a cosine similarity metric follows the canonical discriminative form for which Roeder et al. (2020a) showed linear identifiability: For sufficiently diverse datasets, two CEBRA models trained to convergence on the same dataset will be consistent up to linear transformations. Note that consistency of CEBRA is independent of the exact data distribution: The diversity condition merely requires that for any set of samples $\{\mathbf{y}_1, \ldots, \mathbf{y}_d\}$ from the negative distribution $q(\cdot|\mathbf{x})$, the matrices $[\cdots \mathbf{f}'(\mathbf{y}_i) - \mathbf{f}'(\mathbf{y}_i) \cdots]_{i=1}^{d}$ and the matrix $[\cdots \mathbf{f}(\mathbf{x}_i) \cdots]_{i=1}^{d+1}$ are invertible (i.e., the embeddings are sufficiently diverse). Alternatively, we can derive linear identifiability from assumptions about the data distribution: If the ground truth latents are sufficiently diverse (i.e., vary in all directions under the distributions p and q), and the model is sufficiently parameterized to fit the data, we will also obtain consistency up to a linear transformation. See Suppl. Note 2 for a full formal discussion and proofs.

For the second case, additional assumptions are required regarding the exact form of the data generating distribution. Within the scope of this work, we consider ground truth latents distributed on the hypersphere or Euclidean space. The metric then needs to match assumptions about the variation of the ground truth latents over time. In discovery-driven CEBRA, using the dot product as the similarity measure then encodes the assumption that latents vary according to a von-Mises-Fisher distribution, while

the mean squared error encodes an assumption that latents vary according to a Normal distribution. More broadly, if we assume that the latents have a uniform marginal distribution (which can be ensured by designing un-biased experiments), the similarity measure should be chosen as the log-likelihood of the conditional distribution over time. In this case, CEBRA identifies the data generating distribution up to affine transforms (in the most general case).

This result also explains the empirically high performance of CEBRA for decoding applications: If trained for decoding (using the variable to decode for informing the conditional distribution), it is trivial to select matching conditional distributions, as both quantities are directly selected by the user. CEBRA then "identifies" the context variable up to a linear transformation.

For the third case, we are interested in the hypothesis testing capabilities. We can show that if a mapping exists between the context variable and the signal space, CEBRA will recover this relationship and yield a meaningful embedding, which is also decodable. However, if such a mapping does not exist, we can show that CEBRA will instead learn a default embedding which is the uniform distribution of points on the hypersphere.

CEBRA models

We chose $X = Y$ to be the neural signal with varying amounts of recorded neurons and channels based on the dataset. We used three types of encoder models based on the required receptive field; a receptive field of one sample was used on the synthetic dataset experiments (Fig. 7.1b), a receptive field of 10 samples in all other experiments (rat, monkey, mouse) except for the Neuropixels dataset, where a receptive field of 40 samples is used due to the 4 times higher sampling rate of the dataset.

All feature encoders are parameterized by the number of neurons (input dimension), a hidden dimension to control the model size and capacity, as well as their output (embedding) dimension. For the model with the receptive field of one, a four layer MLP was used. The first and second layers map their respective inputs to the hidden dimension, while the third layer introduces a bottleneck and maps to half the hidden dimension. The final layer maps to the requested output dimension. For the model with receptive field of 10, a convolutional network with five time convolutional layers was used. The first layer had kernel size 2, the next three layers had kernel size 3 and used skip connections. The final layer had kernel size 3 and mapped the hidden dimensions to the output dimension. For the model with receptive field 40, we first preprocessed the signal by concatenating a $2\times$ downsampled version of the signal with a learnable downsample operation implemented as a convolutional layer with kernel size 4 and stride 2, directly followed (without activation function in between) by another convolutional layer with kernel size 3 and stride 2. After these first layers, the signal is subsampled by a factor of 4. Afterwards, similar to the receptive field 10 model, we apply three layers with kernel size 3 and skip connections, and a final

layer with kernel size 3. In all models, Gaussian error linear unit activation functions (GELU; Hendrycks and Gimpel, 2016a) were applied after each layer except the last. The feature vector was normalized after the last layer, unless a mean squared error (MSE) based similarity metric was used (as in Extended Data Fig 7.11).

Our implementation of the InfoNCE criterion received a mini-batch (or the full dataset) of size $n \times d$ for each of the reference, positive, and negative samples. n dot-product similarities are computed between reference and positive samples, $n \times n$ dot-product similarities are computed between reference and negative samples. The similarities were scaled with the inverse of the temperature parameter τ.

```
from torch import einsum, logsumexp, no_grad

def info_nce(ref, pos, neg, τ = 1.0):
  pos_dist = einsum("nd,nd->n",  ref, pos) / τ
  neg_dist = einsum("nd,md->nm", ref, neg) / τ
  with no_grad():
    c, _ = neg_dist.max(dim=1)
  pos_dist = pos_dist - c.detach()
  neg_dist = neg_dist - c.detach()
  pos_loss = -pos_dist.mean()
  neg_loss = logsumexp(neg_dist, dim=1).mean()
  return pos_loss + neg_loss
```

Alternatively, a learnable temperature can be used. For a numerically stable implementation, we store the log inverse temperature $\alpha = -\log \tau$ as a parameter of the loss function. At each step, we scale the distances in the loss function with $\min(\exp \alpha, 1/\tau_{\text{min}})$. The additional parameter τ_{min} is a lower bound on the temperature. The inverse temperature used for scaling the distances in the loss will hence lie in $(0, 1/\tau_{\text{min}}]$.

CEBRA Model parameters used In the main figures we used the default parameters (see https://cebra.ai/docs/api.html) for fitting CEBRA unless otherwise stated in the text (such as dimension, which varied and is noted in figures), or below.
Synthetic data: `model_architecture='offset1-model-mse'`, `conditional='delta'`, `delta=0.1`, `distance='euclidean'`, `batch_size=512`, `learning_rate=1e-4`.
Rat hippocampus: `model_architecture='offset10-model'`, `time_offsets=10`, `batch_size=512`.
Rat Behavioral Data: `model_architecture='offset10-model-mse'`, `distance='euclidean'`, `time_offsets=10`, `batch_size=512`.
Primate S1: `model_architecture='offset10-model'`, `time_offsets=10`, `batch_size=512`.
Allen datasets (2P): `model_architecture='offset10-model'`, `time_offsets=10`, `batch_size=512`.
Allen datasets (NP): `model_architecture='offset40-model-4x-subsample'`, `time_offsets=10`, `batch_size=512`.

CEBRA API and example usage The Python implementation of CEBRA is written in PyTorch (Paszke et al., 2019b) and NumPy (Walt et al., 2011) and provides an API which is fully compatible with scikit-learn (Pedregosa et al., 2011), a commonly used

package for machine learning. This allows to use scikit-learn tools for hyperparameter selection and downstream processing of the embeddings, e.g., decoding. CEBRA can be used as a drop-in replacement in existing data pipelines for algorithms like tSNE, UMAP, PCA or FastICA. Both CPU and GPU implementations are available.

Using the previously introduced notations, suppose we have a dataset containing signals $\mathbf{ss}_t$, continuous context variables $\mathbf{c}_t$ and discrete context variables k_t for all time steps t,

```
import numpy as np
N = 500
s = np.zeros((N, 55), dtype=float)
k = np.zeros((N,), dtype=int)
c = np.zeros((N, 10), dtype=float)
```

along with a second session of data,

```
s2 = np.zeros((N, 75), dtype=float)
c2 = np.zeros((N, 10), dtype=float)
assert c2.shape[1] == c.shape[1]
```

and note that the number of samples as well as the dimension in $\mathbf{ss}'$ does not need to match $\mathbf{ss}$. Session alignment leverages the fact that the second dimension of $\mathbf{c}$ and $\mathbf{c}'$ match. With this dataset in place, different variants of CEBRA can be applied as follows:

```
import cebra
model = cebra.CEBRA(
    output_dimension=8,
    num_hidden_units=32,
    batch_size=1024,
    learning_rate=3e-4,
    max_iterations=1000
)
```

The training mode to use is determined automatically based on what combination of data is passed to the algorithm:

```
# time contrastive learning
model.fit(s)
# discrete behavior contrastive learning
model.fit(s, k)
# continuous behavior contrastive learning
model.fit(s, c)
# mixed behavior contrastive learning
model.fit(s, c, k)
# multi-session training
model.fit([s, s2], [c, c2])
# adapt to new session
model.fit(s, c)
model.fit(s2, c2, adapt = True)
```

Since CEBRA is a parametric method training a neural network internally, it is possible to embed new data points after fitting the model:

```
s_test = np.zeros((N, 55), dtype=float)
# obtain and plot embedding
z = model.transform(s_test)
plt.scatter(z[:, 0], z[:, 1])
plt.show()
```

Besides this simple-to-use API for end users, our implementation of CEBRA is a modular software library including a plugin system, allowing more advanced users to easily add additional model implementations, similarity functions, datasets and data loaders, and distributions for sampling positive and negative pairs.

Consistency of embeddings across runs, subjects, sessions, recording modalities, and areas

To measure the consistency of the embeddings, we used the R^2 score of the linear regression (including an intercept) between the embeddings from different subjects (or sessions). Secondly, pi-VAE, which we benchmarked and improved (Extended Data Fig. 7.3), demonstrated a theoretical guarantee that it can reconstruct the true latent space up to an affine transformation. To measure across runs, we measured the R^2 score of the linear regression between embeddings across 10 runs of the algorithms, yielding 90 comparisons. The runs were done with the same hyperparameters, model, and training setup.

For the rat hippocampus data, the number of neurons recorded were different across subjects. The behavior setting was the same: the rats moved in a 1.6 meter long track, and for analysis the behavior data was binned into 100 bins with equal size for each direction (leftwards, rightwards). We computed averaged feature vectors for each bin by averaging all normalized CEBRA embeddings for a given bin, and re-normalized the average to lie on the hypersphere. If a bin does not contain any sample, it was filled by samples from the two adjacent bins. CEBRA was trained with latent dimension 3 (the minimum) such that it is constrained to lie only on a 2-sphere (making this "3D" space equivalent to 2D Euclidean space). All other methods were trained with 2 latent dimensions in Euclidean space. Note that $n + 1$ dimensions of CEBRA is equivalent to n dimensions of other methods that we compared, since the feature space of CEBRA is normalized (i.e., the feature vectors are normalized to have unit length).

For Allen visual data where the number of behavioral data points are the same across different sessions (i.e., fixed length of video stimuli), we directly computed the R^2 score of linear regression between embeddings from different sessions and the modalities. We surveyed 3, 4, 8, 32, 64, 128 latent dimensions with CEBRA.

To compare the consistency of embeddings between or within the areas we considered, we computed intra-area and inter-area consistency within the same recording modality (2P or NP). Within the same modality, we sampled 400 neurons from each area.

We trained one CEBRA model per area, and computed the linear consistency between all pairs of embeddings. For the intra-area comparison, we sampled an additional 400 disjoint neurons. For each area, we trained two CEBRA models on these two sets of neurons, and computed their linear consistency. We repeated this process three times.

For comparisons across modalities (2P and NP), we sampled 400 neurons from each modality (which are disjoint, as above, because one set was sampled from 2P recordings and the other set from the NP recordings). We trained a multi-session CEBRA model with one encoder for 2P, and one encoder for NP in the same embedding space. For an intra-area comparison, we computed the linear consistency between the the NP and 2P decoder from the same area. For an inter-area comparison, we computed the linear consistency between the NP encoder from one area and the 2P encoder from another area and again considered all combinations of areas. We repeated this process three times.

For the comparison of single- and multi-session training (Extended Data Fig. 7.9), we computed embeddings using encoder models with 8, 16, ..., 128 hidden units for varying the model size, and benchmark 8, 16, ..., 128 latent dimensions. Hyperparameters, except for number of optimization steps, were selected according to validation set decoding R^2 (rat) or accuracy (Allen). Consistency is reported at the point in training where the position decoding error is less than 7 cm for the first rat in the hippocampus dataset, and a decoding accuracy of 60% on the Allen dataset. For single-session training, four embeddings were trained independently on each of the individual animals, while for multi-session the embeddings were trained jointly on all sessions. For multi-session training, the same number of samples was drawn from each session to learn an embedding invariant to the session ID. The consistency vs. decoding error trade-off (Extended Data Fig. 7.9c) was reported as the average consistency across all 12 comparisons (Extended Data Fig. 7.9b) vs. the average decoding performance across all rats and data splits.

Model Comparisons

pi-VAE parameter selection, and modifications to pi-VAE The original implementation of pi-VAE used a single time bin spiking rate as a input. Thus, we modified their code to allow for larger time bin inputs and found that time window input with receptive field of 10 time bins (250 ms) gave a higher consistency across subjects and better preserved the qualitative structure of the embedding (thereby outperforming the results presented by Zhou and Wei (2020); see Extended Data Fig. 7.3). To do this, we used the same encoder neural network architecture as we used for CEBRA, and modified the decoder to a 2D output (we call our modified version conv-pi-VAE). Note, we used this modified pi-VAE for all the experiments except for the synthetic setting, where there is no time dimension, thus the original implementation is sufficient.

The original implementation reported a median absolute error of 12 cm on rat 1 (the animal they considered most in the work), and our implementation of time windowed

input with 10 bins resulted in a median absolute error of 11 cm (Fig. 7.6). For hyperparameters, we tested different epochs between 600 (the published value used) and 1000, and learning rate between 1.0×10^{-6} and 5.0×10^{-4} via a grid search. We fixed the hyperparameters to be those that gave the highest consistency across subjects, which were training epochs of 1000 and learning rate 2.5×10^{-4}. All other hyperparameters were kept as in the original implementation (Zhou & Wei, 2020). Note, that the original paper demonstrated that pi-VAE is fairly robust across different hyperparameters. For decoding (Fig. 7.6) we considered both a simple kNN decoder (that we use for CEBRA) and the computationally more expensive Monte Carlo sampling method originally proposed for pi-VAE (Zhou & Wei, 2020). Our implementation of conv-pi-VAE can be found at: https://github.com/AdaptiveMotorControlLab/CEBRA.

autoLFADS parameter selection AutoLFADS (Keshtkaran et al., 2022) includes a hyperparameter selection and tuning protocol, which we used, and we used the original implementation (https://github.com/snel-repo/autolfads-tf2/, https://github.com/neurallatents/nlb_tools/tree/main/examples/baselines/autolfads). For the rat hippocampus dataset, we chopped the continuous spiking rate (25ms bin size) into 250ms length segments with 225ms overlap between the segments to match the training setup for CEBRA, UMAP, tSNE and piVAE. We used Population Based Training (PBT) for hyperparameter searches and we constrained the search range to default values given in the original script (initial learning rate between 1.0×10^{-5} and 5.0×10^{-3}, dropout rate between 0.0 and 0.6, coordinated dropout rate between 0.01 and 0.7, L2 generator weight between 1.0×10^{-4} and 1.0, L2 controller weight between 1.0×10^{-4} and 1.0 and KL controller weight between 1.0×10^{-6} and 1.0×10^{-4} and KL initial condition weight between 1.0×10^{-6} and 1.0×-3). The negative log-likelihood metric was used to select the best hyperparameters. Each generation of PBT consisted of 25 training epochs and we trained for maximum 5000 epochs with batch size 100 while executing early-stopping after awaiting 50 epochs. PBT search was done using 20 parallel workers on each rat.

UMAP parameter selection For UMAP (McInnes et al., 2018), following the parameter guide (umap-learn.readthedocs.io/), we focused on tuning the number of neighbors (*n_neighbors*) and minimum distance (*min_dist*). The *n_components* parameter was fixed to 2 and we used a cosine metric to make a fair comparison with CEBRA, which also used the cosine distance metric for learning. We performed a grid search with 100 total hyperparameter values in the range of [2, 200] for *n_neighbors* and range of [0.0001, 0.99] for *min_dist*. The highest consistency across runs in the rat hippocampus dataset was achieved with *min_dist* of 0.0001 and *n_neighbors* of 24. For the other datasets in Extended Data Fig. 7.2, we used the default value of *n_neighbors* as 15 and *min_dist* as 0.1.

tSNE parameter selection For tSNE (Van Der Maaten et al., 2009), we used the implementation in openTSNE (Poličar et al., 2019). We performed a sweep on *perplexity* in the range of [5, 50] and *early_exaggeration* in the range [12, 32] following the parameter guide, while fixing *n_components* as 2 and used a cosine metric, to fairly compare to UMAP and CEBRA. We use PCA initialization to improve the run consistency of tSNE (Kobak & Linderman, 2021). The highest consistency across runs in the rat hippocampus dataset was achieved with *perplexity* of 10 and *early_exaggeration* of 16.44. For the other datasets in Extended Data Fig. 7.2, we used the default value of *perplexity* of 30 and *early_exaggeration* of 12.

Decoding Analysis

We primarily used a simple k-Nearest Neighbors (kNN) algorithm, which is a non-parametric supervised learning method, as a decoding method for CEBRA. We used the implementation in scikit-learn (Pedregosa et al., 2011). We used a kNN regressor for continuous value regression and a kNN classifier for discrete label classification, using uniform weights on distances of k-nearest neighbors. For the embeddings obtained with cosine metrics, we used cosine distance metrics for kNN and Euclidean distance metrics for the embeddings obtained in Euclidean space.

For the rat hippocampus data, a kNN regressor, as implemented in scikit-learn (Pedregosa et al., 2011), was used to decode the position, and a kNN classifier to decode the direction. The number of neighbors was searched over the range $\{1, 4, 9, 16, 25\}$ and we used the cosine distance metric. We used the R^2 score of predicted position and direction vector on the validation set as a metric to choose the best *n_neighbors* parameter. We report the median absolute error (MAE) for the positional decoding on the test set. For pi-VAE, we additionally evaluate decoding quality using the originally proposed decoding method based on Monte Carlo sampling, using the settings from the original paper (Zhou & Wei, 2020). For autoLFADS, using their default Ridge regression decoder (Keshtkaran et al., 2022) performed worse than our kNN decoder, which is why we reported all results for the kNN decoder. Note, UMAP, tSNE and CEBRA-Time were trained using the full dataset without label information when learning the embedding, and we used the above split only for training and cross-validation of the decoder.

For the direction decoding within the monkey dataset, we used a Ridge classifier (Pedregosa et al., 2011) as a baseline. The regularization hyperparameter was searched over $[10^{-6}, 10^{2}]$. For CEBRA, we used a kNN classifier for decoding direction with k searched over the range [1, 2500]. For conv-pi-VAE, we searched for the best learning rate over $[1.0 \times 10^{-5}, 1.0 \times 10^{-3}]$. For position decoding, we used Lasso (Pedregosa et al., 2011) as a baseline. The regularization hyperparameter was searched over $[10^{-6}, 10^{2}]$. For conv-pi-VAE, we used 600 epochs and searched for the best learning rates over $[5 \times 10^{-4}, 2.5 \times 10^{-4}, 0.125 \times 10^{-4}, 5 \times 10^{-5}]$, via a grid of (x,y) space in 1 cm bin for each axis as the sampling process for decoding. For CEBRA, we used the kNN

regression, and the number of neighbors k was again searched over [1, 2500].

For the Allen Institute datasets, we performed decoding (frame number or scene classification) for each frame from Movie 1. Here, we used a kNN classifier (Pedregosa et al., 2011) with a population vector kNN as a baseline, similar to the decoding of orientation grating as performed in (de Vries et al., 2020). For CEBRA, we used the same kNN classifier method on the CEBRA features. In both cases, the number of neighbors k was searched over a range of [1, 100] in an exponential fashion. We used the neural data recorded during the first 8 repeats as the train set, and the 9^{th} repeat for validation to choose the hyperparameter, and the last repeat as the test set to report the decoding accuracy. We also used a Gaussian Naive Bayes decoder (Pedregosa et al., 2011) to test linear decoding from the CEBRA model and neural population vector. Here, we assumed uniform priors over frame number and searched over a range of $[10^{-10}, 10^{3}]$ in an exponential manner for *smoothingvar* hyperparameter.

For layer specific decoding we used data from excitatory neurons in area VISp: layer 2/3 [Emx1-IRES-Cre, Slc17a7-IRES2-Cre]; layer 4 [Cux2-CreERT2, Rorb-IRES2-Cre, Scnn1a-Tg3-Cre]; layer 5/6 [Nr5a1-Cre, Rbp4-Cre_KL100, Fezf2-CreER, Tlx3-Cre_PL56, Ntrsr1-cre].

Neural Latent Benchmark (NLB)

We tested CEBRA on the following task from the NLB Benchmark (Pei et al., 2021c): mc-maze 20 ms https://eval.ai/web/challenges/challenge-page/1256/leaderboard/3183.We trained the offset10-model with 48 output dimensions and (128, 256, 512) hidden units, as presented throughout the paper. We trained in total 48 models by additionally varying temperature (0.0001, 0.004) and time offsets (1,2). We performed smoothing of the input neural data using a Gaussian kernel with 50ms standard deviation. Lastly, we took 45 embeddings from the trained models picked by the validation score, aligned the embeddings (using the Procrustes method (Schönemann, 1966)), and averaged the embeddings.

Topological Analysis

For the persistent co-homology analysis, we utilized ripser.py (Tralie et al., 2018b). For the hippocampus dataset we used 1,000 randomly sampled points from CEBRA-Behavior trained with temperature 1, time offset 10 and mini-batch size 512 for 10k training steps on the full dataset, and then analyzed up to the 2D co-homology. Maximum distance considered for filtration was set to infinity. To decide the number of co-cycles in each co-homology dimension with a significant lifespan, we trained 500 CEBRA embeddings with shuffled labels, similar to the approach by Gardner et al. (2022). We took the maximum lifespan of each dimension across these 500 runs as a threshold to determine robust Betti numbers. We surveyed the Betti numbers of CEBRA embeddings across 3, 8, 16, 32, and 64 latent dimensions.

Next, we used DREiMac (Tralie et al., 2018a) to obtain topology-preserving circular coordinates (radial angle) of the first co-cycle (H_1) from the persistent co-homology analysis. Similar to above, we used 1,000 randomly sampled points from the CEBRA-Behavior models of embedding dimensions 3, 8, 16, 32 and 64.

Behavior Embeddings for Video Datasets

High dimensional inputs, such as videos, need further pre-processing for effective use with CEBRA. Firstly, we used the recently presented DINO model (Caron et al., 2021a) to embed video frames into a 768-dimensional feature space. Specifically, we used the pretrained ViT/8 vision transformer model, which was trained by a self-supervised learning objective on the ImageNet database. This model is particularly well-suited for video analysis, and among the state-of-the-art models for embedding natural images into a space appropriate for k-nearest neighbour search (Caron et al., 2021a), a desired property to make the dataset compatible with CEBRA. We obtained a normalized feature vector for each video frame, which was then used as the continuous behavior variable for all further CEBRA experiments.

For scene labels, 3 individuals labeled each video frame using 8 candidate descriptive labels allowing multi-label classes. We took the majority vote of the 3 individuals to decide the label of each frame. In case of multi-labels, we considered this as a new class label. The above procedure resulted in 10 classes of frame annotation.

Acknowledgments

The authors thank Luisa Eck, Célia Benquet, Matthias Bethge, Alexander Mathis, Roland S. Zimmermann, Jakob Macke, Dmitry Kobak, Dylan Paiton, Jessy Lauer, Rodrigo González, and Gary Kane for discussions and feedback on earlier versions of the manuscript or code, and the Tübingen AI Center for computing resources. Funding was provided by SNSF grant no. 310030_201057, a Novartis Foundation for Medical-Biological Research Young Investigator Grant to MWM; Google PhD Fellowship to StS; the German Academic Exchange Service (DAAD) to JHL. StS acknowledges the IMPRS-IS Tübingen and ELLIS PhD program, and JHL thanks the TUM Program in Neuroengineering. MWM is the Bertarelli Foundation Chair of Integrative Neuroscience.

Conflicts

StS and MWM have filed a patent pertaining to the method presented in this work (filing no. 63/302,670). The authors declare no additional conflicts of interest. The funders had no role in the conceptualization, design, data collection, analysis, decision to publish, or preparation of the manuscript.

Data Availability

Hippocampus dataset: https://crcns.org/data-sets/hc/hc-11/about-hc-11 and we used the preprocessing script from https://github.com/zhd96/pi-vae/blob/main/code/rat_preprocess_data.py.
Primate dataset: https://gui.dandiarchive.org/#/dandiset/000127.
Allen Institute dataset: Neuropixels data are at https://allensdk.readthedocs.io/en/latest/visual_coding_neuropixels.html. The pre-processed 2P recordings are available at https://github.com/zivlab/visual_drift/tree/main/data.
As examples with CEBRA, packaged datasets are available at https://github.com/AdaptiveMotorControlLab/CEBRA.

Supplementary Note 1

On identifiability and consistency

When learning (non-linear) representations of a dataset, it is highly desirable that embedding algorithms generate *consistent* embedding spaces. Multiple runs of the algorithm on the same data, multiple runs of the algorithm on data produced in the same way, etc., should generate embedding spaces with a meaningful relation to each other. This "meaningful relation" between algorithm runs can be formalized using tools from identifiability in non-linear independent component analysis (ICA). Suppose we are given two models $\mathbf{f}'$ and $\mathbf{f}^*$ trained on the same dataset, and the performance of these models matches in the sense that they represent the same probability distribution $p' = p^*$. Identifiability then entails that both models are the same up to some known class of transformations (e.g., linear or affine transformations, rotations, permutations and sign-flips, etc.).

For example, one option for parameterizing the distributions is as $p'(\mathbf{y}|\mathbf{x}, \mathbf{y}_1 \dots \mathbf{y}_n) = \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{f}'(\mathbf{y}))/\sum_i \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{f}'(\mathbf{y}_i))$ and respectively for $\tilde{p}'$ defined respectively with $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{f}}'$. Roeder et al. (2020a) show that if the two distributions match contrastive learning models produce consistent embedding spaces, and it is possible to find a linear mapping $\mathbf{L}$ between the feature spaces, i.e., $\mathbf{L}\mathbf{f}(\mathbf{x}) = \tilde{\mathbf{f}}(\mathbf{x})$ for all $\mathbf{x}$ in the dataset. Other theoretical work has shown that contrastive learning with auxiliary variables is identifiable for bijective neural networks using the noise contrastive estimation (NCE) loss (Hyvärinen et al., 2019b), and that with an InfoNCE loss this bijectivity assumption can be removed for certain distributions (Zimmermann et al., 2021a). We will adapt the underlying proofs to our setup in Suppl. Note 2, and give a high-level outline below.

We will consider two important points in both the context of discovery and hypothesis driven training of CEBRA models. Firstly, when applying discovery-driven CEBRA, will two models estimated on comparable experimental data agree in their inferred representation? Second, under which assumptions about the data will we be able able to discover the *true* latent distribution?

CONSISTENCY: For consistency across embedding spaces, we require a dataset with a sufficient amount of variability in time. Intuitively, to estimate a d dimensional embedding that is consistent across runs, points sampled from the embedding via the negative distribution q need to vary in at least d directions for each possible reference sample in the dataset. Interestingly, consistency is mostly independent from the data generating process (i.e., the data modality of the recording) and merely requires a sufficiently varying dataset, as well as a choice of feature encoder that passes this variability on to the embedding space.

For example, consider the reaching dataset in Fig. 3 where we showed embeddings that vary in two dimensions (the direction and distance from the center). In this case, the sampling process needs to be designed such that for each reference point we can draw from the dataset, the embedding of the negative samples will vary in at least two directions. This is clearly the case for our training setup: The neurons encode both position and direction information, this information is transformed by the feature encoder, and the resulting embedding varies in at least two directions when the negative distribution samples uniformly across the dataset.

For the first property, we can leverage previous results on the consistency of contrastive learning models over multiple runs (Roeder et al., 2020a). Consider the case where we train multiple CEBRA models on data originating from the same data distribution, and consider that we can train these models to full convergence. It is then guaranteed that the embedding spaces will agree up to a linear indeterminacy. In other words, it will always be possible to transform one embedding space into the other by applying a linear transformation. Linear consistency of representations is interesting when we consider linear downstream processing of the inferred embedding space, as is common in neuroscience (Urai et al., 2022b). Such a downstream algorithm (e.g., a linear regression or general linear model) will yield the same performance across different CEBRA models.

RECOVERING THE GROUND-TRUTH LATENTS: Note that this notion of consistency only makes a statement about the *inferred* latent representation (and identifiability) across multiple runs of the algorithm, but not yet about the relation between this latent representation and the *true* underlying latent variables that generated the data. This is the second property mentioned above, and requires additional assumptions about the data generating process to resolve the ambiguity of what a "latent" underlying a given dataset actually entails. The assumptions concern the injectivity of the data generating process and the positive distribution p. While for discovery driven training, p is an empirical property of the dataset, hypothesis driven training allows to precisely define p based on the observed auxiliary variables. The same theory applies to both cases. Importantly, as for consistency, note that these results are independent from the actual modality of the data and other properties of the signal space we consider. The assumptions are all with respect to the underlying latent distribution.

For the analyses in this paper, the theory for linear identifiability of the underlying latents applies: For time-contrastive training, it is required that the underlying latent

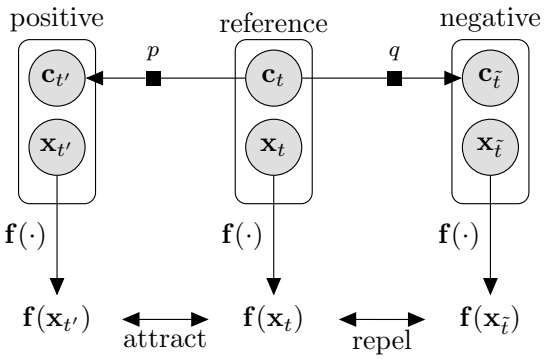


Figure 7.16: The contrastive learning data sampling scheme used in CEBRA.

distribution is uniform (e.g., there is no inherent bias in the experimental data), and that latents of nearby time steps vary according to a distribution of the form $p(\mathbf{v}|\mathbf{u}) = \exp(\phi(\mathbf{u}, \mathbf{v}))$, where $\mathbf{u}$ and $\mathbf{v}$ are the latents underlying the signal variables $\mathbf{x}$ and $\mathbf{y}$. If these conditions are met, the true underlying latents are recovered up to a linear transformation. For hypothesis testing where the user actually specifies the distribution p, this requirement can be easily validated and met.

Relationship between consistency, identifiability, and the sampling mechanism: The CEBRA software package allows for other choices of similarity measures (potentially learnable), which allows to derive guarantees also for these cases. In the most general case, we pick ϕ as a trainable neural network that factorizes into individual components, $\phi(\mathbf{x}, \mathbf{y}) := \sum_i \phi_i(y_i, \mathbf{x})$ where each ϕ_i is an individually trained neural network. For sufficiently variable distributions, this allows to recover the underlying latents up to permutations and point-wise non-linear, bijective transformations.

Likewise, it is possible to modify the encoding networks $\mathbf{f}$ and $\mathbf{f}'$ for $\mathbf{x}$ and $\mathbf{y}$, respectively. While our experiments used one network $\mathbf{f} = \mathbf{f}'$ with $\mathbf{x}$ and $\mathbf{y}$ representing neural data, it is well possible to encode different aspects of the dataset and/or to break the symmetry between the two encoding networks and to train a separate network for each of $\mathbf{f}$ and $\mathbf{f}'$. For example, neural data $\mathbf{y}$ could be encoded using $\mathbf{f}'$, and behavior $\mathbf{x}$ could be encoded using $\mathbf{f}$. It would also be possible to use a composition of neural and behavioral data for $\mathbf{x}$. In these cases, if $\mathbf{f}$ and $\mathbf{f}'$ are parameterized as two individual neural networks and ϕ is defined as the dot-product as before, if $\mathbf{y}|\mathbf{x}$ follows a conditionally exponential distribution, we are able to recover all sufficient statistics of this distribution up to a linear transformation.

Examples: As an example of the aforementioned results, let us consider the rat hippocampus dataset used in Fig. 1 and 2. The auxiliary information in this dataset is the position, velocity, and direction of the rat on the linear track.

We can apply different sampling schemes for investigating this dataset. For example, we can apply discovery-driven, time-contrastive learning. In this setup, we sample time steps uniformly from the dataset to arrive at our reference samples. Given a time offset Δ (that informs the algorithm about the time-scale of interest), we obtain

positive samples. The resulting batch will be composed of samples $\mathbf{ss}_t$ for the reference, $\mathbf{ss}(t+\Delta)$ for the positive, and $\mathbf{ss}(t_i)$ with uniformly sampled time steps $t_1, \ldots, t_n$ for the negative samples. This corresponds to an approximation of the distribution $p(\mathbf{u}_{t+\Delta}|\mathbf{u}_t)$ of how the latents vary over the course of time. If sufficient variation is present in the dataset along d latent directions, CEBRA models will become, after training, consistent across runs. If additionally the *true* distribution $p(\mathbf{u}_{t+\Delta}|\mathbf{u}_t)$ follows, e.g., a Gaussian distribution, CEBRA will identify the ground truth latents.

In comparison, our so-called hypothesis-driven, behavior-label guided contrastive learning approach would leverage the continuous position information as well as the movement direction of the rat. In the Methods, we denoted the continuous variable as $\mathbf{c}_t$ and the discrete variables as k_t. To arrive at a behavior contrastive embedding using this auxiliary information, we would build a set of differences. If the variables are independent (or should reflect this in the embedding), we build one set $D = \{\mathbf{c}_{t+\Delta} - \mathbf{c}_t\}_{t=1}^{T}$. For a reference sample at time step t, we sample $\mathbf{d} \sim D$ uniformly, and apply this difference to the position $\mathbf{c}_t$ at step t. We then pick the point closest to $\mathbf{c}_t + \mathbf{d}$ *and* matching the discrete variable k_t as the positive sample.

A lot of variations of this sampling process are possible to embed desirable properties and test hypotheses about the dataset. For instance, consider the primate reaching dataset: Here, $\mathbf{c}_t$ could be selected as the x/y position in space, and the discrete label k_t could denote the reaching direction. However, the 2D differences $\mathbf{c}_{t+\Delta} - \mathbf{c}_t$ will depend on k_t: A reaching direction towards the left will have most variance in negative x-direction $[-1, 0]^\top$, a reaching direction towards the top will have most variance in positive y-direction $[0, 1]^\top$. One way to work around this issue is to consider a polar representation of the position and direction and apply the scheme outlined above. Another alternative is to build the set of differences conditional on the direction k_t, i.e., $D(k) = \{\mathbf{c}_{t+\Delta} - \mathbf{c}_t\}_{t:k_t=k}$. The sampling process is almost analogous to the rat hippocampus example above: We would sample a time step t, look up the discrete variable k_t, but then only sample from $D(k_t)$ to reflect the conditioning on the direction.

Finally, e.g., for very complex movements, it is simple to adapt additional pre-processing schemes. These could involve other deep learning algorithms like DINO used for pre-processing video data in the Allen dataset (to convert pixel data without a meaningful metric into an embedding space with desirable distance properties); they could also involve simpler processing, such as computing the principal component analysis of a higher dimensional dataset, and using the behavior data in this space.

Many other variations are possible. While the most common use cases are reflected in the CEBRA software toolbox and high-level API and readily usable, more customized use cases can be easily added by the user thanks to a straightforward extension mechanism.

Improving pi-VAE

Zhou et al (Zhou & Wei, 2020) demonstrate that pi-VAE outperforms LFADS (Pandarinath et al., 2018b), demixed-PCA (Kobak et al., 2016b), UMAP (McInnes et al., 2018), PCA, and pfLDS (Gao et al., 2016a) using the rat and/or primate datasets (Extended Data Fig. 1). We improved the performance of pi-VAE by modifying the encoder, which allows for longer time inputs (Extended Data Fig. 1), and this improved version is used throughout, unless noted.

Comparison to autoLFADS

The overall goal of CEBRA and LFADS is different. In LFADS/autoLFADS (Keshtkaran et al., 2022; Pandarinath et al., 2018b) the primary goal is to build high performance spiking rate predictions and downstream decoding, which are excellent tasks for auto-encoders. It is important to note that LFADS assumes a data generating process that is Poisson, as does pi-VAE (Zhou & Wei, 2020), and requires trial-data. A major assumption in LFADS is that from the derived factors there is a linear projection to the neural rate predictions. In contrast, as we show, CEBRA can be used on many data types, from spiking neural data, calcium imaging, or behavior-only (such as from pose estimation tools, see Fig. 2b), and CEBRA does not require trials. Another consideration is that one cannot use theoretically-motivated goodness of fit metrics for finding latent embedding spaces that best fit the data. While this issue is partly addressed during parameter selection in autoLFADS, it also requires a substantial computational overhead for hyperparameter tuning. Thus, while LFADS is excellent at several spike-prediction tasks (Pei et al., 2021a), it is not fully suitable for the range of settings that can be used with CEBRA.

Utilizing CEBRA across contexts

Within our framework, we assume that independent latent variables are combined by a non-linear bijective mixing function to produce neural activity. The latent variables are assumed to change over time, or be correlated to the observed auxiliary variables used to train CEBRA. No additional special structure, or implicit generative models during training are needed.

CEBRA allows for minimizing the impact of selected features on the embedding, while testing the role of others. For example, suppose you have neural data from four different animals, each from the hippocampus while the animal navigated a linear track. You hypothesize that the hippocampus encodes a continuous mapping of space along the track. In this scenario the animal ID is not important, but the spatial location of the animal is. Here, the user can specify to obtain an embedding that is invariant to the animal ID, but should incorporate the position information. Another amendable scenario is a hypothesis-free, discovery-driven approach (akin to unsupervised clustering). Here too, CEBRA can be used, with only time as the input

(Fig. 1). Collectively, CEBRA can be used for both visualization of data and latent-space based embedding of neural activity for downstream tasks like decoding.

The flexibility in choosing different auxiliary variables during data analysis allows users to leverage the same algorithm for a variety of applications on a given dataset: Discovery-driven analysis by purely self-supervised learning with time-contrastive learning, hypothesis-driven analysis by comparing embedding quality derived from different behavioral variables, or replacing supervised decoding algorithms, e.g., in brain-machine-interface contexts.

Supplementary Note 2

Here we provide theoretical results for consistency and identifiability of models trained within the CEBRA framework. We proceed by showing properties of the InfoNCE loss (Prop. 2), and use them as the basis for showing that encoders trained on this loss function will become bijective under mild assumptions (Prop. 3). We then revisit existing theory on contrastive learning, and show that CEBRA falls into a category of models for which we can obtain theoretical guarantees on both consistency (Prop 4) across different model runs and identifiability of the ground truth latent distribution for both the discovery-driven (time-contrastive) learning mode (Prop. 3) and the hypothesis-driven mode (Prop. 8). Our results leverage theory by Hyvärinen et al. (2019b), Roeder et al. (2020a), Wang and Isola (2020), and Zimmermann et al. (2021a).

It should be noted that while consistency results between model runs do not require strong assumptions about the underlying data generating process, understanding the relation between the embedding space given by CEBRA and the underlying *ground truth data generating process* naturally requires such assumptions. However, compared to assumptions in generative models (e.g., VAEs), these assumptions concern the relationship between the ground-truth latent variables, rather than making statements about the signal space.

Notation, data generation, and learning algorithm

We will use the notation presented in the Methods Section. We additionally introduce the latents $\mathbf{u}$ and $\mathbf{v}$ underlying the samples $\mathbf{x}$ and $\mathbf{y}$. We will interchangeably use the distributions $p_{\mathcal{D}}$, p and q for either the latents $\mathbf{u}$ and $\mathbf{v}$ or their respective samples $\mathbf{x}$ and $\mathbf{y}$ depending on their arguments (we will show after Proposition 2 that this treatment is also formally correct due to the training setup considered here). Definitions, propositions and theorems adapted from other works are cited and denoted by upper-case letters and are otherwise adapted to our notation.

Definition 2 (Data generating process and encoder)**.** *Let* $\mathbf{u} \in \mathbb{R}^d, \mathbf{v} \in \mathbb{R}^{d'}$ *denote latents corresponding to the samples* $\mathbf{x} \in \mathbb{R}^D$ *and* $\mathbf{y} \in \mathbb{R}^{D'}$ *in the respective signal space which are generated according to two differentiable and injective mixing functions* $\mathbf{g} : \mathbb{R}^d \mapsto \mathbb{R}^D$ *and*

$\mathbf{g}' : \mathbb{R}^{d'} \mapsto \mathbb{R}^{D'}$,

$$\mathbf{x} = \mathbf{g}(\mathbf{u}), \quad \mathbf{y} = \mathbf{g}'(\mathbf{v}) \tag{7.4}$$

and there exist optimal differentiable encoders $\mathbf{f} : \mathbb{R}^D \mapsto \mathbb{R}^E$ *and* $\mathbf{f}' : \mathbb{R}^{D'} \mapsto \mathbb{R}^E$ *such that*

$$f(\mathbf{g}(\mathbf{u}))_i = u_i \quad f'(\mathbf{g}'(\mathbf{v}))_j = v_j. \tag{7.5}$$

We will refer to the composition of the data generators and the encoders as $\mathbf{h} = \mathbf{f} \circ \mathbf{g}$ *and* $\mathbf{h}' = \mathbf{f}' \circ \mathbf{g}'$. *In setups where two models are trained on potentially different mixing functions, we denote the second data generator, feature encoder, and the composition of both as* $\tilde{\mathbf{h}} = \tilde{\mathbf{f}} \circ \tilde{\mathbf{g}}$ *and* $\tilde{\mathbf{h}}' = \tilde{\mathbf{f}}' \circ \tilde{\mathbf{g}}'$.

We consider a marginal distribution $p_{\mathcal{D}}(\cdot)$, the positive sample conditional distribution $p(\cdot|\cdot)$ and the negative conditional distribution $q(\cdot|\cdot)$. Reference samples $\mathbf{u}$ from the (true) latent space are mapped to signal space by the injective function $\mathbf{g}$, positive/negative samples $\mathbf{v}$ from the (potentially different latent space) are mapped to (a possibly different) signal space $\mathbf{y}$ by the injective function $\mathbf{g}'$. The encoder $\mathbf{f}$ is applied to $\mathbf{x}$ and the encoder $\mathbf{f}'$ is applied to $\mathbf{y}$ to recover the respective latents underlying $\mathbf{x}$ and $\mathbf{y}$. The similarity measure is denoted as ϕ with $\mathbf{f}(\mathbf{x})$ and $\mathbf{f}'(\mathbf{y})$ as its arguments. Note that ϕ does not need to be a fixed function and can also be parameterized by a learnable neural network. As in the Methods, we denote $\psi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}'(\mathbf{y}))$ and additionally introduce $\psi(\mathbf{u}, \mathbf{v}) := \phi(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v}))$ without additional subscripts, as the desired shortcut will be clear from the context and its arguments.

Note that this is a very general setup. We typically would assume that the number of dimensions in the (shared) latent space is the same, $d = d'$, and could further assume that the number of dimensions E of the embedding space is also matching.

We recall the contrastive objective in the limit of unlimited samples from the main text:

Definition 3 (Generalized InfoNCE objective)**.** *In the limit of unlimited negative samples, the InfoNCE objective is a functional*

$$L[\psi]_{\text{asympt}} = \int p_{\mathcal{D}}(\mathbf{x}) \left[\log \int q(\mathbf{y}|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y})} d\mathbf{y} - \int p(\mathbf{y}|\mathbf{x}) \psi(\mathbf{x}, \mathbf{y}) d\mathbf{y} \right] d\mathbf{x}, \tag{7.6}$$

depending on the positive sample conditional density $p(\mathbf{y}|\mathbf{x})$, *the negative sample density* $q(\mathbf{y}|\mathbf{x})$, *the marginal density* $p_{\mathcal{D}}(\mathbf{x})$ *and the embedding similarity* ψ *as defined above.*

We call this objective "generalized" as it extends the original definition of the InfoNCE loss used in the literature. Oord et al. (2018) introduced an objective where the marginal (there: prior) $p_{\mathcal{D}}$ and the negative sample distribution q matched, which influences the types of functions that can be learned. The discussion of the objective by Wang and Isola (2020) makes stronger assumptions about the nature of the conditional distribution p for the positive pair, and only considers uniform choices for the marginal $p_{\mathcal{D}}$ and negative conditional q.

Overall, in CEBRA, the key difference to prior uses of the InfoNCE objective is the ability to control the properties of the embedding space through p and q and ϕ, and leverage this to retrieve the discovery-driven, hypothesis-driven, and hybrid modes as demonstrated in the main text. In fact, for hypothesis-driven training, the distributions used for sampling do not even need to be connected to the underlying data generating process—instead, varying p and testing to enforce various neighbourhood relations on the neural data is used as a tool to discover meaningful relations between the auxiliary variables (e.g., behavior) and signal (e.g., neural activity). In the same manner, p and q can be selected such that a particular factor is sampled uniformly, to enforce invariance (e.g. across a subject, or a modality variable).

The loss optimized in practice acts on a limited number of negative samples in each mini-batch:

Definition 4 (Generalized InfoNCE objective with limited batch size)**.** *For a fixed number of negative samples n, the InfoNCE objective is the functional*

$$L[\psi]_n = \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x}),\ \mathbf{y}_+\sim p(\mathbf{y}|\mathbf{x}) \\ \mathbf{y}_1,\dots,\mathbf{y}_n\sim q(\mathbf{y}|\mathbf{x})}} \left[-\psi(\mathbf{x},\mathbf{y}_+) + \log \sum_{i=1}^{n} e^{\psi(\mathbf{x},\mathbf{y}_i)} \right],$$

depending on the positive sample conditional density $p(\mathbf{y}|\mathbf{x})$, the negative sample density $q(\mathbf{y}|\mathbf{x})$, the marginal density $p_{\mathcal{D}}(\mathbf{x})$ and the embedding similarity ψ as defined above.

Both losses can be related due to Theorem 1 by Wang and Isola (2020). In the limit of unlimited samples $n \to \infty$, we obtain for the batch size n:

$$L[\psi]_{\text{asympt}} = \lim_{n\to\infty} \left(\mathcal{L}[\psi]_n - \log n \right). \tag{7.7}$$

For a sufficiently large batch size, we can leverage the quantity $\mathcal{L}[\psi]_n - \log n$ as a goodness of fit measure (as outlined in the Methods) that estimates the distance from a "default" embedding. When comparing models with equal batch size n, note that the InfoNCE loss can also directly serve as this metric.

Minimizers of the generalized InfoNCE loss

In this section, we show that optimizing the generalized InfoNCE objective from Def. 3 yields the unique minimizer $\psi(\mathbf{x},\mathbf{y}) = C(\mathbf{x}) + \log p(\mathbf{y}|\mathbf{x})/q(\mathbf{y}|\mathbf{x})$ or equivalently $\psi(\mathbf{u},\mathbf{v}) = C'(\mathbf{u}) + \log p(\mathbf{v}|\mathbf{u})/q(\mathbf{v}|\mathbf{u})$ up to an arbitrary constant function $C'(\mathbf{u})$ than can depend on the latents of the reference latents. In this regard, the InfoNCE loss is more flexible than the standard noise constrastive estimation (NCE) loss which has a similar minimizer, but is limited to $C(\mathbf{x}) = C'(\mathbf{u}) = 0$. The minimum loss value is the negative Kullbach-Leibler divergence between the positive and negative distributions. To obtain non-trivial solutions, it is therefore important that p and q differ in a non-trivial way (which is the case for both time-contrastive and behavior-contrastive sampling outlined in the context of CEBRA).

Proposition 2. *Let $p(\cdot|\cdot)$ be the conditional distribution of the positive samples, $q(\cdot|\cdot)$ the conditional distribution of the negative samples and $p_{\mathcal{D}}(\cdot)$ the marginal distribution of the reference samples. The generalized InfoNCE objective (Def. 3) is convex in ψ with the unique minimizer*

$$\psi^*(\mathbf{x}, \mathbf{y}) = \log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} + C(\mathbf{x}), \quad \textit{with} \quad L[\psi^*]_{\text{asympt}} = -\mathcal{D}_{\text{KL}}(p(\cdot|\cdot)\|q(\cdot|\cdot)) \tag{7.8}$$

on the support of $p_{\mathcal{D}}$, where $C : \mathbb{R}^d \to \mathbb{R}$ is an arbitrary mapping.

Proof. We rewrite the objective as

$$L[\psi]_{\text{asympt}} = \int p_{\mathcal{D}}(\mathbf{x}) \left[\log \int q(\mathbf{y}|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y})} d\mathbf{y} - \int p(\mathbf{y}|\mathbf{x}) \psi(\mathbf{x}, \mathbf{y}) d\mathbf{y} \right] d\mathbf{x}, \tag{7.9}$$

and we can compute the first-order functional derivative (using the method discussed in Cahill 2014[1])

$$\begin{aligned} &\delta L[\psi][h]_{\text{asympt}} \\ &= \frac{d}{d\epsilon} \mathcal{L}[\psi + \epsilon h]\big|_{\epsilon=0} \\ &= \int p_{\mathcal{D}}(\mathbf{x}) \left[\frac{1}{Z_\psi(\mathbf{x})} \int q(\mathbf{y}|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y})} h(\mathbf{x}, \mathbf{y}) d\mathbf{y} - \int p(\mathbf{y}|\mathbf{x}) h(\mathbf{x}, \mathbf{y}) d\mathbf{y} \right] d\mathbf{x}, \\ &\text{with } Z_\psi(\mathbf{x}) = \int q(\mathbf{y}'|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y}')} d\mathbf{y}'. \end{aligned} \tag{7.10}$$

The first-order functional derivative vanishes for all functions $h(\mathbf{x}, \mathbf{y})$ whenever

$$p_{\mathcal{D}}(\mathbf{x}) \left[\frac{1}{Z_\psi(\mathbf{x})} q(\mathbf{y}|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y})} - p(\mathbf{y}|\mathbf{x}) \right] = 0. \tag{7.11}$$

This is the case iff at any point $(\mathbf{x}, \mathbf{y})$, either $p_{\mathcal{D}}(\mathbf{x}) = 0$ or

$$Z_\psi(\mathbf{x}) = \frac{q(\mathbf{y}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x})} e^{\psi(\mathbf{x},\mathbf{y})}. \tag{7.12}$$

Since the left hand side of Eq. (7.12) is independent of $\mathbf{y}$, the right hand side must be independent of $\mathbf{y}$ as well. Hence all functions $\psi^*(\mathbf{x}, \mathbf{y})$ which are solutions to Eq. (7.12) are of the form

$$\psi^*(\mathbf{x}, \mathbf{y}) = \log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} + C(\mathbf{x}), \tag{7.13}$$

where C is an arbitrary function depending only on $\mathbf{x}$ and not on $\mathbf{y}$. Then by definition of $Z_\psi(\mathbf{x})$,

$$Z_\psi(\mathbf{x}) = \int q(\mathbf{y}'|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y}')} d\mathbf{y}' = e^{C(\mathbf{x})} \tag{7.14}$$

[1] http://quantum.phys.unm.edu/523-14/ch15.pdf

which is consistent when inserted into Eq. (7.12). Therefore the minimizers of $L[\psi]_{\text{asympt}}$ form a convex connected set $\mathcal{M}$,

$$\mathcal{M} = \left\{ \psi^* : \psi^*(\mathbf{x}, \mathbf{y}) = \begin{cases} \log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} + C(\mathbf{x}) & \text{if } p_{\mathcal{D}}(\mathbf{x}) \neq 0 \\ f(\mathbf{x}, \mathbf{y}) & \text{if } p_{\mathcal{D}}(\mathbf{x}) = 0 \end{cases} \right\}, \tag{7.15}$$

where C, f are arbitrary functions. It can be checked that all minima achieve the same value $L[\psi^*]$ given by

$$\begin{aligned} L[\psi^*]_{\text{asympt}} &= -\int p_{\mathcal{D}}(\mathbf{x}) \int p(\mathbf{y}|\mathbf{x}) \log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} d\mathbf{y} \\ &= -\int p_{\mathcal{D}}(\mathbf{x}) D_{\text{KL}} \left[p(\cdot|\mathbf{x}) || q(\cdot|\mathbf{x}) \right] d\mathbf{x} \leq 0. \end{aligned} \tag{7.16}$$

It is left to show that the objective function is convex. The second-order functional derivative is given by

$$\begin{aligned} \delta^2 L[\psi][h]_{\text{asympt}} =& \frac{d^2}{d\epsilon^2} \mathcal{L}[\psi + \epsilon h]\big|_{\epsilon=0} \\ =& \frac{d}{d\epsilon} \int p(\mathbf{x}) \left[\frac{1}{Z_{\psi+\epsilon h}(\mathbf{x})} \int q(\mathbf{y}|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y}) + \epsilon h(\mathbf{x},\mathbf{y})} h(\mathbf{x}, \mathbf{y}) d\mathbf{y} - \int p(\mathbf{y}|\mathbf{x}) h(\mathbf{x}, \mathbf{y}) d\mathbf{y} \right] \Bigg|_{\epsilon=0} \\ =& \int p(\mathbf{x}) \left[\mathbb{E}_{g(\mathbf{y}|\mathbf{x})} \left[h(\mathbf{x}, \mathbf{y})^2 \right] - \mathbb{E}_{g(\mathbf{y}|\mathbf{x})} \left[h(\mathbf{x}, \mathbf{y}) \right]^2 \right], \\ & \text{with } g(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\psi(\mathbf{x})} q(\mathbf{y}|\mathbf{x}) e^{\psi(\mathbf{x},\mathbf{y})}, \text{ and } \int g(\mathbf{y}|\mathbf{x}) d\mathbf{y} = 1. \end{aligned} \tag{7.17}$$

Since $g(\mathbf{y}|\mathbf{x})$ is a probability density function, we can apply Jensen's inequality to the convex function $A \to A^2$ for the random variable $A := h(\mathbf{x}, \mathbf{y})$ to obtain

$$\begin{aligned} & \mathbb{E}_{g(\mathbf{y}|\mathbf{x})} \left[h(\mathbf{x}, \mathbf{y})^2 \right] - \left(\mathbb{E}_{g(\mathbf{y}|\mathbf{x})} \left[h(\mathbf{x}, \mathbf{y}) \right] \right)^2 \geq 0 \\ & \Rightarrow \delta^2 L[\psi][h]_{\text{asympt}} = \int p(\mathbf{x}) \left[\mathbb{E}_{g(\mathbf{y}|\mathbf{x})} \left[h(\mathbf{x}, \mathbf{y})^2 \right] - \left(\mathbb{E}_{g(\mathbf{y}|\mathbf{x})} \left[h(\mathbf{x}, \mathbf{y}) \right] \right)^2 \right] \geq 0. \end{aligned} \tag{7.18}$$

and it follows that the InfoNCE loss is convex in ψ.

To prove uniqueness of the minimum: Note that since the mapping $A \to A^2$ is not affine, a necessary and sufficient condition for equality is A to be constant, which holds iff $h(\mathbf{x}, \mathbf{y}) = h(\mathbf{x}, \mathbf{y}') := h(\mathbf{x})$ for all $\mathbf{y}$. The objective is hence strictly convex for all variations involving a variation in $\mathbf{y}$, and the second derivative vanishes for variations that only depend on $\mathbf{x}$. Variations that only depend on $\mathbf{x}$ are represented by the function C which appeared in the set of minimizers $\mathcal{M}$. □

Note that the difference between the minimizer of the NCE loss and the InfoNCE loss is the additional constant function C depending on the reference sample, which makes the loss function more flexible. Another common formulation of the InfoNCE

minimizer in the literature is given as $\psi(\mathbf{x}, \mathbf{y}) = \log p(\mathbf{x}, \mathbf{y})/(p_{\mathcal{D}}(\mathbf{x})p_{\mathcal{D}}(\mathbf{y}))$ which is a special case of our more general solution if $C(\mathbf{x}) = -\log p_{\mathcal{D}}(\mathbf{x})$, the negative distribution is chosen to be the marginal, $q = p_{\mathcal{D}}$, and the learning setup is symmetric.

Let us also confirm that our interchangeable use of the latents $(\mathbf{u}, \mathbf{v})$ and signal variables $(\mathbf{x}, \mathbf{y})$ is formally correct; due to the transformation theorem for any distribution $p_{\mathbf{u}}(\mathbf{u}) = p_{\mathbf{x}}(\mathbf{g}(\mathbf{u})) \det \mathbf{J}_{\mathbf{g}}(\mathbf{u})$ and respectively for $\mathbf{v}, \mathbf{g}', \mathbf{y}$. At the minimizer, we then arrive at

$$\log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} + C(\mathbf{x}) = \log \frac{p(\mathbf{y}|\mathbf{x})p_{\mathcal{D}}(\mathbf{x})}{q(\mathbf{y}|\mathbf{x})p_{\mathcal{D}}(\mathbf{x})} + C(\mathbf{x}) \tag{7.19}$$

$$= \log \frac{p(\mathbf{v}|\mathbf{u})p_{\mathcal{D}}(\mathbf{u}) \det \mathbf{J}_{\mathbf{g}}(\mathbf{u}) \det \mathbf{J}_{\mathbf{g}'}(\mathbf{v})}{q(\mathbf{v}|\mathbf{u})p_{\mathcal{D}}(\mathbf{u}) \det \mathbf{J}_{\mathbf{g}}(\mathbf{u}) \det \mathbf{J}_{\mathbf{g}'}(\mathbf{v})} + C'(\mathbf{u}) \tag{7.20}$$

$$= \log \frac{p(\mathbf{v}|\mathbf{u})}{q(\mathbf{v}|\mathbf{u})} + C'(\mathbf{u}), \quad \text{with } C(\mathbf{g}(\mathbf{u})) = C'(\mathbf{u}), \tag{7.21}$$

i.e., the minimizer can be equivalently written in terms of the latents and the signal variables.

Minimizers of the InfoNCE loss become bijective

A property that allows us to weaken some of the conditions given by Hyvärinen et al. (2019b), Zimmermann et al. (2021a) and Roeder et al. (2020a) is the observation that the composition of data generating process and feature encoder becomes bijective for the optimal value of the generalized InfoNCE objective. We introduce the following:

Definition 5 (Diversity condition for bijectivity)**.** *The sampling process composed of distributions p and q is sufficiently diverse if their log-likelihoods satisfy*

$$\operatorname{rank}\left(\left[\frac{\partial^2 \log p(\mathbf{v}|\mathbf{u})}{\partial u_i \partial v_j}\right]_{i\in[d],j\in[d]} - \left[\frac{\partial^2 \log q(\mathbf{v}|\mathbf{u})}{\partial u_i \partial v_j}\right]_{i\in[d],j\in[d]}\right) = d, \tag{7.22}$$

for all $\mathbf{u}$ in the support of the marginal distribution $p_{\mathcal{D}}$ and $d = d'$.

Def. 5 is a mild condition on the distributions p and q: Intuitively, the condition requires that for all samples $\mathbf{u}$, we can sample positive samples $\mathbf{v}$ that sufficiently vary in all d latent directions, which would be independent from the samples given by the negative distribution q. Suppose q is chosen to be uniform; then the condition is fulfilled for common choices like a Normal distribution with $\log p(\mathbf{v}|\mathbf{u}) = Z(\mathbf{u}) - (\mathbf{u} - \mathbf{v})^\top \mathbf{\Sigma}(\mathbf{u} - \mathbf{v})$ (where $\operatorname{rank}(-\mathbf{\Sigma}) = d$) or a von Mises-Fisher distribution with $\log p(\mathbf{v}|\mathbf{u}) = Z(\mathbf{u}) + \kappa \mathbf{u}^\top \mathbf{v}$ (where $\operatorname{rank} \kappa \mathbf{I} = d$).

We make two additional observations: Firstly, for simple distributions q that do not depend on $\mathbf{u}$, the diversity assumption only affects the positive distribution p as the second term vanishes. Secondly, if p and q are selected to train the network to become invariant to one factor v_i with $p(\mathbf{v}|\mathbf{u}) = p(v_i)p(\mathbf{v}_i|\mathbf{u})$ and $q(\mathbf{v}|\mathbf{u}) = p(v_i)q(\mathbf{v}_i|\mathbf{u})$,

the distributions $p(v_i)$ will cancel out in the condition, and reduce the rank by one dimension (which is as intended, as the factor should be discarded during training).

From this diversity condition, we can derive bijectivity of the composition $\mathbf{h} = \mathbf{f} \circ \mathbf{g}$ of the data generating process and feature encoder:

Proposition 3. *Assume that:*

1. *ψ with $\psi(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v}))$ is a minimizer of the InfoNCE objective (Def. 3) in a learning setup as outlined in Def 2.*

2. *The distributions p and q satisfy the diversity condition for bijectivity (Def. 5).*

Then $\mathbf{h}$ and $\mathbf{h}'$ are bijective on the support of $p_{\mathcal{D}}$.

Proof. By Proposition 2, the minimizer of the InfoNCE loss on the support of $p_{\mathcal{D}}$ is

$$\psi(\mathbf{u}, \mathbf{v}) = \log \frac{p(\mathbf{v}|\mathbf{u})}{q(\mathbf{v}|\mathbf{u})} + C(\mathbf{u}) \tag{7.23}$$

For ψ, we compute the second derivatives and arrange them in matrix form as

$$\left[\frac{\partial^2 \psi(\mathbf{u}, \mathbf{v})}{\partial u_i \partial v_j}\right]_{i \in [d], j \in [d]} = \mathbf{J}^\top(\mathbf{u}) \mathbf{P}(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v})) \mathbf{J}'(\mathbf{v}) \tag{7.24}$$

where we used the shorthand $\mathbf{P}(\mathbf{a}, \mathbf{b})_{ij} := \partial^2 \phi(\mathbf{a}, \mathbf{b}) / \partial \mathbf{a}_i \mathbf{b}_j$. $\mathbf{J}$ is the Jacobi matrix of $\mathbf{h}$, and $\mathbf{J}'$ is the Jacobi matrix of $\mathbf{h}'$. For the right hand side of the previous equation, we note that

$$\begin{aligned} &\operatorname{rank}(\mathbf{J}^\top(\mathbf{u}) \mathbf{P}(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v})) \mathbf{J}'(\mathbf{v})) \\ \leq &\min\{\operatorname{rank} \mathbf{J}(\mathbf{u}), \operatorname{rank} \mathbf{J}'(\mathbf{v}), \operatorname{rank} \mathbf{P}(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v}))\}, \end{aligned} \tag{7.25}$$

and the rank of the left hand side is given by inserting Eq. 7.23 into the diversity assumption (2):

$$\begin{aligned} &\operatorname{rank}\left[\frac{\partial^2 \psi(\mathbf{u}, \mathbf{v})}{\partial u_i \partial v_j}\right]_{i \in [d], j \in [d]} \\ = &\operatorname{rank}\left(\left[\frac{\partial^2 (\log p(\mathbf{v}|\mathbf{u}) - \log q(\mathbf{v}|\mathbf{u}) + C(\mathbf{u}))}{\partial u_i \partial v_j}\right]_{i \in [d], j \in [d]}\right) = d. \end{aligned}$$

Combining both results gives

$$\begin{aligned} &\operatorname{rank}(\mathbf{J}^\top(\mathbf{u}) \mathbf{P}(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v})) \mathbf{J}'(\mathbf{v})) = d \\ \leq &\min\{\operatorname{rank} \mathbf{J}(\mathbf{u}), \operatorname{rank} \mathbf{J}'(\mathbf{v}), \operatorname{rank} \mathbf{P}(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v}))\}, \end{aligned} \tag{7.26}$$

and implies

$$\operatorname{rank} \mathbf{J}(\mathbf{v}) = \operatorname{rank} \mathbf{J}'(\mathbf{u}) = \operatorname{rank} \mathbf{P}(\mathbf{h}(\mathbf{u}), \mathbf{h}'(\mathbf{v})) = d. \tag{7.27}$$

Then, both Jacobi matrices have full rank on the support of $p_{\mathcal{D}}$, hence $\mathbf{h}$ and $\mathbf{h}'$ are bijective, concluding the proof. □

Notably, this result is independent of the particular choice of the (potentially learnable) similarity measure ϕ. The similarity measure is implicitly constrained by the requirement that ψ needs to match the log-likelihood ratio of p and q up to a constant.

CEBRA models are consistent

We proceed by showing that CEBRA models are *consistent* under weak assumptions on the data distribution. Consistency entails that the embedding spaces of two different models can be mapped onto each other by some known transformation. In this subsection, we consider the class of *linear* transformations and in the following subsection we will discuss alternative transformations. We denote two independently trained CEBRA models as $\{\mathbf{f}, \mathbf{f}'\}$ and $\{\tilde{\mathbf{f}}, \tilde{\mathbf{f}}'\}$, and make statements about when linear transformations exist such that $\mathbf{f} = \mathbf{L}\tilde{\mathbf{f}}$ and $\mathbf{f}' = \mathbf{M}\tilde{\mathbf{f}}'$ for two full rank matrices $\mathbf{L}$ and $\mathbf{M}$.

We begin by recalling the Canonical Discriminative Form and Diversity Condition in Roeder et al. (2020a), adapted to our notation:

Definition A (Canonical Discriminative Form, Roeder et al. (2020a))**.** *Given a data distribution $p_{\mathcal{D}}(\mathbf{x}, \mathbf{y}, S)$ with random variables $\mathbf{x}$ and $\mathbf{y}$ and a set S containing the possible values of $\mathbf{y}$ given $\mathbf{x}$,*

$$p_{\mathcal{D}}(\mathbf{y}|\mathbf{x}, S) > 0 \iff \mathbf{y} \in S, \tag{7.28}$$

a generalized discriminative model family may be defined by its parameterization of the probability of the target variable $\mathbf{y}$ conditioned on an observed variable $\mathbf{x}$ and the set S that contains not only the true target label $\mathbf{y}$, but also a collection of distractors $\mathbf{y}'$:

$$p_{\mathbf{f},\mathbf{f}'}(\mathbf{y}|\mathbf{x}, S) = \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{f}'(\mathbf{y}))}{\sum_{\mathbf{y}' \in S} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{f}'(\mathbf{y}'))}. \tag{7.29}$$

Note that the feature extractors $\mathbf{f}$ and $\mathbf{f}'$ could be two separate networks, as we already discussed in Suppl. Note 1. Roeder et al. (2020a) consider $\mathbf{f}$ to be a "data encoder" and $\mathbf{f}'$ to be a "context encoder". This is in contrast to the setup by Hyvärinen et al. (2019b) which we will revisit in the context of recovering the data generating factors, where $\mathbf{f}(\mathbf{x})$ would play the role of an auxiliary variable while $\mathbf{f}'$ is the feature encoder later used in downstream tasks and analysis.

In CEBRA, the choice and role of both functions can be configured, which is why we will discuss theoretical guarantees for both use cases. We proceed by re-stating two diversity conditions needed for $\mathbf{f}$ or $\mathbf{f}'$ to become consistent:

Definition B (Diversity conditions for consistency, Roeder et al. (2020a))**.** *Let $Z(\mathbf{x}, S) := -\log \sum_{\mathbf{y}' \in S} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{f}'(\mathbf{y}'))$. Assume that for the encoders $(\mathbf{f}, \mathbf{f}', \tilde{\mathbf{f}}, \tilde{\mathbf{f}}')$ for which it holds that $p_{\mathbf{f},\mathbf{f}'} = p_{\tilde{\mathbf{f}},\tilde{\mathbf{f}}'}$*

1. *for any given* $\mathbf{y}$*, there exist* $M+1$ *tuples* $\{(\mathbf{x}^{(i)}, S^{(i)})\}_{i=1}^{M+1}$*, such that* $p_{\mathcal{D}}(\mathbf{x}^{(i)}, \mathbf{y}, S^{(i)}) > 0$*, and such that the* $((M+1)\times(M+1))$ *matrices* $\mathbf{M}$ *and* $\tilde{\mathbf{M}}$ *are invertible, where* $\mathbf{M}$ *consists of columns* $[-Z(\mathbf{x}^{(i)}, S^{(i)}); \mathbf{f}(\mathbf{x}^{(i)})]$*, and* $\tilde{\mathbf{M}}$ *consists of columns* $[-Z(\mathbf{x}^{(i)}, S^{(i)}); \tilde{\mathbf{f}}(\mathbf{x}^{(i)})]$*,*

2. *for any given* $\mathbf{x}$*, by repeated sampling* $S \sim p_D(S|\mathbf{x})$ *and picking two points* $\mathbf{y}_A, \mathbf{y}_B \in S$*, we can construct a set of* M *distinct tuples* $(\mathbf{y}_A^{(i)}, \mathbf{y}_B^{(i)})_{i=1}^{M}$ *such that the matrices* $\mathbf{L}$ *and* $\tilde{\mathbf{L}}$ *are invertible, where* $\mathbf{L}$ *consists of columns* $(\mathbf{f}'(\mathbf{y}_A(i)) - \mathbf{f}'(\mathbf{y}_B(i)))$*, and* $\tilde{\mathbf{L}}$ *consists of columns* $(\tilde{\mathbf{f}}'(\mathbf{y}_A(i)) - \tilde{\mathbf{f}}'(\mathbf{y}_B(i))$*,* $i \in 1, \dots, M$*.*

With the diversity condition in place, we recall Theorem 1 from Roeder et al. (2020a), adapted to our notation:

Theorem A (Roeder et al. (2020a))**.** *Under the diversity condition (Def. B), models following the canonical discriminative form (Def. A) are linearly identifiable. That is, for any encoders* $\{\mathbf{f}, \mathbf{f}'\}$*, and* $\{\tilde{\mathbf{f}}, \tilde{\mathbf{f}}'\}$ *it holds that*

$$p_{\mathbf{f},\mathbf{f}'} = p_{\tilde{\mathbf{f}},\tilde{\mathbf{f}}'} \quad \Longrightarrow \quad \mathbf{f}(\mathbf{x}) = \mathbf{L}\tilde{\mathbf{f}}(\mathbf{x}), \quad \mathbf{f}'(\mathbf{y}) = \mathbf{L}\tilde{\mathbf{f}}'(\mathbf{y}) \tag{7.30}$$

for all samples ($\mathbf{x}$, $\mathbf{y}$) *in the support of the data distribution.*

Proof. See Theorem 1, Roeder et al. (2020a), where we replaced the equivalence condition on the right hand side by inserting its definition for clarity. □

We can leverage this result as CEBRA falls into the class of models models described in Definition A:

Proposition 4 (CEBRA models are consistent.)**.** *Assume that two CEBRA models are trained on data from the same latent data distribution, and denote the feature encoders of the trained models as* $\mathbf{f}, \mathbf{f}'$ *and* $\tilde{\mathbf{f}}, \tilde{\mathbf{f}}'$*. Further assume that for both models, the similarity measure* ϕ *is the dot-product similarity and* ψ *minimizes the generalized InfoNCE loss. Finally assume that the diversity condition (Def. B) holds. Then the feature encoders* $\mathbf{f}$, $\mathbf{f}'$ *are consistent up to a linear transformation, and* $\mathbf{f}(\mathbf{x}) = \mathbf{L}\tilde{\mathbf{f}}(\mathbf{x})$, $\mathbf{f}'(\mathbf{y}) = \mathbf{L}'\tilde{\mathbf{f}}'(\mathbf{y})$*, for linear transformations* $\mathbf{L}$, $\mathbf{L}'$ *for any pair of points* $\mathbf{x}, \mathbf{y}$ *in the support of the data distribution.*

Proof. The generalized InfoNCE objective with limited samples (Def. 4) can be written as

$$\mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p(\mathbf{x}),\ \mathbf{y}_+\sim p(\mathbf{y}|\mathbf{x}) \\ \mathbf{y}_1,\dots,\mathbf{y}_n\sim q(\mathbf{y}|\mathbf{x})}} [\log p(\mathbf{y}^+|\mathbf{x}, S)], \tag{7.31}$$

with

$$\begin{aligned} \log p(\mathbf{y}^+|\mathbf{x}, S) &= -\log \frac{\exp \psi(\mathbf{x}, \mathbf{y}_+)}{\sum_{i=1}^{n} \exp \psi(\mathbf{x}, \mathbf{y}_i)} \\ &= -\log \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{f}'(\mathbf{y}_+)/\tau)}{\sum_{i=1}^{n} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{f}'(\mathbf{y}_i)/\tau)} \end{aligned} \tag{7.32}$$

which matches the canonical discriminative form (Def. A) with encoders $\mathbf{f}(\cdot)$ and $\mathbf{f}'(\cdot)/\tau$. The composition of the set $S := \{\mathbf{y}_i\}_{i=1}^{N}$ is given by the distribution $p_{\mathcal{D}}(\mathbf{x})q(\mathbf{y}|\mathbf{x})$ and S

fulfills Def. B by assumption. At the minimizer, the values of the loss functions match, from which it follows that $p_{\mathbf{f},\mathbf{f}'} = p_{\tilde{\mathbf{f}},\tilde{\mathbf{f}}'}$. Hence, we apply Theorem A to find that $\mathbf{f}$ and $\mathbf{f}'$ are consistent up to a linear transform, concluding the proof. □

It is worth noting that this result also *holds for datasets with limited samples*, i.e., the objective in Def. 4, as long as the dataset fulfills the diversity condition (Def. B). Checking the diversity condition as well as matching distributions for the two CEBRA models is possible in practice using only the dataset and the trained model.

Both diversity conditions from Roeder et al. (2020a) depend on the variability of the ground truth latent distribution (and the presence of this variability after mapping the latents to signal space), and on the properties of the encoders $\mathbf{f}$ and $\mathbf{f}'$. While Roeder et al. (2020a) already discuss that in practice, a randomized neural network will fulfill the diversity conditions, with our diversity criterion for bijectivity (Def. 5) and the bijectivity of $\mathbf{f}$ and $\mathbf{f}'$ that follows, we can strengthen this argument and ensure that both conditions hold upon convergence for minimizers of the generalized InfoNCE objective:

Proposition 5. *Assume that the encoders* $(\mathbf{f}, \mathbf{f}', \tilde{\mathbf{f}}, \tilde{\mathbf{f}}')$ *and hence also the compositions of data generator and encoders* $(\mathbf{h}, \mathbf{h}', \tilde{\mathbf{h}}, \tilde{\mathbf{h}}')$ *minimize the InfoNCE objective. Assume that upon convergence, the diversity condition for bijectivity (Def. 5) holds. Then,* $\mathbf{h}(\mathbf{u}) = \mathbf{h}(\mathbf{u})$ *and* $\mathbf{h}'(\mathbf{v}) = \mathbf{B}\tilde{\mathbf{h}}'(\mathbf{v})$ *for all latents* $\mathbf{u}, \mathbf{v}$ *in the support of the data distribution for two full-rank matrices ,* $\mathbf{B}$*.*

Proof. Both models share the minimizer

$$\mathbf{h}(\mathbf{u})^\top \mathbf{h}'(\mathbf{v}) = \tilde{\mathbf{h}}(\mathbf{u})^\top \tilde{\mathbf{h}}'(\mathbf{v}) = \log \frac{p(\mathbf{v}|\mathbf{u})}{q(\mathbf{v}|\mathbf{u})} + C(\mathbf{u}) \tag{7.33}$$

and we have

$$\mathbf{h}(\mathbf{u})^\top \mathbf{h}'(\mathbf{v}) = \tilde{\mathbf{h}}(\mathbf{u})^\top \tilde{\mathbf{h}}'(\mathbf{v}) \tag{7.34}$$

taking derivatives with respect to $\mathbf{v}$, and then to $\mathbf{u}$, we arrive at

$$\mathbf{J}(\mathbf{u})^\top \mathbf{J}'(\mathbf{v}) = \tilde{\mathbf{J}}(\mathbf{u})^\top \tilde{\mathbf{J}}'(\mathbf{v}) \tag{7.35}$$

where all Jacobian matrices have full rank due to Prop. 3. We can hence derive

$$\mathbf{J}'(\mathbf{v}) = (\mathbf{J}(\mathbf{u})^{-\top} \tilde{\mathbf{J}}(\mathbf{u})^\top) \tilde{\mathbf{J}}'(\mathbf{v}) \quad \mathbf{J}(\mathbf{u})^\top = \tilde{\mathbf{J}}(\mathbf{u})^\top (\tilde{\mathbf{J}}'(\mathbf{v}) \mathbf{J}'(\mathbf{v})^{-1}) \tag{7.36}$$

$$\mathbf{J}'(\mathbf{v}) = (\mathbf{u}) \tilde{\mathbf{J}}'(\mathbf{v}) \quad \mathbf{J}(\mathbf{u})^\top = \tilde{\mathbf{J}}(\mathbf{u})^\top \mathbf{B}(\mathbf{v}) \tag{7.37}$$

for some full rank matrices $(\mathbf{u})$ and $\mathbf{B}(\mathbf{v})$. Because the left hand side do not depend on the argument of and $\mathbf{B}$, both matrices need to be constant, leaving

$$\mathbf{J}'(\mathbf{v}) = \tilde{\mathbf{J}}'(\mathbf{v}) \quad \mathbf{J}(\mathbf{u})^\top = \tilde{\mathbf{J}}(\mathbf{u})^\top \mathbf{B} \tag{7.38}$$

from which it follows that

$$\mathbf{h}(\mathbf{u}) = \mathbf{B}^{-1}\tilde{\mathbf{h}}(\mathbf{u}) \quad \mathbf{h}'(\mathbf{v}) = \tilde{\mathbf{h}}'(\mathbf{v}). \tag{7.39}$$

concluding the proof. □

Note that the previous proposition applies even when the data generating functions (i.e., the datasets) between the model run differ, and merely the latent distributions match. In this case, the same latent $\mathbf{u}_0$ can be mapped to different points $\mathbf{x}_0$ and $\tilde{\mathbf{x}}_0$ and the resulting embedding points would still satisfy $\mathbf{f}(\mathbf{u}_0) = \tilde{\mathbf{f}}(\mathbf{u}_0)$. For this reason, the proposition is written w.r.t. the composition $\tilde{\mathbf{h}}$ of data generating process and encoder. If the data generating processes match, it is clear that Eq. 7.39 can be equivalently written as $\mathbf{f}(\mathbf{x}) = \mathbf{B}^{-1}\tilde{\mathbf{f}}(\mathbf{x})$, $\mathbf{f}'(\mathbf{y}) = \tilde{\mathbf{f}}'(\mathbf{y})$, which matches the statement by Roeder et al. (2020a) (but for our modified diversity condition).

For symmetric encoders, we can give the following result:

Proposition 6. *Assume that the encoders $\mathbf{f} = \mathbf{f}', \tilde{\mathbf{f}} = \tilde{\mathbf{f}}'$ are shared, and $-\phi$ is a norm, $\phi(\mathbf{a}, \mathbf{b}) = -\|\mathbf{a} - \mathbf{b}\|$. Assume that the model minimizes the InfoNCE loss and assume that the co-domain of $\mathbf{f}$, $\mathbf{f}'$ is a normed space over $\mathbb{R}^d$. Then $\mathbf{h} = \mathbf{L}\tilde{\mathbf{h}}$.*

Proof. We write the data in terms of the underlying latents $\mathbf{x} = \mathbf{g}(\mathbf{u})$ and $\mathbf{y} = \mathbf{g}'(\mathbf{v})$. At the minimizer of the InfoNCE loss it then holds that

$$\|\mathbf{h}(\mathbf{u}) - \mathbf{h}(\mathbf{v})\| + C(\mathbf{u}) = \|\tilde{\mathbf{h}}(\mathbf{u}) - \tilde{\mathbf{h}}(\mathbf{v})\| + \tilde{C}(\mathbf{u}). \tag{7.40}$$

for all points in the dataset. Inserting $\mathbf{v} = \mathbf{u}$ gives $C(\mathbf{u}) = \tilde{C}(\mathbf{u})$. Because $\mathbf{h}, \tilde{\mathbf{h}}$ bijective, we define points $\mathbf{a} = \tilde{\mathbf{h}}(\mathbf{u})$ and $\mathbf{b} = \tilde{\mathbf{h}}(\mathbf{v})$, and it holds

$$\|\mathbf{h}(\tilde{\mathbf{h}}^{-1}(\mathbf{a})) - \mathbf{h}(\tilde{\mathbf{h}}^{-1}(\mathbf{b}))\| = \|\mathbf{a} - \mathbf{b}\|. \tag{7.41}$$

Due to the Mazur–Ulam theorem, the map $\mathbf{h} \circ \tilde{\mathbf{h}}^{-1}$ is then affine, concluding the proof. □

Note that all results in this section are also independent of the mixing functions $\mathbf{g}$ and $\mathbf{g}'$. This means that *consistency can be guaranteed irrespective of the exact data modality and generative process*, as long as the *underlying* latent distribution matches. This is given in well-controlled experiments (as one assumes when e.g., repeating experiments).

CEBRA models recover the ground-truth latents

In contrast to the identifiability results in the previous section (which made a connection between two models trained on the same or similar data distribution), in this section we will make a connection between the *ground truth model* and the model trained on the data. To make this connection, assumptions are needed about the ground truth model.

As introduced in Def. 2, we will denote the ground truth model(s) as $\mathbf{g}$ and $\mathbf{g}'$, and data from these models is then encoded with $\mathbf{f}$ and $\mathbf{f}'$, respectively. Our goal is to understand properties of their composition $\mathbf{h} = \mathbf{g} \circ \mathbf{f}$ and $\mathbf{h}' = \mathbf{g}' \circ \mathbf{f}'$, and when this composition reduces to an affine or linear transformation. Based on the property of the model (especially the similarity measure), other guarantees are possible. Towards the end of this subsection we will discuss CEBRA model setups where we obtain guarantees up to permutations and sign-flips and point-wise non-linear transformations by leveraging existing contrastive learning theory (Hyvärinen et al., 2019b; Zimmermann et al., 2021a). We base our theory on the identifiability proofs of contrastive learning given by Hyvärinen et al. (2019b) and Zimmermann et al. (2021a), and give new proofs to complete the theory for the most important usage modes in CEBRA. Note that the theory also extends to settings not explicitly demonstrated in this paper, but integrated into the CEBRA software package (e.g., trainable similarity functions ϕ).

One key property of CEBRA is its distinction of discovery-driven and hypothesis-driven training (or hybrid training, which is a combination of both where the feature encoders need to minimize both the time-contrastive and behavior-contrastive objectives). For time- and behavior-contrastive learning, similar theory applies. A key difference is that in time-contrastive (discovery-driven) learning, an underlying distribution of the latents is inherent to the dataset and "given" by the temporal variation in the data. If it is desirable to recover the ground truth latents, the similarity measure needs to be suitable to allow full InfoNCE minimization, e.g., by model selection on basis of the InfoNCE or goodness of fit metric. Note that a cosine similarity measure is already quite flexible for a wide range of distributions and a good default, and also note that InfoNCE minimization is known to be empirically robust to minor violations between the model assumptions and ground-truth conditional distribution (Zimmermann et al., 2021a). For hypothesis-driven learning, CEBRA will always be able to find a suitable learning setup that recovers the auxiliary variables: The positive and negative sample distributions can be chosen and selected such that the propositions in this section will hold. Even if empirical distributions are used (e.g., the "time delta" distribution outlined in the Methods), it is possible to check that this distribution matches the similarity measure of the model.

Because we make statements about the relationship between continual and discrete context variables ($\mathbf{c}_t$ and k_t) and the signal space ($\mathbf{ss}_t$) for each time-point t, we will again use this notation from the Methods; we use the variable names introduced in Def. 2 to denote the underlying ground-truth latents, and the samples fed to the model (which can, but do not necessarily need to, equal to the signal). We start our discussion with discovery-driven training of CEBRA using time information:

Proposition 7 (Discovery-driven CEBRA)**.** *Assume the learning setup in Def. 2, and that the ground-truth latents $\mathbf{u}_1, \ldots, \mathbf{u}_T$ for each time point follow a uniform marginal distribution and the change between subsequent time steps is given by the conditional distribution of the*

form

$$p(\mathbf{u}_{t+\Delta t}|\mathbf{u}_t) = \frac{1}{Z(\mathbf{u}_t)} \exp \delta(\mathbf{u}_{t+\Delta t}, \mathbf{u}_t) \tag{7.42}$$

where δ is either a (scaled) dot product (and $\mathbf{u}_t \in \mathcal{S}^{n-1} \subset \mathbb{R}^d$ lies on the $(n-1)$-sphere $\mathcal{S}^{n-1}$) or an arbitrary semi-metric (and $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^d$ lies in a convex body $\mathcal{U}$). Assume that the data generating process $\mathbf{g}$ with $\mathbf{ss}_t = \mathbf{g}(\mathbf{u}_t)$ is injective. Assume we train a symmetric CEBRA model with encoder $\mathbf{f} = \mathbf{f}'$ and the similarity measure including a fixed temperature $\tau > 0$ is set to or sufficiently flexible such that $\phi = \delta$ for all arguments. Then $\mathbf{h} = \mathbf{h}' = \mathbf{g} \circ \mathbf{f}$ is affine.

Proof. For δ being the dot product, the result follows from the proof of Theorem 2 in Zimmermann et al. (2021a). For δ being a semi-metric, the result follows from the proof of Theorem 5 in Zimmermann et al. (2021a). □

We will next consider the hypothesis-driven mode in CEBRA. Here, we either choose a parametric or non-parametric positive distribution p to shape the embedding space. Our goal is find an embedding space reflecting the auxiliary variable, in case there is a meaningful relationship between the signal and this variable.

Naturally, the actual signal will depend on additional latents that we do not record as auxiliary information. The full data generating process can be written as

$$\mathbf{ss}_t = \mathbf{g}(\mathbf{c}_t, k_t, \mathbf{z}_t) \tag{7.43}$$

where $\mathbf{z}_t$ are additional latent sources not observed during training. Since $\mathbf{g}$ is an injective function, it follows that $\mathbf{ss}_t = \mathbf{ss}_{t'}$ implies $\mathbf{c}_t = \mathbf{c}_{t'}$, $k_t = k_{t'}$, $\mathbf{z}_t = \mathbf{z}_{t'}$. Applying hypothesis-driven learning in this setup will force an embedding space representing $\mathbf{c}$ and k, but not $\mathbf{z}$. We can denote the set $G(\mathbf{c}, k) := \{\mathbf{z} \in Z : \mathbf{g}(\mathbf{c}, k, \mathbf{z})\}$ which contains all points in signal space corresponding to a particular set of auxiliary variables $\mathbf{c}, k$. Because $\mathbf{g}$ is injective, there exists a function $\tilde{\mathbf{g}}$, which will retrieve the original auxiliary variables: $\tilde{\mathbf{g}}(\mathbf{g}(\mathbf{c}, k, \mathbf{z})) = (\mathbf{c}, k)^\top$.

Proposition 8 (Hypothesis-driven CEBRA)**.** *Assume a partially observable data generating process, where*

$$\mathbf{y} = \mathbf{g}'(\mathbf{c}, \mathbf{v}) \tag{7.44}$$

with $\mathbf{g}'$ injective and with $\mathbf{c}$ as an observable context variable, and $\mathbf{v}$ as a latent. As in Prop. 3 assume that $\mathbf{c}$ lies on a hypersphere if ϕ is the dot-product similarity, and on a convex body if $-\phi$ is a semi-metric. Then the minimizer of the InfoNCE loss trained with a distribution $p(\mathbf{c}|\tilde{\mathbf{c}}) = \exp(\phi(\mathbf{c}, \tilde{\mathbf{c}}))/Z(\tilde{\mathbf{c}})$ and a uniform marginal and negative distribution $q = p_{\mathcal{D}}$ will yield a $\mathbf{h}$ that recovers $\mathbf{c}$ up to an affine transformation.

Proof. $\mathbf{h}$ is the composition $\mathbf{g} \circ \mathbf{f}$. Per assumption, the similarity function ϕ is sufficiently flexible such that

$$\phi(\mathbf{h}(\mathbf{c}, \mathbf{z}), \mathbf{h}(\tilde{\mathbf{c}}, \mathbf{z})) = \log p(\mathbf{c}, \tilde{\mathbf{c}}) + C(\tilde{\mathbf{c}}) \quad \forall \mathbf{z}. \tag{7.45}$$

Because $\mathbf{g}$ is injective, $\mathbf{g}(\mathbf{c}, \mathbf{z}) = \mathbf{g}(\mathbf{c}', \mathbf{z}')$ implies that $\mathbf{c} = \mathbf{c}'$ and $\mathbf{z} = \mathbf{z}'$ and there exists a function $\tilde{\mathbf{g}}$ such that $\tilde{\mathbf{g}}(\mathbf{g}(\mathbf{c}, \mathbf{z})) = \mathbf{c}$ for all $\mathbf{z}$. It remains to show that $\mathbf{f} = \mathbf{L}\tilde{\mathbf{g}}$ at the minimizer for a full-rank matrix $\mathbf{L}$. By comparing arguments on the left hand side and right and side, and inserting the form of p, solutions of Eq. 7.45 need to simplify to

$$\phi(\mathbf{h}(\mathbf{c}), \mathbf{h}(\tilde{\mathbf{c}})) = \phi(\mathbf{c}, \tilde{\mathbf{c}}) + C(\tilde{\mathbf{c}}) \tag{7.46}$$

By symmetry of ϕ, $C(\tilde{\mathbf{c}}) = 0$ vanishes. Then the result follows again from Theorem 2 in Zimmermann et al. (2021a) for ϕ being the dot-product similarity, and from Theorem 5 in Zimmermann et al. (2021a) for $-\phi$ being a semi-metric. □

With our results from Prop. 2 and Prop. 3, it is also possible for us to extend the results for distributions fulfilling the diversity condition for bijectivity (Def. 5). The following proposition enables a statement about distribution where the data generating process and/or feature encoder differs between the reference and positive/negative pairs. An example used in the main text is multi-session training, or training across data modalities. We first need to strengthen the assumption about the positive and negative distributions from Def. 5. Again assume that the latent dimensions $d = d'$ match.

Definition 6 (Strong assumption of diversity for bijectivity)**.** *The sampling process with distributions p and q satisfies*

$$\left[\frac{\partial^2 \log p(\mathbf{v}|\mathbf{u})}{\partial u_i \partial v_j}\right]_{i\in[d], j\in[d]} - \left[\frac{\partial^2 \log q(\mathbf{v}|\mathbf{u})}{\partial u_i \partial v_j}\right]_{i\in[d], j\in[d]} = \mathbf{L} \tag{7.47}$$

for some full-rank matrix $\mathbf{L} \in \mathbb{R}^{d\times d}$ for all $\mathbf{u} \in \mathbb{R}^d$ in the support of the marginal distribution $p_{\mathcal{D}}$.

Note that Def. 6 is a special case of the diversity condition for bijectivity in Def. 5. The stronger condition still covers important distributions like a Gaussian conditional, even with an additional mean and covariance $p(\mathbf{v}|\mathbf{u}) = \exp(-(\mathbf{u} - \mathbf{v} - \mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{u} - \mathbf{v} - \mu))$. Likewise, it would cover van Mises-Fisher (vMF) distributions, even with a rotated reference sample and $p(\mathbf{v}|\mathbf{u}) = \exp(\kappa \mathbf{u}^\top \mathbf{Q}\mathbf{v})$. We will cover these selected special cases using the stronger diversity assumption below.

Proposition 9. *Consider a data generating process with uniform marginal and positive/negative distributions satisfying Def. 6. Assume the data is generated by mixing functions $\mathbf{g} : \mathbb{R}^d \mapsto \mathbb{R}^D$ and $\mathbf{g}' \in \mathbb{R}^d \mapsto \mathbb{R}^{D'}$ and encoded into a shared E-dimensional embedding space with two separate encoders $\mathbf{f} : \mathbb{R}^D \mapsto \mathbb{R}^E$ and $\mathbf{f}' : \mathbb{R}^{D'} \mapsto \mathbb{R}^E$ and assume that ϕ is the dot-product. Then there are d dimensions in $\mathbf{h}$, and d dimensions in $\mathbf{h}'$ which represent the latents up to an affine transform.*

Proof. Let us again denote the compositions $\mathbf{h} = \mathbf{g} \circ \mathbf{f} : \mathbb{R}^d \mapsto \mathbb{R}^E$ and $\mathbf{h}' = \mathbf{g}' \circ \mathbf{f}' : \mathbb{R}^d \mapsto \mathbb{R}^E$ and let us denote the Jacobian matrices as $\mathbf{J} : \mathbb{R}^d \mapsto \mathbb{R}^{E\times d}$ and $\mathbf{J}' : \mathbb{R}^d \mapsto \mathbb{R}^{E\times d}$,

respectively. Without loss of generality (the indices can be arbitrarily permuted), let us split each $\mathbf{h}$ into two parts, with $\mathbf{h}_1 := [h_1, \ldots, h_d]^\top$ and $\mathbf{h}_2 := [h_{d+1}, \ldots, h_E]^\top$ (in case $E > d$), and respectively for $\mathbf{h}'$. At the minimizer of the InfoNCE loss, we get the condition

$$\mathbf{h}(\mathbf{u})^\top \mathbf{h}'(\mathbf{v})/\tau = \log \frac{p(\mathbf{v}|\mathbf{u})}{q(\mathbf{v}|\mathbf{u})} + C(\mathbf{u}). \tag{7.48}$$

We take the derivative w.r.t. $\mathbf{v}$ on both sides, with gives

$$\mathbf{J}'(\mathbf{v})^\top \mathbf{h}(\mathbf{u})/\tau = \frac{\partial}{\partial \mathbf{v}} \left[\log \frac{p(\mathbf{v}|\mathbf{u})}{q(\mathbf{v}|\mathbf{u})} \right], \tag{7.49}$$

with $\mathbf{J}'$ denoting the Jacobian matrix of $\mathbf{h}'$. We now take the derivative w.r.t. $\mathbf{u}$ on both sides, with gives

$$\mathbf{J}(\mathbf{u})^\top \mathbf{J}'(\mathbf{v})/\tau = \frac{\partial^2}{\partial \mathbf{u} \partial \mathbf{v}} \left[\log \frac{p(\mathbf{v}|\mathbf{u})}{q(\mathbf{v}|\mathbf{u})} \right]. \tag{7.50}$$

and by assumption we can insert Def. 6 and re-arrange to

$$\mathbf{J}(\mathbf{u})\mathbf{J}'(\mathbf{v})^\top = \tau \mathbf{L} \tag{7.51}$$

for some full-rank matrix $\mathbf{L} \in \mathbb{R}^{d \times d}$. Note that when $E > d$ this condition is under-constrained. Without loss of generality, let us split the Jacobian matrices into two parts:

$$\mathbf{J}(\mathbf{u}) := [\mathbf{J}_1(\mathbf{u}); \mathbf{0}] \quad \mathbf{J}'(\mathbf{v}) := [\mathbf{J}'_1(\mathbf{v}); \mathbf{J}'_2(\mathbf{v})] \tag{7.52}$$

where $\mathbf{J}_1 : \mathbb{R}^d \mapsto \mathbb{R}^{d \times d}$, $\mathbf{J}'_1 : \mathbb{R}^d \mapsto \mathbb{R}^{d \times d}$, $\mathbf{J}'_2 : \mathbb{R}^d \mapsto \mathbb{R}^{(E-d) \times d}$. This allows to re-write the condition as

$$\mathbf{J}_1(\mathbf{u})\mathbf{J}'_1(\mathbf{v})^\top = \tau \mathbf{L} \tag{7.53}$$

which is no longer underconstrained. This equation is valid for any $\mathbf{u}, \mathbf{v}$ in the support of the marginal and the positive/negative distribution, which is why the left hand side cannot depend on $\mathbf{u}$ and $\mathbf{v}$, leaving the final form of the Jacobians:

$$\mathbf{J}(\mathbf{u}) := [\mathbf{J}_1; \mathbf{0}] \quad \mathbf{J}'(\mathbf{v}) := [\mathbf{J}'_1; \mathbf{a}(\mathbf{v})], \tag{7.54}$$

where $\mathbf{a} : \mathbb{R}^d \mapsto \mathbb{R}^{(E-d) \times d}$ is an arbitrary function that ensures that Eq. 7.48 can hold. It follows that $\mathbf{h}_1$ and $\mathbf{h}'_1$ are affine transforms, $\mathbf{h}_2$ is a constant, and $\mathbf{h}'_2$ will be a potentially non-linear transform to match the InfoNCE minimizer, concluding the proof. □

We next discuss two special cases that are of interest to users of CEBRA and demonstrate that the previous result is more flexible than the results presented in Zimmermann et al. (2021a). Consider a setup where we have a constant "offset" between the latents vs. having a perfectly symmetric p. We are still able to recover the underlying latents for both vMF and Normal conditional distributions:

Corollary 1. *The aforementioned result holds for vMF distributions with a constant offset between* $\mathbf{u}$ *and* $\mathbf{v}$ *on the hypersphere, parameterized by a rotation matrix* $\mathbf{Q}$*, with* $\log p(\mathbf{v}|\mathbf{u}) = \kappa \mathbf{u}^\top \mathbf{Q}\mathbf{v}$*. Assume* $d = d' = k$ *and assume the domain and co-domain of* $\mathbf{h}$, $\mathbf{h}'$ *is the unit sphere.*

Proof. The distributions satisfies constancy of the second derivative, and the condition on the Jacobian matrices is

$$\mathbf{J}_1 \mathbf{J}_1'^\top = (\tau\kappa)\mathbf{Q} \tag{7.55}$$

and since all vectors $\mathbf{h}(\mathbf{a}) = \mathbf{J}^\top \mathbf{a}$ need to be normalized and $\mathbf{Q}$ is orthogonal, we can consider any column $\mathbf{q}_i$ and the corresponding vector $\mathbf{a}_i$ of $\mathbf{J}_1$

$$\mathbf{J}_1 \mathbf{a}_i = (\tau\kappa)\mathbf{q}_i \Rightarrow \|\mathbf{J}_1 \mathbf{a}_i\|^2 = (\tau\kappa)\|\mathbf{q}_i\|^2 \Rightarrow 1 = (\tau\kappa)1 \tag{7.56}$$

and it follows $\tau = 1/\kappa$. □

Corollary 2. *The aforementioned result holds for a Gaussian distribution with a constant offset between* $\mathbf{u}$ *and* $\mathbf{v}$ *parameterized, with* $\log p(\mathbf{v}|\mathbf{u}) = \|\mathbf{u} - \mathbf{v} - \mu\|^2$*. Assume* $d = d' = k$*.*

Proof. We rewrite the log-conditional as

$$\begin{aligned}
\log p(\mathbf{v}|\mathbf{u}) &= -\|(\mathbf{u} - \mathbf{v}) - \boldsymbol{\mu}\|^2 && (7.57)\\
&= -\|\mathbf{u} - \mathbf{v}\|^2 - \|\boldsymbol{\mu}\|^2 + 2(\mathbf{u} - \mathbf{v})^\top \boldsymbol{\mu} && (7.58)\\
&= -\|\mathbf{u}\|^2 - \|\mathbf{v}\|^2 - \|\boldsymbol{\mu}\|^2 + 2\mathbf{u}^\top \boldsymbol{\mu} + 2\mathbf{v}^\top \boldsymbol{\mu} + 2\mathbf{u}^\top \mathbf{v} && (7.59)\\
& && (7.60)
\end{aligned}$$

which gives, at the minimizer of the InfoNCE objective,

$$\mathbf{h}(\mathbf{u})^\top \mathbf{h}'(\mathbf{v})/\tau = 2\mathbf{u}^\top \mathbf{v} + (-\|\mathbf{v}\|^2 + 2\mathbf{v}^\top \boldsymbol{\mu}) + (-\|\mathbf{u}\|^2 + 2\mathbf{u}^\top \boldsymbol{\mu} + C(\mathbf{u})) \tag{7.61}$$

and we recover

$$\begin{aligned}
\mathbf{h}_1(\mathbf{u}) &= \mathbf{J}_1 \mathbf{u}, && (7.62)\\
\mathbf{h}_1'(\mathbf{v}) &= \mathbf{J}_1' \mathbf{v}, && (7.63)\\
C(\mathbf{u}) &= \|\mathbf{u}\|^2 - 2\mathbf{u}^\top \boldsymbol{\mu} && (7.64)\\
\mathbf{h}_2'(\mathbf{v}) &= -\|\mathbf{v}\|^2 + 2\mathbf{v}^\top \boldsymbol{\mu}. && (7.65)
\end{aligned}$$

□

While the above results extended the theory of Zimmermann et al. (2021a) to training setups within the CEBRA library and used in the main text (training with dot-product and negative mean squared error as the similarity measure and sampling procedure), we can also leverage previous results from Hyvärinen et al. (2019b) to further extend the families of possible distributions.

Firstly, for conditional exponential family distributions within an ICA framework, the dot-product similarity can be used in conjunction with non-symmetric encoders $\mathbf{f}$, $\mathbf{f}'$ (i.e., two separate networks) to recover the components up to a linear indeterminacy. We recall Definition 1 from Hyvärinen et al. (2019b):

Definition C (Conditionally exponential distributions (Def. 1 from Hyvärinen et al. (2019b)))**.** *A random variable (independent component) v_i is conditionally exponential of order k given random vector $\mathbf{x}$ if its conditional probability density function can be given in the form*

$$p(v_i|\mathbf{x}) = \frac{Q_i(v_i)}{Z_i(\mathbf{x})} \exp\left[\sum_{j=1}^{k} \tilde{q}_{ij}(v_i)\lambda_{ij}(\mathbf{x})\right] \tag{7.66}$$

almost everywhere in the support of $\mathbf{x}$, with $\tilde{q}_{ij}$, λ_{ij}, Q_i, and Z_i scalar-valued functions. The sufficient statistics $\tilde{q}_{ij}$ are assumed linearly independent (over j, for each fixed i).

We observe a similar functional form as used in Prop. 9. Values of $\tilde{q}_{ij}(v_i)$ would be represented by $\mathbf{f}'$, and values in $\lambda_{ij}(\mathbf{x})$ would be represented by $\mathbf{f}$. As in the previous proposition, more dimensions in $\mathbf{f}$, $\mathbf{f}'$ as latent variables are required ($E > d$) for representing conditional distributions of the conditional exponential form. In this setup, Theorem 3 from Hyvärinen et al. (2019b) implies in our context that the sufficient statistics of the latents can be recovered up to a linear transformation.

Secondly, for arbitrary conditional distributions, as long as variability conditions are satisfied within an ICA framework, contrastive learning can recover the underlying components up to a permutation and point-wise invertible transformation (Hyvärinen et al., 2019b). Using this mode applies when a similarity measure ϕ which gives

$$\psi(\mathbf{u}, \mathbf{v}) := \sum_{i=1}^{E} \phi_i(h_i'(\mathbf{v}), \mathbf{h}(\mathbf{u})) \tag{7.67}$$

or w.r.t. to the signal variables,

$$\psi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{E} \phi_i(f_i'(\mathbf{y}), \mathbf{f}(\mathbf{x})), \tag{7.68}$$

is used within the CEBRA framework. In this case, Theorem 1 in Hyvärinen et al. (2019b) applies and gives an identifiability guarantee of $\mathbf{h}$ to recover the ground truth latents $\mathbf{v}$ up to permutations and component-wise non-linear transformations.

8

Identifiable attribution maps using regularized contrastive learning

The following pages contain the postprint version the NeurIPS 2023 workshop paper

> Steffen Schneider, Rodrigo González Laiz, Markus Frey, and Mackenzie Weygandt Mathis. "Identifiable attribution maps using regularized contrastive learning." *NeurIPS 2023 workshop on self-supervised learning – theory and practice.*

An extended abstract version of the work was presented at COSYNE 2023, Montreal.

Author Contributions Conceptualization: StS, MWM; Methodology: StS, MWM, RG; Software: RG, StS; Theory: StS; Formal analysis: StS, RG, MF; Investigation: RG, StS, MF; Data Curation: StS, RG, MF; Writing-Original Draft: StS, MWM; Writing-Editing: all authors.

Summary

Gradient-based attribution methods aim to explain decisions of deep learning models, but so far lack identifiability guarantees. Here, we propose a method to generate attribution maps with identifiability guarantees by developing a regularized contrastive learning algorithm trained on time series data with continuous target labels. We show theoretically that our formulation of hybrid contrastive learning has favorable properties for identifying the Jacobian matrix of the data generating process, and is unable to overfit to random training distributions. Empirically, we demonstrate robust approximation of the ground-truth attribution map on synthetic data, and significant improvements across previous attribution methods based on feature ablation, Shapley values, and other gradient-based methods. Our work constitutes a first example of identifiable inference of attribution maps, and opens avenues for improving future attribution tools and better understanding neural dynamics and neural networks.

Introduction

Distilling knowledge from data is a core tenet of science. After pre-processing raw data, we want to abstract relationships in the experimentally observed data to observed variables. In the case of neuroscience this could be the raw neural signal and the behavior of the animal (Jazayeri & Ostojic, 2021; Urai et al., 2022b). Often times linear methods (such as GLMs (McCullagh & Nelder, 1972)) are used for interpretability, even though the underlying data did not necessarily arise from linear processes. Yet, non-linear methods are difficult to interpret (Breen et al., 2018; Samek et al., 2019).

In machine learning, especially in computer vision, many algorithms exist for explaining the decisions of trained (non-linear) neural networks, often on classification tasks (Ancona et al., 2017; Lundberg & Lee, 2017; Montavon et al., 2015; Samek et al., 2019; Shrikumar et al., 2016; Simonyan et al., 2013; Sundararajan et al., 2017). In particular, gradient-based attribution methods have shown empirical success, but can be computationally costly and/or lack theoretical grounding (Lundberg & Lee, 2017; Simonyan et al., 2013), which ultimately limits their utility and scope in scientific applications that benefit from theoretical guarantees.

Contrastive learning recently showed promise in its performance for learning representations while providing theoretical guarantees about its representations (Hyvarinen et al., 2019; Hyvärinen and Morioka, 2016, Chapters 6–7). In this work, we aim to unify the empirical performance of gradient-based attribution methods for generating explanations of large scale datasets with complex non-linear relationships between their variables. We propose a contrastive learning method that provably identifies an attribution map underlying the data. Our framework is depicted in Fig. 8.1, and contributes the following: (1) We formulate an estimation algorithm for global attribution maps based on contrastive learning; (2) We show identifiability guarantees for the (global) attribution map and verify our theory on synthetic datasets.

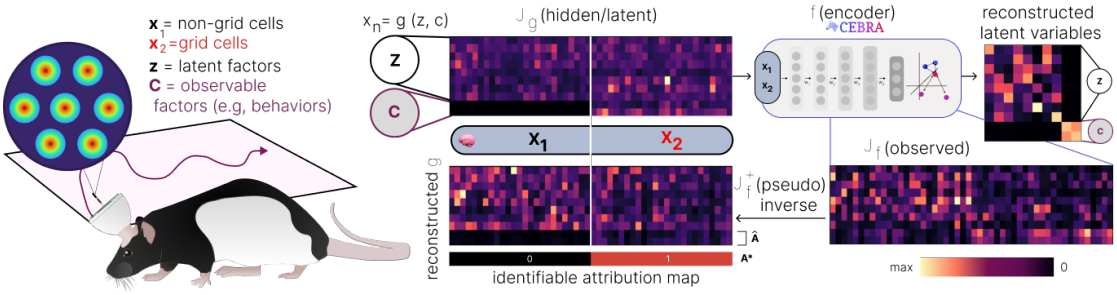


Figure 8.1: **Identifiable attribution maps for time-series data.** Using time-series data (such as neural data recorded during navigation, as depicted), our inference framework estimates the ground-truth Jacobian matrix $\mathbf{J_g}$ (i.e., $\mathbf{g}$ is the observed neural data linked to latents $\mathbf{z}$ and $\mathbf{c}$, where $\mathbf{c}$ is the explicit behavioral latent that would be linked to grid cells) by identifying the inverse data generation process up to a linear indeterminacy $\mathbf{L}$. Then, we estimate the Jacobian $\mathbf{J_f}$ of the encoder by minimizing a generalized InfoNCE objective. Inverting this Jacobian $\mathbf{J_f^+}$, which approximates $\mathbf{J_g}$, allows us to construct the attribution map.

Identifiability of Attribution Maps with Regularized Contrastive Learning

Throughout the paper, we will use a notion of attribution maps grounded in the causal structure of the data generating process. We assume that observations $\mathbf{x} \in \mathcal{X}$ are generated by an injective generative process (mixing function) $\mathbf{g} : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^D$ is the space of observations and $\mathcal{Z} \subseteq \mathbb{R}^d$ is the space of latent factors and d denotes the number of factors. We have $d < D$.

Definition 2 (Data generating process)**.** *We assume a non-linear ICA problem with a mixing function* $\mathbf{g} : \mathcal{Z} \to \mathcal{X}$ *mapping* parts *of the input factors* $\mathbf{z} := [\mathbf{z}_1; \ldots; \mathbf{z}_G] \in \mathbb{R}^d$ *onto outputs,* $x_i = g_i(\mathbf{z}) = g_i([\mathbf{z}_j]_{j \in P_i})$. P_i *is an index set, and* $j \in P_i$ *implies that factor* $\mathbf{z}_j \in \mathbb{R}^{d_i}$ *is used to generate the output* x_i*. We further assume that maximally one of the parts is not observable. All other parts are observable through bijective maps* γ_i *s.t.* $\mathbf{z}_i = \gamma_i(\mathbf{c}_i)$*, where* $\mathbf{c}_i$ *then denotes an observable factor.*

For application of attribution methods to $\mathbf{g}$, we need additional structure in the data generating process. Specifically, we are interested in how the factors $\mathbf{z}$ are *connected* to the output variables $\mathbf{x}$ by means of any non-linear mapping. This gives rise to the following definition of the attribution map:

Definition 3 (Ground-truth attribution maps)**.** *Let* $\mathbf{g}$ *be the mixing function. For all* $\mathbf{x} := \mathbf{g}(\mathbf{z})$ *in the support of* $p(\mathbf{z})$*, the ground-truth attribution map* $\mathbf{A}[\mathbf{g}]$ *has values*

$$A[\mathbf{g}]_{ij} = \begin{cases} 1 & \text{if} \quad \dfrac{\partial g_i(\mathbf{z})}{\partial z_j} \neq 0 \quad \exists \mathbf{z} \in \mathcal{Z} \\ 0 & \text{otherwise.} \end{cases} \tag{8.1}$$

and is specified through the index sets $P_1, \ldots, P_G$ *defined in Def. 2.*

Under these definitions, we aim to identify the attribution map using a suitable representation learning algorithm. We require two components: First, the algorithm needs to be able to identify the latent and observable factors of our data generation process and it is well-studied that contrastive learning algorithms have this property (Hyvarinen et al., 2019; Schneider et al., 2023; Zimmermann et al., 2021b). In the following, let us call p the positive and q the negative sample distribution. We call $(\mathbf{x}, \mathbf{x}^+)$ a positive pair, and all $(\mathbf{x}, \mathbf{x}_i^-)$ negative pairs. The function $\mathbf{f} := [\mathbf{f}_1; \ldots; \mathbf{f}_G]$ is the feature encoder that maps samples into an embedding space, and we apply similarity metrics ϕ_i to the different parts of this feature encoder, abbreviated as $\psi(\mathbf{x}, \mathbf{y}) := \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y}))$. The minimizer of the generalized InfoNCE (van den Oord et al., 2018, Chapter 7) objective

$$\mathcal{L}_N[\psi] = \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p(\mathbf{x}),\ \mathbf{x}^+\sim p_i(\mathbf{x}^+|\mathbf{x}),\\ \mathbf{x}_1^-\ldots\mathbf{x}_N^-\sim q(\mathbf{x}^-|\mathbf{x})}} \left[-\psi(\mathbf{x}, \mathbf{x}_i^+) + \log \sum_{j=1}^{N} e^{\psi(\mathbf{x}, \mathbf{x}_j^-)} \right], \tag{8.2}$$

is $\psi(\mathbf{x}, \mathbf{y}) = \log p(\mathbf{y}|\mathbf{x})/q(\mathbf{y}|\mathbf{x}) + C(\mathbf{x})$ and identifies the ground truth latents up to a linear transform for suitable choice of ϕ, p and q, $\mathbf{f}(\gg(\mathbf{z})) = \mathbf{L}\mathbf{z}$ (Schneider et al., 2023). Importantly, we can separate a meaningful fit from random data as expressed in:

Theorem 1. *Assume ψ^* is a minimizer of the generalized InfoNCE loss under the ICA problem in Def. 2 for $G = 1$ parts in the limit $N \to \infty$. Assume the observable factors $\mathbf{c}$ with $\mathbf{z} = \gamma(\mathbf{c})$ are independent of $\mathbf{z}$. Then, $\psi^* = const.$ is the trivial solution with $\lim_{N\to\infty} \mathcal{L}_N[\psi^*] = \log N$.*

Proof Sketch. The minimizer of the contrastive learning objective is $\psi(\mathbf{x}, \mathbf{y}) = \log p(\mathbf{y}|\mathbf{x})/q(\mathbf{y}|\mathbf{x}) + C(\mathbf{x})$. Assume that the latents $\mathbf{z}$ are independent from the used labels $\mathbf{c}$, then we have $\psi(\mathbf{x}, \mathbf{y}) = C(\mathbf{x}) = \psi^*(\mathbf{x})$ independent of $\mathbf{x}$. Inserting into the objective functions gives $\mathcal{L}[\psi^*] = \log N$. The full proof is given in below, "Proof of Theorem 1". □

To identify the ground-truth attribution map, we apply this learning scheme to each partition of the latent variables. In addition, we need to regularize the Jacobian matrix (Hoffman et al., 2019) of the feature encoder to become minimal. With these constraints, we obtain the objective function for Regularized Contrastive Learning (RegCL) for all parts of the representation:

$$\mathcal{L}_N[\psi; \lambda] = \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p(\mathbf{x}),\\ \mathbf{x}^+\sim p_i(\mathbf{x}^+|\mathbf{x})\ \forall i\in[G]\\ \mathbf{x}_1^-\ldots\mathbf{x}_N^-\sim q(\mathbf{x}^-|\mathbf{x})}} \left[\sum_{i=1}^{G} \Big(-\psi_i(\mathbf{x}, \mathbf{x}_i^+) + \log \sum_{j=1}^{N} e^{\psi_i(\mathbf{x}, \mathbf{x}_j^-)} \Big) + \lambda \|\mathbf{J}_\mathbf{f}(\mathbf{x})\|_F^2 \right]. \tag{8.3}$$

where $\mathbf{J}_\mathbf{f}(\mathbf{x})$ is the Jacobian of the feature encoder $\mathbf{f}$ optimized as part of ψ, $\|\cdot\|_F$ denotes the Frobenius norm and λ is a hyperparameter tuned based on the learning dynamics. λ is tuned to the highest value possible that still allows the InfoNCE component of the loss to stay at its minimum. Intuitively, this loss function solves G non-linear ICA problems using the single feature encoder $\mathbf{f}$ — for observable $\mathbf{z}_i = \gamma_i(\mathbf{c}_i)$

we leverage supervised contrastive learning with continuous labels (Chapter 7), for the non-observable $\mathbf{z}_G$ we apply time-contrastive learning using the time-series structure (Hyvärinen & Morioka, 2016; Schneider et al., 2023).

Theorem 2. *Consider a non-linear ICA problem with mixing function $\gg$ mapping latent factors $\mathbf{z}$ to a signal space such that $\mathbf{x} = \gg(\mathbf{z})$ according to Def. 2. Let $A_{ij} = 1\{\exists \mathbf{z} : |\partial g_i(\mathbf{z})/\partial z_j| \neq 0\}$ be the entries of the global attribution map of the mixing function. Assume that in the limit $N \to \infty$, the differentiable feature encoder $\mathbf{f}$ minimizes the regularized contrastive loss (Eq. 8.3) on $p(\mathbf{z})$. Then, we identify the global attribution map through the pseudo-inverses $\mathbf{J}_\mathbf{f}^+(\mathbf{x})$,*

$$\mathbf{A} = \mathbf{J}_\mathbf{f}^+(\mathbf{x}) \odot \mathbf{L}(\mathbf{x}), \tag{8.4}$$

up to component-wise scaling $\mathbf{L}(\mathbf{x})$ of the entries.

Proof Sketch. The individual parts of the loss function result in $\psi(\mathbf{x}, \mathbf{x}') = \log p_i(\mathbf{z}_i'|\mathbf{z}_i)/q(\mathbf{z}_i')$ from which a linear indeterminacy follows, $\mathbf{f}_i(\mathbf{g}(\mathbf{z})) = \mathbf{L}_i\mathbf{z}$. We can express the result as $\mathbf{f}(\mathbf{g}(\mathbf{z})) = \mathbf{L}\mathbf{z}$ where $\mathbf{L}$ is a block-diagonal matrix with zeros in its lower block triangular part. Hence, $\mathbf{L}^{-1}$ will have the same property. It then follows that $\mathbf{J}_\mathbf{f}(\mathbf{x})\mathbf{J}_\mathbf{g}(\mathbf{z}) = \mathbf{L}$ and since $\mathbf{J}_\mathbf{f}$ has minimum norm everywhere, $\mathbf{J}_\mathbf{f}^+(\mathbf{x})$ is the Moore-Penrose pseudo-inverse of $\mathbf{J}_\mathbf{g}(\mathbf{z})\mathbf{L}^{-1}$. Multiplication with $\mathbf{L}^{-1}$ does not alter the location of zero entries in $\mathbf{J}_\mathbf{g}(\mathbf{z})$, and hence thresholding $\mathbf{J}_\mathbf{f}^+(\mathbf{x})$ across samples $\mathbf{x}$ in the dataset is an estimator of the ground-truth attribution map. The full proof is given below, "Proof of Theorem 2". □

Experimental verification

Experiment setup To verify our theory, we generate a synthetic dataset following Def. 2, cf. Appendix, "Synthetic data design" for details. We sample 10 different datasets with 100,000 samples, each with a different mixing function $\gg$. The mixing functions consist of randomly initialized 3-layer MLPs (Hyvärinen & Morioka, 2016) and we ensure injectivity by monitoring the condition number of each matrix layer, following previous work (Hyvärinen & Morioka, 2016; Zimmermann et al., 2021b). Similar to Schneider et al. (2023), the feature encoder $\mathbf{f}$ is an MLP with three layers followed by GELU activations (Hendrycks & Gimpel, 2016b), and one layer followed by a scaled tanh to decode the latents. We train on batches with 5,000 samples each. The first 2,500 training steps minimize the InfoNCE or supervised loss with $\lambda = 0$, we then ramp up λ to its maximum value over the following 2,500 steps, and continue to train until 20,000 total steps. We compute the R^2 for predicting the observable factors $\mathbf{c}$ from the feature space after a linear regression, and ensure that this metric is close to 100% for both our baseline and contrastive learning models to remove performance as a potential confounder.

As a comparison to previous work, we vary the training method (hybrid contrastive, supervised contrastive, standard supervised) and consider baseline methods for estimating the attribution maps (neuron gradients, Mudrakarta et al., 2018; Simonyan et al.,

Table 8.1: Comparison of attribution methods (rows), and combinations of training/regularization schemes (columns). Our proposed method uses regularized hybrid contrastive learning. Numbers average across different configurations of number of factors (4 to 9), for 10 different datasets. Sub- and superscript values denote the 95% confidence interval obtained through bootstrapping (n=1,000).

	supervised		supervised contrastive		hybrid contrastive	
attribution method	none	regularized	none	regularized	none	regularized **(RegCL)**
Neuron Gradient	$79.2^{81.0}_{77.4}$	$93.0^{94.5}_{91.5}$	$80.6^{82.4}_{78.8}$	$86.7^{89.0}_{84.6}$	$79.2^{81.0}_{77.5}$	$88.0^{90.1}_{85.8}$
Feature Ablation	$83.1^{84.8}_{81.3}$	$88.5^{90.0}_{87.0}$	$84.0^{85.6}_{82.1}$	$84.7^{86.5}_{82.8}$	$82.9^{84.5}_{81.3}$	$85.2^{86.9}_{83.4}$
Integrated Gradients	$81.0^{82.7}_{79.2}$	$84.9^{86.6}_{83.1}$	$81.9^{83.7}_{80.2}$	$82.3^{84.3}_{80.5}$	$83.9^{85.6}_{82.1}$	$86.9^{88.8}_{84.9}$
Shapley, shuffled	$82.0^{83.7}_{80.3}$	$89.2^{90.8}_{87.6}$	$83.3^{84.9}_{81.4}$	$84.6^{86.6}_{82.6}$	$81.6^{83.2}_{80.1}$	$85.1^{87.1}_{83.0}$
Shapley, zeros	$81.0^{82.8}_{79.3}$	$84.9^{86.8}_{83.1}$	$82.0^{83.7}_{80.2}$	$82.4^{84.3}_{80.4}$	$81.6^{83.4}_{79.9}$	$83.2^{85.0}_{81.2}$
J_f^+ **(ours)**	$76.9^{78.7}_{74.9}$	$92.9^{94.5}_{91.5}$	$77.5^{79.4}_{75.5}$	$86.1^{88.3}_{83.8}$	$87.9^{89.5}_{86.3}$	$\mathbf{98.2^{98.9}_{97.4}}$

2013, integrated gradients, Shrikumar et al., 2018; Sundararajan et al., 2017, Shapley values, Lundberg and Lee, 2017; Shapley et al., 1953, and feature ablation, Molnar, 2022), which are commonly used algorithms in scientific applications (Molnar, 2022; Samek et al., 2019). To compute these attribution maps, we leveraged the open source library *captum* (Kokhlikyan et al., 2020). We also compare regularized and non-regularized training. Hyperparameters are identical between training setups, the regularizer λ, and number of training steps are informed by the training dynamics.

Regularized, hybrid contrastive learning identifies the ground truth attribution map. Table 8.1 shows the AUC for recovering using combinations of training schemes (supervised, supervised contrastive, hybrid contrastive), Jacobian regularization, and methods for estimating attribution methods. We investigate the effect of the different factors with an ordinary least squares (OLS) ANOVA (F=17.0, p<1e-5) followed by a Tukey HSD posthoc test, see Appendix, "Statistical analysis" for statistical methods and full results. Both the combination of regularized training followed by estimating the pseudo-inverse ($p<0.01$), and combining regularized training with hybrid contrastive learning ($p<0.001$) significantly outperform all considered baselines.

Contrastive learning is critical for large numbers of latent factors. The importance of using hybrid contrastive learning (which can identify the latent factors) becomes most apparent with an increasing number of latent factors, as we would expect in a realistic dataset. Figure 8.2 shows the variation in performance as we keep the number of observable factors fixed at 2 and vary the number of total latents from 4 to 9 variables. Beyond this value, the drop in R^2 becomes too large, prohibiting us to compute a meaningful attribution map. Performance scales with the number of available training samples, and we observed that increasing dataset size besides 100,000 samples allows to work with even higher numbers of latents.

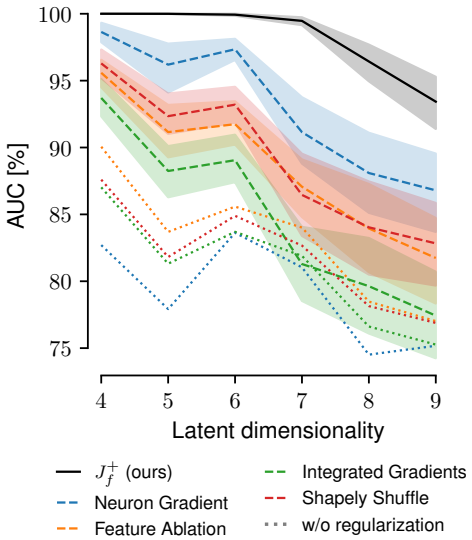


Figure 8.2: RegCL (ours, black) and supervised baselines AUC vs.# of latent factors.

	none	reg.
Neuron Gradient	$69.3^{72.4}_{66.0}$	$91.9^{95.3}_{88.3}$
Feature Ablation	$77.1^{80.4}_{73.8}$	$86.7^{89.9}_{83.1}$
Integrated Gradient	$77.5^{79.6}_{75.4}$	$86.8^{89.9}_{83.4}$
Shapley shuffled	$74.4^{77.5}_{71.2}$	$87.5^{91.0}_{84.0}$
Shapley, zeros	$75.8^{78.7}_{72.6}$	$85.3^{88.6}_{82.0}$
J_f^+ **(ours)**	$84.2^{86.7}_{81.2}$	$\mathbf{99.2^{99.8}_{98.4}}$

Table 8.2: Contrastive learning (CL) can estimate attribution maps w.r.t. latent factors: Results for identifying the attribution map, avg. across 10 seeds and 4–9 latents.

Hybrid contrastive learning allows attribution computation with latent factors. In contrast to supervised algorithms, hybrid contrastive learning allows us to estimate the attribution map with respect to latent factors, i.e., we treat $\mathbf{z}_1$ as the observable, and $\mathbf{z}_2$ as the latent factor. With hybrid contrastive learning, we can continue to estimate the attribution map at AUC=99.2% (Table 8.2).

Related Work

There are two main approaches to model understanding. The first approach is to use interpretable models from the start, e.g., linear regression. The second approach is to explain complex models using post-hoc interpretability methods. Unfortunately, the first approach is often not feasible due to complex non-linearities in the data, and therefore we focus on the second approach, making use of methods that will be discussed below, such as saliency maps.

Depending on the type of explanation we want to obtain, there are different post-hoc interpretability methods available in the literature (Samek et al., 2019). First, we can differentiate between local and global explanations. *Global explanations* provide an interpretable description of the behavior of the model as a whole. *Local explanations* provide a description of the model behavior in a specific neighborhood/for an individual prediction.

In the case of local explanations, we can categorize the methods (non-extensively)

in the following way:

Feature attribution methods are explanations where we assign a weight to each feature in the input space that indicates its importance or effect. We can distinguish between:

- *Perturbation based* compute a relevance score by removing, masking or altering the input, running a forward pass on the new input and measuring the difference with the original input. Methods include LIME or SHAP (Lundberg & Lee, 2017; Ribeiro et al., 2016).

- *Gradient based* methods locally evaluate the gradient $\partial f / \partial x_i$ or variations of it (e.g., absolute value of the gradient). Methods include Integrated Gradients, SmoothGrad, or Grad-CAM (Selvaraju et al., 2017; Smilkov et al., 2017; Sundararajan et al., 2017).

- *Propagation based* methods decompose the prediction of the network going backward (from output to input) following some propagation rules. Common methods are Deep Taylor decomposition and Layer Relevance Propagation (LRP) (Bach et al., 2015; Montavon et al., 2017).

Prototype-based methods are methods based on creating a prototype in the input domain that is interpretable and representative of the abstract learned concept, such as activation maximization (van den Oord et al., 2018). This is used to answer questions such as: What type of input is easier to mis-classify?

For *global explanations* we can differentiate between:

- *Meta-explanations methods* aggregate and analyze a collection of multiple individual explanations to identify general patterns in the model behavior. A recent example is SpRAy (Lapuschkin et al., 2019), which computes meta-explanations by clustering individual heatmaps.

- *Representation-based methods* analyze intermediate representations of a neural network. An example of this approach is network dissection (Bau et al., 2017), which consists of evaluating the semantics of hidden units to determine the model's reliance on concepts that are semantically similar to humans. Another example is TCAV (Kim et al., 2018), which measures the sensitivity of a model's prediction in terms of user-defined concepts.

- *Model distillation methods* create a simpler and more interpretable model that is constructed such that it mimics the original model's predictions. An example is using decision trees (Bastani et al., 2017).

Causal discovery and Identifiability The goal of causal discovery is to learn the causal structure of the data, often represented as a Directed Acyclic Graph (DAG) (Pearl, 2009; Peters et al., 2017). Importantly, there is a deep connection between causal discovery and identifiability as both aim to infer the ground truth data generating process. As a result, a growing number of studies are showcasing this connection (Morioka & Hyvarinen, 2023; Reizinger et al., 2022).

Additional Discussion and Limitations

We demonstrated a theoretically grounded algorithm for estimating attribution maps with identifiability guarantees. While we were able to demonstrate its performance on synthetic datasets matching the theoretical conditions up to real-world data.

Our theoretical results currently hold for fully converged contrastive learning models and true minimizers of the InfoNCE loss (van den Oord et al., 2018) in the limit of infinite data. While Wang and Isola (2020) show favorable properties of contrastive learning in limited data settings, which can be confirmed by our finite data experiments, it is less straightforward to theoretically connect the quality of the attribution score to the goodness of fit of the model. For the purpose of this work, we show that the R^2 of recovering the observable factors is a good indicator, and recommend comparing this to the theoretically best result (of a supervised baseline).

In future work, the presented results should be extended and studied under various violations of the data-distributions, and scaled to real-world datasets.

Conclusions

We proposed a novel approach for estimating attribution maps in time-series data based on regularized, hybrid contrastive learning. Scientific inference in non-linear problems requires identifiable attribution maps estimated for the *data generating process*. We theoretically and empirically showed that contrastive learning can be leveraged to estimate this map by inverting the data generating process. Our empirical results demonstrate the importance of estimating *all latent variables* along with the observable factors for effective estimation of the attribution map. In future extensions of this work, we will apply our approach to real scientific data, e.g., for applications in neuroscience. Even beyond, we think that our findings might spark future work in improving the estimation of global explanations in vision, speech, and language.

Theory

We will now derive identifiability guarantees for the global attribution map under the ICA model described in the main paper. Given a data generating process and a ground truth global attribution map of the data generating process, we aim for a guarantee of

the form

$$\hat{\mathbf{J}}_{\mathbf{g}} = \mathbf{J}_{\mathbf{g}} \odot \mathbf{L} \tag{8.5}$$

for a suitable estimator $\hat{\mathbf{J}}_{\mathbf{g}}$ up to a matrix $\mathbf{L}$ that scales the ground truth derivatives in $\mathbf{J}_{\mathbf{g}}$ point-wise and will hence not affect the "real zeros" in the Jacobians relevant for Def. 3.

We use contrastive learning to obtain a feature encoder $\mathbf{f}$ which identifies the ground-truth latents up to a linear indeterminacy. We structure this feature encoder to reconstruct different parts of the latent representation in different dimensions of the reconstructed latent space.

Then, we estimate the attribution map by computing the pseudo-inverse of the feature encoder's Jacobian, which is directly related to the Jacobian of the mixing function. To obtain the correct pseudo-inverse, we need to obtain a minimum-Jacobian solution of the feature encoding network. We hence introduce a new regularized contrastive learning objective.

The underlying constrained optimization problem is

$$\min_{\mathbf{f}} \|\mathbf{J}_{\mathbf{f}}(\mathbf{x})\|_F^2 \quad \text{s.t.} \quad \phi_i(\mathbf{f}_i(\mathbf{x}), \mathbf{f}_i(\mathbf{y})) = \log \frac{p_i(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} + C_i(\mathbf{x}) \quad \forall i \in [G], \tag{8.6}$$

with the positive sample distribution p_i and the negative sample distribution q. We call $(\mathbf{x}, \mathbf{y}_+)$ the positive pair, and all $(\mathbf{x}, \mathbf{y}_i^-)$ negative pairs. In the following we define $\psi_i(\mathbf{x}, \mathbf{y}) := \phi_i(\mathbf{f}_i(\mathbf{x}), \mathbf{f}_i(\mathbf{y}))$ where $\mathbf{f} := [\mathbf{f}_1; \ldots; \mathbf{f}_G]$ is the feature encoder and ϕ_i are similarity metrics. We re-state the RegCL objective function which is a relaxation of Eq. 8.6:

$$\mathcal{L}_N[\psi; \lambda] = \mathop{\mathbb{E}}_{\substack{\mathbf{x} \sim p(\mathbf{x}), \\ \mathbf{y}^+ \sim p_i(\mathbf{y}|\mathbf{x}) \ \forall i \in [G] \\ \mathbf{y}_1^- \cdots \mathbf{y}_N^- \sim q(\mathbf{y}|\mathbf{x})}} \left[\sum_{i=1}^{G} \Big(-\psi_i(\mathbf{x}, \mathbf{y}_i^+) + \log \sum_{j=1}^{N} e^{\psi_i(\mathbf{x}, \mathbf{y}_j^-)} \Big) + \lambda \|\mathbf{J}_{\mathbf{f}}(\mathbf{x})\|_F^2 \right]. \tag{8.7}$$

In principle, this objective is able to identify an arbitrary amount of separate factor groups (G), given sufficient capacity of the model. The choice of ψ_i for the individual parts of the feature representation depends on the exact distribution underlying data generation, and is discussed below.

Preliminaries

Before proving our results on identifiable attribution maps, it is useful to restate a few known results from the literature, concerning properties of the InfoNCE loss. Hyvarinen et al. (2019) showed that contrastive learning with auxiliary variables is identifiable up to permutations or linear transformations for conditionally exponential distributions. Zimmermann et al. (2021b) related this to identifiability for models trained with the InfoNCE loss, and showed that assumptions about the data-generating process can be incorporated in to the choice of loss function. Schneider et al. (2023) then formulated a supervised contrastive learning objective based on selecting the positive

and negative distributions in the generalized InfoNCE objective.

We will first re-state the minimizer of the InfoNCE loss (Def. 8.2) used in our algorithm:

Proposition 2 (restated from Chapter 7, Schneider et al. (2023))**.** *Let $p(\cdot|\cdot)$ be the conditional distribution of the positive samples, $q(\cdot|\cdot)$ the conditional distribution of the negative samples and $p(\cdot)$ the marginal distribution of the reference samples. The generalized InfoNCE objective (Def. 8.2) is convex in ψ with the unique minimizer*

$$\psi^*(\mathbf{x}, \mathbf{y}) = \log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} + C(\mathbf{x}), \quad \textit{with} \quad \mathcal{L}_N[\psi^*] = \log N - \mathcal{D}_{\mathrm{KL}}(p(\cdot|\cdot)\|q(\cdot|\cdot)) \tag{8.8}$$

for $N \to \infty$ on the support of $p(\mathbf{x})$, where $C : \mathbb{R}^d \to \mathbb{R}$ is an arbitrary mapping.

Proof. See Chapter 7 and Schneider et al. (2023), but note that we added the batch size N. □

We also re-state

Proposition 3 (restated Proposition 6 in Schneider et al. (2023))**.** *Assume the learning setup in Def. 1 (Schneider et al., 2023), and that the ground-truth latents $\mathbf{u}_1, \ldots, \mathbf{u}_T$ for each time point follow a uniform marginal distribution and the change between subsequent time steps is given by the conditional distribution of the form*

$$p(\mathbf{u}_{t+\Delta t}|\mathbf{u}_t) = \frac{1}{Z(\mathbf{u}_t)} \exp \delta(\mathbf{u}_{t+\Delta t}, \mathbf{u}_t) \tag{8.9}$$

where δ is either a (scaled) dot product (and $\mathbf{u}_t \in \mathcal{S}^{n-1} \subset \mathbb{R}^d$ lies on the $(n-1)$-sphere $\mathcal{S}^{n-1}$) or an arbitrary semi-metric (and $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^d$ lies in a convex body $\mathcal{U}$). Assume that the data generating process $\mathbf{g}$ with $\mathbf{ss}_t = \mathbf{g}(\mathbf{u}_t)$ is injective. Assume we train a symmetric CEBRA (Schneider et al., 2023) model with encoder $\mathbf{f} = \mathbf{f}'$ and the similarity measure including a fixed temperature $\tau > 0$ is set to or sufficiently flexible such that $\phi = \delta$ for all arguments. Then $\mathbf{h} = \mathbf{h}' = \mathbf{g} \circ \mathbf{f}$ is affine.

Proof. For δ being the dot product, the result follows from the proof of Theorem 2 in Zimmermann et al. (2021b). For δ being a semi-metric, the result follows from the proof of Theorem 5 in Zimmermann et al. (2021b). □

Positive distributions for self-supervised and supervised contrastive learning

Self-supervised contrastive learning Up to one of the parts in the latent representation $\mathbf{z}$ can be estimated using self-supervised learning by leveraging time information in the signal. The underlying assumption is that latents vary over time according to a distribution we can model with ψ. For instance, Brownian motion $p(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)}) = \mathcal{N}(\mathbf{z}^{(t+1)} - \mathbf{z}^{(t)}|0, \sigma^2\mathbf{I})$ can be estimated by selecting $\phi(\mathbf{x}, \mathbf{y}) = -\|\mathbf{x} - \mathbf{y}\|^2$. On the hypersphere with a vMF conditional across timesteps, the dot product is a suitable

choice for $\phi(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$. Due to Proposition 3, this training scheme is able to identify the ground truth latents up to a linear indeterminacy.

Supervised contrastive learning For supervised contrastive learning, we uniformly sample a timestep (and hence, a sample $\mathbf{x}$) from the dataset. This timestep is associated to the label $\mathbf{c}$, and we then sample $\mathbf{c}'$ from the conditional distribution $p(\mathbf{c}'|\mathbf{c})$. We select the nearest neighbour to $\mathbf{c}'$ with the corresponding sample $\mathbf{x}'$.

The conditional distribution $p(\mathbf{c}'|\mathbf{c})$ can be constructed as an *empirical* distribution: For instance, if we assume non-stationarity, $\mathbf{c}^{(t-1)} - \mathbf{c}^{(t)}$ can be computed across the dataset. Let us call this distribution $\hat{p}(\mathbf{c}' - \mathbf{c})$. Then, sampling from $p(\mathbf{c}'|\mathbf{c})$ can take the form of sampling $\mathbf{c}' = \mathbf{c} + \Delta$ with $\Delta \sim \hat{p}(\mathbf{c}' - \mathbf{c})$.

If this approximation is correct under the underlying latent distribution, have we have $p(\mathbf{c}'|\mathbf{c}) \det \mathbf{J}_\gamma^{-1}(\mathbf{c}') = p(\mathbf{z}'|\mathbf{z})$. This means that the solutions of the supervised and self-supervised contrastive learning solutions coincide.

Superposition of self-supervised and supervised contrastive learning Depending on the assumptions about the ground truth data distribution, different estimation schemes can be combined to obtain a latent representation. In the end, the feature encoder $\mathbf{f}$ should identify the original latents $\mathbf{z}$ up to a linear transformation,

$$\mathbf{f}(\mathbf{g}(\mathbf{z})) = \mathbf{L}\mathbf{z}. \tag{8.10}$$

Our goal is to obtain block-structure in $\mathbf{L}$, with zeros in the lower block triangular part of the matrix.

This is possible by simultaneously solving multiple contrastive learning objectives, which requires

$$\mathbf{f}_i(\mathbf{g}(\mathbf{z})) = \mathbf{L}_i\mathbf{z}. \tag{8.11}$$

for each part i of the latent representation. Assume without loss of generality that we apply self-supervised contrastive learning to the G-th part, and supervised contrastive learning to all remaining parts. For supervised contrastive learning we then obtain

$$\mathbf{f}_i(\mathbf{g}(\mathbf{z})) = \mathbf{L}_i\mathbf{z} = \mathbf{L}_i'\mathbf{z}_i. \tag{8.12}$$

If all latents $\mathbf{z}$ satisfy the conditions for time-contrastive learning, we can then also apply time-contrastive learning to the full representation, which gives us the following constraints:

$$\mathbf{f}_i(\mathbf{g}(\mathbf{z})) = \mathbf{L}_i\mathbf{z} = \mathbf{L}_i'\mathbf{z}_i \quad \forall i \in [G-1] \tag{8.13}$$

$$\mathbf{f}(\mathbf{g}(\mathbf{z})) = \mathbf{L}\mathbf{z} \tag{8.14}$$

from which we can follow the matrix structure

$$\mathbf{f}(\mathbf{g}(\mathbf{z})) = \operatorname{diag}(\mathbf{L}_1, \ldots, \mathbf{L}_G) \tag{8.15}$$

In cases where this is not possible, note that it is always possible to treat all contrastive learning problems separately, and learn separate regions of the feature space in $\mathbf{f}$. This gives the same result, but re-uses less of the representation (e.g., the self-supervised part of the representation would be learned separately from the supervised part).

Consider a time-series dataset where $p(\mathbf{z}_t|\mathbf{z}_{t-1})$, i.e., all latents, follow Brownian motion. We can then produce the solution

$$\psi_i(\mathbf{x}, \mathbf{x}') := \phi_i(\mathbf{f}_i(\mathbf{x}), \mathbf{f}_i(\mathbf{x}')) = \log \frac{p(\mathbf{c}_i'|\mathbf{c}_i)}{q(\mathbf{c}_i'|\mathbf{c}_i)} \quad i \in \{1, \dots, G-1\} \tag{8.16}$$

$$\psi_G(\mathbf{x}, \mathbf{x}') := \sum_{i=1}^{G} \phi_i(\mathbf{f}_i(\mathbf{x}), \mathbf{f}_i(\mathbf{x}')) = \log \frac{p(\mathbf{z}'|\mathbf{z})}{q(\mathbf{z}'|\mathbf{z})} = \log \frac{p(\mathbf{z}_G'|\mathbf{z}_G)}{q(\mathbf{z}_G'|\mathbf{z}_G)} + \sum_{i=1}^{G-1} \log \frac{p(\mathbf{c}_i'|\mathbf{c}_i)|\mathbf{J}_{\gamma_i}^{-1}(\mathbf{z}_i')|}{q(\mathbf{c}_i'|\mathbf{c}_i)|\mathbf{J}_{\gamma_i}^{-1}(\mathbf{z}_i')|} \tag{8.17}$$

in case our training distributions for supervised contrastive learning, $p(\mathbf{c}_i|\mathbf{c}_i)$ are a sufficiently good approximation of the variation in the ground truth latents, we can select $\psi_G(\mathbf{x}, \mathbf{y}) := \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y}))$ to be trained on the whole feature space using self-supervised learning, while all other objectives on ψ_i would solve supervised contrastive losses. If this training setup is not possible, it would be required to parametrize $\psi_G(\mathbf{x}, \mathbf{y}) := \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y}))$ as a separate part of the feature space.

While it is beyond the scope of the current work to thoroughly investigate the trade-offs between the two methods, our verification experiments assume the former case: The time contrastive objective is applied to the whole objective function, and the behavior contrastive objective to the previous latent variable groups.

Proof of Theorem 1

An interesting property of contrastive learning algorithms is the natural definition of a "goodness of fit" metric for the model. This goodness of fit can be derived from the value of the InfoNCE metric which is bounded from below and above as follows (Schneider et al., 2023):

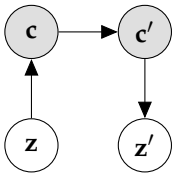
$$\log N - D_{\mathrm{KL}}(p||q) \leq \mathcal{L}_N[\psi] \leq \log N. \tag{8.18}$$

In scientific applications, we can leverage the distance to the trivial solution $\log N$ as a quality measure for the model fit. Theorem 1 states that if during supervised contrastive learning with labels $\mathbf{c}$ there is no meaningful relation between $\mathbf{c}$ and $\mathbf{x}$, we will observe a trivial solution with loss value at $\log N$.

For the following proof, let us recall from Def. 2 that we can split the latents $\mathbf{z}$ that fully define the data through the mixing function, $\mathbf{x} = \mathbf{g}(\mathbf{z})$. We can split $\mathbf{z}$ into different parts, $\mathbf{z} = [\mathbf{z}_1, \dots, \mathbf{z}_G]$ and assume that $\mathbf{c}_i$ is the observable factor corresponding to the i-th part. For notational brevity, we omit the i in the following formulation of the proof without loss of generality.

Proof of Theorem 1

Proof. Assume that the distribution p is informed by labels. In the most general case, we can depict the sampling scheme for supervised contrastive learning with continuous labels $\mathbf{c}$ and $\mathbf{c}'$ and latents $\mathbf{z}$ and $\mathbf{z}'$ with the following graphical model:



The reference sample $\mathbf{x}$ is linked to the observable factor/label $\mathbf{c}$, and the conditional $p(\mathbf{c}'|\mathbf{c})$ links both samples. In particular, $\mathbf{z}'$ and hence $\mathbf{x}'$ are selected based on $\mathbf{c}'$ in the dataset.

The distributions for positive and negative samples then factorize into

$$p(\mathbf{z}'|\mathbf{z}) = \int\int d\mathbf{c}'d\mathbf{c}p(\mathbf{z}'|\mathbf{c}')p(\mathbf{c}'|\mathbf{c})p(\mathbf{c}|\mathbf{z}) \tag{8.19}$$

$$q(\mathbf{z}'|\mathbf{z}) = \int\int d\mathbf{c}'d\mathbf{c}p(\mathbf{z}'|\mathbf{c}')q(\mathbf{c}'|\mathbf{c})p(\mathbf{c}|\mathbf{z}) \tag{8.20}$$

and note that only $p(\mathbf{c}'|\mathbf{c})$ and $q(\mathbf{c}'|\mathbf{c})$ are selected by the user of the algorithm, the remaining distributions are empirical properties of the dataset.

We can compute the density ratio

$$\frac{p(\mathbf{z}'|\mathbf{z})}{q(\mathbf{z}'|\mathbf{z})} = \frac{\int\int d\mathbf{c}'d\mathbf{c}p(\mathbf{z}'|\mathbf{c}')p(\mathbf{c}'|\mathbf{c})p(\mathbf{c}|\mathbf{z})}{\int\int d\mathbf{c}'d\mathbf{c}p(\mathbf{z}'|\mathbf{c}')q(\mathbf{c}'|\mathbf{c})p(\mathbf{c}|\mathbf{z})} \tag{8.21}$$

In the case where latents and observables are independent variables, we have $p(\mathbf{z}'|\mathbf{c}') = p(\mathbf{z}')$ and $p(\mathbf{c}|\mathbf{z}) = p(\mathbf{c})$. The equation then reduces to

$$= \frac{\int\int d\mathbf{c}'d\mathbf{c}p(\mathbf{z}')p(\mathbf{c}'|\mathbf{c})p(\mathbf{c})}{\int\int d\mathbf{c}'d\mathbf{c}p(\mathbf{z}')q(\mathbf{c}'|\mathbf{c})p(\mathbf{c})} \tag{8.22}$$

$$= \frac{p(\mathbf{z}')\int\int d\mathbf{c}'d\mathbf{c}p(\mathbf{c}'|\mathbf{c})p(\mathbf{c})}{p(\mathbf{z}')\int\int d\mathbf{c}'d\mathbf{c}q(\mathbf{c}'|\mathbf{c})p(\mathbf{c})} = 1. \tag{8.23}$$

Consequently, the minimizer is $\psi(\mathbf{x}, \mathbf{y}) = C(\mathbf{x})$ and we obtain the maximum value of the loss with $\mathcal{L}[\psi] = \log N$ in the limit of $N \to \infty$. Note, for any symmetrically parametrized similarity metric (like the cosine or Euclidean loss), it follows that $\psi(\mathbf{x}, \mathbf{y}) = \psi$ is constant, i.e., the function collapses onto a single point.

□

Proof of Theorem 2

Proof. If $\mathbf{f}$ is a minimizer of the InfoNCE loss under the assumed generative model, it follows that we part-wise identify the underlying latents,

$$\mathbf{f}(\mathbf{g}(\mathbf{z})) = \mathbf{B}\mathbf{z} \tag{8.24}$$

with some block diagonal matrix $\mathbf{B}$. By taking the derivative w.r.t. $\mathbf{z}$ it follows that

$$\mathbf{J}_\mathbf{f}(\mathbf{x})\mathbf{J}_\mathbf{g}(\mathbf{z}) = \mathbf{B}. \tag{8.25}$$

We need to show that at each point $\mathbf{z}$ in the factor space, we can recover $\mathbf{J}_g$ up to some indeterminacy. We will re-arrange the equation to obtain

$$\mathbf{J}_\mathbf{f}(\mathbf{x})\mathbf{J}_\mathbf{g}(\mathbf{z})\mathbf{B}^{-1} = \mathbf{I}, \tag{8.26}$$

$$\mathbf{J}_\mathbf{f}(\mathbf{x})\tilde{\mathbf{J}}_\mathbf{g}(\mathbf{z}) = \mathbf{I}. \tag{8.27}$$

It is clear that for each point in the support of p, $\mathbf{J}_\mathbf{f}(\mathbf{x})$ is a left inverse of $\tilde{\mathbf{J}}_\mathbf{g}(\mathbf{z})$.

$$\mathbf{J}_\mathbf{f}(\mathbf{x}) = \tilde{\mathbf{J}}_\mathbf{g}^+(\mathbf{z}) + \mathbf{V}, \mathbf{v}_i \in \ker \tilde{\mathbf{J}}_\mathbf{g}(\mathbf{z}) \tag{8.28}$$

Among these solutions, it is well-known that the minimum norm solution $\mathbf{J}^*$ to

$$\min_{\mathbf{J}(\mathbf{z})} \|\mathbf{J}(\mathbf{z})\|_F^2 \text{ s.t. } \mathbf{J}(\mathbf{z})\mathbf{J}_\mathbf{g}(\mathbf{z}) = \mathbf{I} \tag{8.29}$$

is the Moore-Penrose inverse, $\mathbf{J}^*(\mathbf{z}) = \tilde{\mathbf{J}}_\mathbf{g}^+(\mathbf{z})$. By invoking assumption (2), we arrive at this solution and have

$$\mathbf{J}_\mathbf{f}(\mathbf{x}) = \tilde{\mathbf{J}}_\mathbf{g}^+(\mathbf{z}) \tag{8.30}$$

$$\mathbf{J}_\mathbf{f}^+(\mathbf{x}) = \tilde{\mathbf{J}}_\mathbf{g}(\mathbf{z}) \tag{8.31}$$

$$\mathbf{J}_\mathbf{f}^+(\mathbf{x}) = \mathbf{J}_\mathbf{g}(\mathbf{z})\mathbf{B}^{-1} \tag{8.32}$$

Because $\mathbf{B}$ is block-diagonal with zeros in the off-diagonal blocks, this also applies to $\mathbf{B}^{-1}$. It follows that

$$\mathbf{J}_\mathbf{f}^+(\mathbf{g}(\mathbf{z})) \propto \mathbf{J}_\mathbf{g}(\mathbf{z}) \tag{8.33}$$

concluding the proof. □

9

Discussion

In an era of increasingly large datasets in neuroscience, it seems crucial to develop robust machine learning tools. But why do we even need new ways of processing and analyzing data?

Firstly, due to the nature of scientific questions posed in neuroscience: How does neural activity relate to complex, naturalistic behaviors? How and where are highly structured, yet noisy sensory signals – proprioception, visual scenes, speech, etc. – parsed and processed to inform future actions? Many such fundamental questions in neuroscience can likely only be addressed by placing animal models into complex and naturalistic experimental environments, and performing recordings covering large parts of their brain[1]. Structural and dynamical data needs to be integrated to reason about their relationship, and ideally data from many experiments can be used and integrated to enhance our understanding of how processing is distributed across various areas of the brain. Methods that are able to parse and process large datasets can aid both in hypothesis formation (discovery driven) during a data exploration phase, and be used for providing evidence towards or against an existing hypothesis (hypothesis driven).

Secondly, much like we require physical understanding of the data acquisition process through measurement instruments like electrodes, cameras, or the dynamics of a calcium indicator, we need to advance our understanding of machine learning algorithms to utilize them effectively and reliably in an experimental setup. We want to analyze data recorded from complex experimental setups, usually without underlying clear mathematical theories like e.g., in physics, and still be able to trust the data processed and analyzed with our machine learning tools. The data complexity can arise either from naturalistic behaviors (yielding high-dimensional behavioral recordings

[1] If these recordings are performed across multiple recording sessions, we additionally need to solve the problem of data integration across these different sessions.

with a lot of variability), the use of naturalistic stimuli (real world images/videos instead of synthetic, hand-designed stimuli like gratings), and from the acquisition systems used which record high-dimensional data (e.g., high resolution videos) at high spatial and/or temporal frequencies.

Finally, large scale recordings and closed-loop design of experiments can potentially speed up scientific discovery, and the kind of discoveries that are possible. When experimental setups are constrained (by the areas to record from, the recording time, etc.) in a classical hypothesis-driven approach to scientific inference, experimenters need to employ heuristics based on their current incomplete knowledge of the system behavior. With easier access to recordings, it is possible to find patterns that would have been otherwise missed due to this *a priori* sub-selection of data. The dichotomy between this form of data-driven and the typical form of hypothesis-driven research has been extensively (and controversially) discussed, see Zalta (2020) for an overview. However, Mazzocchi (2015) summarize that "[the] data-driven approach constitutes a novel tool for scientific research", and will supplement, rather than replacing the established, hypothesis-driven way of conducting experiments. Yet, we can argue that data-driven design of experiments allows a two-step process where a large dataset is first used to find potentially interesting or even unexpected patterns in the data, followed by employing the established toolbox of hypothesis-driven inference to investigate the cause of an observed phenomenon. For instance, neuron populations in unexpected areas could be discovered by such a data-driven approach and a suitable attribution technique, followed by an experiment employing optogenetics, lesions, or other means of interventions to causally study the role of these populations.

In the context of this motivation, I presented progress on machine learning systems suitable for both the data processing (Chapters 2–5) and data analysis (Chapters 6–7) phases, and for informing hypotheses relevant for the next data collection phase through interpretable machine learning (Chapter 8). I investigated these algorithms from the perspective of core machine learning on commonly used and newly constructed benchmark datasets (Chapters 2–4,6), and in their application within the context on actual neuroscience datasets (Chapters 7–8). In the following, I will discuss the key findings in the light of the current research landscape.

The importance of robustness and adaptation for machine vision

In Chapters 2–5, I investigated the application of computer vision systems for data processing. I started with core computer vision applications and benchmarks in Chapters 2–4, while Chapter 5 showed the relevance of robustness and adaptation to a scientific application of pose estimation algorithms.

When applying deep networks to image classification and other data processing problems, it is crucial to ensure robustness to perturbations and other systematic shifts at test-time. Sometimes, *ad-hoc* robustness[2] is hard to achieve. In such cases, we fall

[2]With *ad-hoc* robustness, I refer to models that are by design robust to any possible perturbation,

back to the model's ability to adapt to a perturbation or distribution shift at test-time, and to make this problem feasible, allow access to samples from the test-distribution.

I explored these topics from the perspective of both gradient-free (Chapter 2) and gradient-based model adaptation (Chapters 3 and 4). I, along with my collaborators, proposed new adaptation methods, datasets, and benchmarking setups. Availability of suitable evaluation protocols and baseline methods is crucial to ensure the measurement of progress in the field.

Batch norm adaptation presented in Chapter 2 showed the large discrepancy in the evaluation of test-time adaptation tools based on the respective evaluation scenario. In the context of applications in the life sciences, distribution shifts commonly occur systematically, rather than abruptly changing between subsequent samples in a dataset. This can be due to changes in the recording setup, camera systems and their perspective, the light source during imaging, surroundings or image background, the reference electrode during electrophysiological recordings, or also the identity of the animal. For instance, Zhao et al. (2020b) discuss such fluctuations for retinal recordings based on small variations in temperature, light conditions and expression levels of their used biosensor. Goh et al. (2017) discuss these effects more broadly in biological data analysis. The WILDS benchmark (Koh et al., 2020) which I used in Chapter 3 also have multiple examples of biological datasets[3]. Adapting our evaluation protocol to acknowledge this evaluation setup, and developing methods like Batch Norm adaptation targeted at it, allowed consistent gains in classification performance.

While batch norm adaptation is crucial for the evaluation of models under systematic domain shift, it also fell short of adapting very large models, or models readily trained on large, diverse datasets. In this context, I investigated self-learning objectives that allowed to fine-tune the model during deployment. These techniques can successfully adapt models on all scales, regardless of their "history", i.e., training or existing adaptation objectives. While powerful, a very important limitation of these works is their instability over longer adaptation timescales. Yet, and importantly, we saw that all tested models – small or large in terms or parameters, and trained on small or large scale datasets – could benefit from additional adaptation to their test domain. However, in the light of the instability of these approaches it remains an open question whether this gain in performance is worth the potential safety implications during deployment.

In subsequent work (Niu et al., 2022b; Wang et al., 2022), multiple attempts have been made to mitigate this "collapse" behavior of self-learning objectives, but so far with limited success: In Chapter 4, I critically examined the existing progress, and demonstrated that a "memory-less" method based on periodic model resetting can outperform or perform on-par with existing state-of-the-art approaches. Up to date, we were unable to find a technique that is effective at adapting a pre-trained model while keeping the history of knowledge of its adaptation path and meaningfully improves

or distribution shift, without access to further information or examples from the test distribution. See Chapter 2 for an extensive discussion.

[3]Camelyon17: whole slide imaging, RxRx1: cell images, OGB-MolPCBA: molecular graphs

over our baseline. Our work also again highlighted the importance of benchmark design: Based on previous practices, it was in some cases not possible to detect the collapsing behavior of certain algorithms on the established benchmarks. Our dataset, CCC, extended these previous benchmarks by an order of magnitude in length[4].

How do considerations of robustness and adaptation relate to real-world applications of vision algorithms? In Chapter 5, I studied this question in the context of an application in pose estimation. In experimental recording setups, two big sources of variation can be the recorded subject (i.e., the identity of an animal), and changes in the recording setup itself through use of different cameras or their positioning, environmental conditions, etc. How to build models that perform favorably in these conditions? In computer vision, a question was whether transfer learning from ImageNet-pretrained weights is required for good downstream task performance. He et al. (2018) showed that on downstream tasks like object recognition, training networks from scratch can match transfer learning performance, if the optimization setup is carefully designed.

In Chapter 5 I investigated this finding for the case of evaluating ImageNet-pretrained models for pose estimation. My colleagues and I designed two datasets, Horse-10, and Horse-C, which introduce different distribution shifts, allowing us to benchmark the performance of models within-domain (no corruption applied, train/test animal identities match) and compare this to out-of-domain performance (by adding image corruptions, varying the animal identity, or both). Our study replicates the finding of He et al. (2018) for the in-domain data, but shows that for out-of-domain robustness to either shift, pre-training makes a considerable difference.

This is relevant for the design of future models: Up to date, pose estimation toolboxes like DeepLabCut use ImageNet pre-trained weights – however, potentially other pre-training tasks can further improve over ImageNet pretraining. Current work on "SuperAnimal models" (Ye et al., 2023a) investigates more general ways of pre-training better suited for pose estimation.

It is an interesting question how the field of robustness and adaptation will evolve in the era of foundation models. An effect I observed was that very large models obtained by weakly supervised pre-training (Mahajan et al., 2018) or noisy student pre-training (Xie et al., 2020a) considerably improved base model performance. While self-learning was still effective at further improving model performance, other techniques like batch norm adaptations had no additional effect on performance in these larger models. Without dismissing work on adaptation, today the most effective (and pragmatic) approach to achieving robust generalization is large-scale pre-training, covering as many meaningful variations through large datasets and/or data augmentation as possible. Some examples include CLIP (Radford et al., 2019) and noisy student training (Xie et al., 2020a) which were however both trained on proprietary datasets. Replication

[4]CCC is also suitable for generating even longer benchmarking runs with identical methodology, and the open source code base my colleagues and I released is suitable for such follow up studies. However, as of now, the current length seemed sufficient for approximating the asymptotic behavior of the considered algorithms.

efforts like OpenCLIP (Cherti et al., 2023; Ilharco et al., 2021) also exists on datasets like LAION-5B (Schuhmann et al., 2022).

Do foundation models alleviate the need for adaptation algorithms? Probably not, for two reasons: Firstly, an obvious downside of such large models is higher deployment cost, higher latency during inference, etc. The question of how task-relevant knowledge can be distilled into smaller models in itself is tightly related to the discussed class of adaptation methods, and smaller models are again more likely to benefit from adaptation. Secondly, I showed that even such large models can benefit from adaptation. Arguably, adaptation of models at test-time should *always* improve model performance with the right objective, as it effectively focuses a model's scope to a more narrow task or domain. Thirdly, beyond the work presented here, also currently employed large-language models (LLMs) are usually used in conjunction with further fine-tuning, potentially continually when deployed alongside a feedback workflow. This makes research on optimal adaptation mechanisms very relevant, and e.g. low rank adaptation (LoRA; Hu et al., 2021) is widely used in this context.

To apply adaptation methods in outcome-sensitive domains like the life sciences and health, further work is needed on objective functions suitable for adaptation on long-time scales. Algorithms should be empirically, and ideally theoretically stable also in the asymptotic limit of very long deployment times. As of now, usage of large pre-trained models promises the largest gains in model robustness. If processing time and memory requirements are not the main bottleneck, modern foundation models can be considered the most pragmatic solution to obtaining robust predictions in practice. The largest challenge in applying adaptation techniques in practice are their missing safety guarantees: Truly continual robust adaptation is still not practically possible, and if it is, only with marginal or no improvements over more naïve baseline algorithms.

The importance of identifiability for scientific inference

The second theme in this work, explored in Chapters 6–8, are the opportunities of identifiable representation learning algorithms (e.g., self-supervised learning) for applications in neuroscience. In contrast to the first theme, here the ML system is not used to process the data. Instead, we aim to perform statistical inference, and gain insights into the dataset, and/or evidence in favor or against our hypothesis. Ideally, such data analysis can then lead to formation of new hypotheses, and inform the collection of data in the next iteration of the experiment, or inform the design of interventions.

Is identifiability strictly required for representation learning algorithms in this context? On purely technical applications in speech, vision and language, the trend today is towards large scale models that use objectives like next of sequence prediction (Radford et al., 2019) or denoising autoencoders (Devlin et al., 2019) using the transformer architecture at scale (Vaswani et al., 2017). While some theory exists for identifiability of e.g. a GPT model trained on sequences (Roeder et al., 2021), and identifiability for

variational autoencoders is well-understood (Khemakhem et al., 2020a), it is less clear on how to expand such results in masked auto-encoding frameworks for continuous valued data[5]. The previous generation of self-supervised learning models was powered by contrastive learning, which also became successful in supervised learning contexts for classification (Khosla et al., 2020) and are still popular as multi-modal text-image foundation models (Radford et al., 2021).

In Chapter 6, I investigated the connection of modern contrastive learning algorithms trained with the InfoNCE objective to identifiability through the lens of non-linear independent component analysis (ICA). Through the design of suitable loss functions, we can influence the learned representation, and incorporate additional assumptions about the data generating distribution. Building on earlier work (Hyvarinen & Morioka, 2016, 2017; Hyvarinen et al., 2019; Wang & Isola, 2020), our work connects the currently used InfoNCE metrics to assumptions about the data generating process. I also discussed the empirical trade-offs under violations of these theoretical assumptions. Notably, my colleagues and I were able to show that these identifiability guarantees also hold when scaling models to ImageNet-sized datasets[6].

For data analysis applications in science, it is crucial that algorithms satisfy such identifiability guarantees, which hold in empirically relevant settings. This gives a clean understanding of how assumptions translate into properties of the inferred representations and models. These assumptions should closely match the realistic recording settings as best as possible: For instance, if variability in spike statistics and shapes is known to be present in a neural dataset, an algorithm should not assume an overly simplified (e.g., Poisson) distribution for modeling.

Given the theoretical and empirical success of contrastive learning across various disciplines in machine learning Chen et al. (2020a), Mikolov et al. (2013), and Schneider et al. (2019), I derive loss functions specifically targeted at scientific inference in Chapter 7, and provide identifiability guarantees for them. This translates into favorable properties like consistency/reproducibility, and a clean definition of how "latents" and their dimensionality are defined within this framework. These properties are essential when applying non-linear algorithms to neuroscientific problems. In Chapter 8, I further extend these guarantees towards the computation of attribution methods connecting the latent and signal space for interpretability.

Identifiability for the purpose of interpretability under clear assumptions is also connected to causal representation learning (Schölkopf et al., 2012). We are interested in making inferences about the causal structure underlying the data generation, and dissecting/modeling parts of it. A key question is the relation how elements in the

[5]One approach to translate from the continuous domain to the discrete is through quantization, as demonstrated by Baevski et al. (2020b) for applications in speech processing.

[6]In our case, I proposed the "3DIdent" dataset which is comprised of 250,000 images at size 224×224px, with controlled variations in the data-generating factors. In image space, there is a lot of interplay between factors, as e.g. the position of the light source and rotation of the object jointly determine the shade observable in the image. In addition, the data generating process is explicitly not using a neural net, but a full rendering pipeline, making it a challenging problem to infer the data-generating factors that art input to the pipeline.

signal space (neurons, bodyparts, etc.) relate to latent variables inferred by our models.

As an example for the data-driven design of interventions, let us consider the use of brain-machine interfaces to study population activity during learning and adaptation: Sadtler et al. (2014) discuss how low dimensional subspaces of activity constrain adaptations to new tasks. In their case, the "neural manifold" is obtained using PCA, a linear method. On the considered scale of 100 recording channels this revealed a 10d subspace of interest in which different perturbations (within manifold vs. without manifold) could be computed. These data-driven interventions were then applied in subsequent experiments, and used to study learning dynamics over short (Golub et al., 2018) and long (Oby et al., 2019) timescales.

While these studies used a 2D cursor control task using linear dimensionality reduction and decoding algorithms, it would be very interesting to leverage non-linear statistical tools to make such paradigms amendable to even more complex tasks. How can we efficiently represent and perturb on manifolds modeled by a non-linear feature encoder, vs. a PCA subspace? How can we compare and measure changes between these latent embedding spaces, e.g., during learning? These are interesting avenues for future investigation, and non-linear decoding algorithms offer the opportunity to work towards these goals.

How can we encourage future algorithm development for identifiable representation learning? Benchmarking in machine learning was arguably a key driver of success in terms of methods development. The ability to specify desired properties of algorithms in forms of benchmark data and evaluation metrics allows objective comparisons (if agreement about benchmarks exist). Right now, existing neuroscience benchmarks for latent variable and representation learning models mostly focus on modeling data in its signal space (i.e., neural activity), e.g. in the benchmarks proposed by Pei et al. (2021b), Schrimpf et al. (2018), and Turishcheva et al. (2023), which is reasonable given the strong focus in the field towards generative models (see Introduction, Table 1.1, or Hurwitz et al. (2021)). Contrastive learning and other self-supervised algorithms allow to decouple the representation learning phase from the generation of neural activity. This development puts more emphasis on the original problem – identifying the latent representation – rather than modeling the neural activity (including task- and/or representation irrelevant noise). In future benchmarks, we should increasingly adopt metrics that quantify consistency, recovery of ground-truth factors on synthetic data, and clear notions on metrics for comparing embedding spaces across subjects or runs (see e.g., Williams et al., 2021). I made a few proposals in Chapter 7. Such practices are already common in machine learning, including for the benchmarking of generative models (Locatello et al., 2019a), and the synthetic and real benchmark settings I discussed in Chapter 6–8. Progress in this area will ultimately also evolve the fields agreement on what is considered a "latent space", "neural manifold", and will broaden these terms based on the employed algorithms.

Non-linear, interpretable algorithms for neuroscientific inference

Chapters 7 and 8 introduced non-linear, deep learning based algorithms for statistical inference in neuroscience. Linking behavior and neural activity is a research question that might be addressable with the wealth of data we are now able to record (Urai et al., 2022a), and new algorithms are needed. The primary goal is not to solve a downstream task (like in a vision, speech or language application in ML), but rather to collect evidence in favor or against an hypothesis, or help to inform new hypotheses and experiments based on analysis of experimental data.

The introduction of new analysis tools does not necessarily imply a change in analysis practice: For instance, PCA is a very commonly used tool to study low dimensional subspaces of neural activity (Roweis & Ghahramani, 1999). As data becomes more complex, this approach might fall short of accurately capturing the signals of interest. Yet, analysis practices on large datasets do not necessarily need to change: Non-linear analysis tools can fulfill similar goals as e.g. their linear counter parts – dimensionality reduction, regression, decoding, etc. – although the inner working of these algorithms will be different, and is potentially more complex. For instance, linear regression models can use neural activity, and predict the position or pose of an animal. This allows us to obtain regression coefficients, which can be tested against the null hypothesis of being zero. The fundamental information we obtain from such a model is its goodness of fit (for instance, measured by the explained variance, or R^2), as well as defining if a certain neuron is meaningfully involved in the regression (statistical test for non-zero slope). Hence, the fundamental information from this model is: (1) How well can we reconstruct the observed behavior, given the neural data? (2) Which neurons – under this model and mapping between neurons and behavior – are influencing the behavioral variable in the decoding task?

This is informative, but as we want to scale the number of neurons from 10s or 100s to a 1000s or 10,000s distributed across brain areas, might lack explanatory power. In these cases, we might be interested in either reducing the complexity from the 1000-10,000 dimensional space of activity vectors to a lower number of latents, or replacing the linear analysis with and end-to-end non-linear analysis technique. The fundamental information we would like to obtain from this more complex model does not change: Which latents (and hence, neurons) are influencing the behavioral variable, and how good is the model fit?

How can we enable such analysis on more complex datasets that require more powerful models, and potentially non-linear analysis methods? In Chapter 7, I introduced CEBRA, a non-linear contrastive learning algorithm suitable for self-supervised representation on neural and behavioral data individually, and linking both data modalities with a new variant of supervised contrastive learning.

CEBRA fulfills two goals: First, it is a dimensionality reduction tool which maps high-dimensional recordings into lower dimensional embedding spaces. This use case is most similar to linear algorithms like PCA (Pearson, 1901; Roweis & Ghahramani,

1999) or FastICA (Hyvärinen & Oja, 2000). However, CEBRA as a non-linear algorithm is substantially more flexible in how the data is reduced. Yet, compared to other non-linear data visualization tools like tSNE (Van der Maaten & Hinton, 2008) or UMAP (McInnes et al., 2018), the algorithm does not remove information from the original signal to perform the reduction: All relevant latent information from the original data is represented in the lower dimensional embedding space. In addition, its (linear) identifiability guarantees yield consistent results across runs or subjects[7]. In the context of large scale data, dimensionality reduction is most useful as it allows to broaden the analysis scope of existing data analysis workflows: Consider analyses that were previously executed on smaller datasets of e.g. 10s or 100s or neurons. For instance, this includes computing tuning curves, decoding behavioral variables with regression models, or visualizing principal components of the data. With non-linear methods, the same analyses can now be done on orders of magnitude larger datasets by first reducing these datasets to a lower dimensional space, and then performing similar downstream analysis on the resulting latent dimensions.

In Chapter 8, I built on CEBRA and investigate options for computing attribution maps with identifiability guarantees. Inferring information about how input neurons to the networks are involved in the computation of a particular latent is useful to design interventions for follow-up experiments. It is also interesting to view it as a non-linear extension of statistical testing of each neurons' individual contribution to a regression variable we could perform with a linear model: For instance, in regression models, of the form $y = x_1 w_1 + \cdots + x_n w_n + b + \epsilon$ we are interested in testing the coefficients w_i of each neuron for a significant deviation from $w_i = 0$, our null hypothesis. In a non-linear neural net with many layers, such a test is more involved – nevertheless, we would like to address very similar questions, and quantify the contribution of each neuron even for a more complicated non-linear interaction with a behavioral variable.

We want to perform statistical inference on the role of individual neurons, and their computational function within their circuit. At the same time, we want to acknowledge the complexity and non-linear nature of these systems. As in representation learning, such feature attribution offer important insights about a biological system. Leveraging non-linear representation learning tools in this space allows us to obtain individual neuron's roles in computations relevant to a particular latent dimension. In this context, wrong attribution can at least yield false positive results, and at worst generate wrong scientific conclusions. As with CEBRA, it is therefore necessary to theoretically and empirically verify guarantees for such tools, and assess their limitations.

Linking behavior to its neural origins is (one of) the key quests in neuroscience, and doing so under naturalistic conditions requires tools that can handle the complexity of the real world. I advocated for the use of contrastive representation learning algorithms

[7]Crucially, such guarantees are testable on synthetic and real datasets. We can sample data according to our algorithmic assumptions, and empirically test identifiability, also under violations of our assumptions. On real data, identifiability translates into reproducible results as we repeat an analysis on the same subject, or consider different subjects undergoing the same experimental setup.

which do not rely on the requirement of accurately modeling the data through a generative objective.

A current frontier in machine learning is the availability of generative modeling algorithms based on the transformer architecture (Vaswani et al., 2017) and generative pre-training (Radford et al., 2019). It is a compelling example, and alternative, to the use of contrastive learning in the algorithms presented here. It will be certainly an interesting avenue to explore such pre-training techniques in the context of neural data, potentially also in the context of integrating data from various modalities (Bachmann et al., 2022), and recording sessions.

Future work should continue to transfer the best available machine learning approaches to neuroscientific inference. An important distinction between a machine learning application in vision, speech, or language compared to a scientific application is the strong focus on guarantees for interpretability, reliability and robustness for a scientific application. Such tools should be viewed as statistical methods, and less as data processing tools, which I discussed before.

Summary

I contributed to two algorithmic areas useful for scientific inference: Firstly, to the development of more robust and adaptable data processing systems in the context of computer vision. I discussed the new benchmark paradigm of evaluating robustness under partial or full availability of the test dataset, and studied the implications of this evaluation paradigm. With these benchmarks, my colleagues and I could demonstrate the effectiveness of gradient-free and gradient-based adaptation methods at deployment time, and also hint towards the open problem of stable, continual adaptation over long timescales. This work contributed to a now de-facto standard evaluation paradigm that allows a more holistic performance evaluation of computer vision models in a variety of deployment scenarios. While the presented work was carried out on benchmark datasets common in the computer vision community, I showed how these insights can be transferred to applications in pose estimation and medical image analysis in multiple follow-up works.

Secondly, I proposed a new framework for identifiable representation learning on biological data by leveraging contrastive learning. This work draws a connection between contrastive learning in computer vision and the problem of non-linear ICA, and leverages these machine learning insights to arrive at a new algorithmic formulation suitable for scientific inference. Within a single algorithmic framework, my colleagues and I demonstrated state-of-the art performance in computing consistent embedding spaces for scientific data analysis, robust decoding of behavioral and sensory data from brain areas across species and recording modalities, and the combination of discovery- and hypothesis-driven data analysis. This work in data analysis will ease the transition to analyzing the increasingly large and complex datasets generated in the life sciences, aid with discovery-driven analysis of these large datasets, and potentially reduce the

need for animal experimentation by more efficiently combining and exploiting the datasets within a scientific experiment.

Outlook

Open data in neuroscience research: New opportunities for theoretical, computational, and experimental collaboration

I distinguished between the data collection, processing and analysis phases. Oftentimes, methods and applications for all three phases of scientific experimentation are developed and executed under a 'single PI' model in many research labs. Historically, a final publication would often only share the processed data that went into the final analysis, and not the raw dataset in its state *before* the data processing phase. Sharing such pre-processed datasets has its role in ensuring reproducibility, but does not allow to perform explorative re-analysis of a dataset with a different hypothesis or purpose in mind than the original study.

The lack of openly shared raw data might also be considered problematic from an ethical standpoint, as reference experiments might be conducted multiple times, unpublished negative results get replicated by multiple labs, etc. – in contrast, access to raw data would allow to truly build on top of existing knowledge, and allow to re-use existing raw data for use cases not originally envisioned or – especially if re-analysis takes place much later – methodologically not possible at the time of the primary publication. Today, we see shifts from this model, and initiatives in different countries work towards encouraging or enforcing data sharing. This includes the data sharing mandate adopted by the NIH[8] in the US or the 3R (replacement, reduction and refinement of animal experimentation) initiative in Switzerland[9]. Efforts like the OpenScope or IBL publish large scale raw datasets, which allow re-analysis beyond the primary publication, modeled after research collaborations in physics and other sciences, where data collection is performed by a central entity, and then distributed. Finally, platforms like Dandi[10] facilitate data sharing without a cap on recording sizes[11].

Data sharing is effectively a way to encourage a "division of labor" in neuroscience: Theoreticians can propose experiments, centralized Observatories collect such experiments and execute especially promising and relevant ideas with optimized technology, and computationalists benefit from the availability of high quality data. This more distributed effort to neuroscience has the potential of increasing the pace of the field, increase interdisciplinary work. Machine learning worked under a similar model for several years now: Datasets and technical artifacts were shared openly in many

[8]NIH Policy for Data Management and Sharing, effective January 25, 2023, https://grants.nih.gov/grants/guide/notice-files/NOT-OD-21-013.html

[9]https://swiss3rcc.org/

[10]https://dandiarchive.org/

[11]As of October 10, 2023, the largest structural dataset on Dandi is Dandiset 118, "Light sheet imaging of the human brain" at 364TB, and the largest functional dataset is the "IBL Brain Wide Map" at 34TB.

instances, giving labs with smaller resources[12] the opportunity to work on topics like robustness, new benchmarks, analysis, and applications.

Does science need foundation models?

In recent years, "foundation models" (Bommasani et al., 2021) became an important topic in the machine learning community. Foundation models are trained on large amounts of data, and can then be flexibly adapted to a variety of tasks in a data-efficient way. They are interesting in the context of neuroscience in (at least) three very different ways.

Firstly, as an object of study themselves: Without dismissing previous leaps in the development of models in speech, vision, and text, current foundation models might be the first examples of "sufficiently intelligent" systems that are "worth studying", similar to the model organisms we use in neuroscience research. Foundation models are commonly built on top of the transformer architecture, a very versatile structure that allows to solves problems in vision, speech and language using the same underlying model architecture. The transformer architecture very much resembles currently theories of how cortex is structured and replicated (Jones, 2000; Mathis, 2023; Mountcastle, 1957) – and even though the substrate for computation, and most likely also the inner structure of these models is completely different from the brain, their property of a common scalable architecture is a very compelling result that has not been observed before. So, what can we learn from foundation models? Machine learning's primary focus is typically less on model understanding, and more on actually replicating intelligence in computers. As neuroscientists, it is an interesting challenge to understand how large foundation models are actually working. They are interesting as a subject of study, because they are fully observable: Experiments are cheap to run, and we are able to observe any single neuron in the whole model. In this vein, they could also be interesting testbeds for new analysis methods. Within our framework of collection, processing and analysis, this effectively makes these models a subject of study, and hence of data collection, the first step.

Secondly, as a tool for data processing: As we saw in the first chapters, larger models tend to perform favorably in terms of their robustness properties on real world downstream tasks like pose estimation and other computer vision tasks. Today, a plethora of methods exist for pre-training models in the visual and other domains. Using a large model often comes with the disadvantage of slower inference times, but huge gains in their performance. As such, available computer vision foundation models will have tremendous applications in object and animal detection, segmentation and tracking, future approaches to pose estimation, and other forms for representing behavior.

Finally, to aid data analysis: Especially models that work on the interface of text and program code, foundation model could be regarded as tools to be integrated into

[12] in machine learning, this translates to compute resources

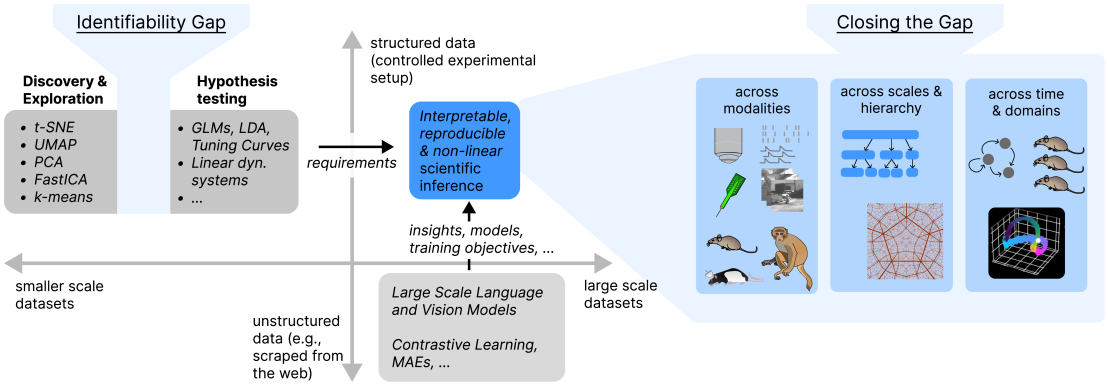


Figure 9.1: We observe an identifiability gap between current methods (upper left) for explorative analysis (unsupervised) and hypothesis testing (supervised). These methods operate in the regime of highly structured and controlled experimental data, but at limited scale. In contrast, modern machine learning methods (lower right) are trained on vast amounts of unstructured data, but are not suited directly for scientific inference. I propose to close the identifiability gap by non-linear algorithms inspired by modern ML methods, that fulfill and go beyond the desiderata of classical inference algorithms when it comes to robustness and interpretability. Some figure elements taken from https://scidraw.io.

data analysis pipelines, to perform suggestions on data pre-processing and ways to setup analysis. Emerging approaches like AmadeusGPT (Ye et al., 2023b) use language models to aid at designing behavioral analyses, and lower the entry barrier for advanced data analysis. Other publicly available machine learning tools based on foundation models like GPT-4 will continue to speed up the turnaround time for preparing and improving data analysis tasks.

The identifiability gap: Frontiers in representation learning

While this work made progress towards using non-linear algorithms for analyzing various neuroscience datasets, a lot of open problems in the design of such algorithms remain. Here I will discuss two: The extension of algorithmic frameworks as presented in Chapters 7 and 8, and the possibilities to leverage other recent developments in machine learning research to build better models for neuroscience.

It remains an open problem to identify full dynamical systems based on interventional and observational data. How can we extend identifiability guarantees towards such setups, that ideally also capture hierarchical structure of signals, and the interaction of data from many different modalities? A related interesting avenue lies in building "digital twins" of large, multi-modal datasets. Recordings are often not perfectly paired, number of neurons or body keypoints can vary across experimental sessions. Such datasets are also common in machine learning, and pre-training approaches based on masking can naturally deal with such inconsistencies in a dataset.

Both approaches work towards closing the "Identifiability Gap" between data-driven science, and the hypothesis-driven scientific method (Figure 9.1). A worthwhile

future avenue is to leverage the insights from machine learning to build algorithms fulfilling the stricter requirements for scientific inference. This approach ideally yields more powerful, unified ways to process, model, and analyzing data, allow to close the loop, and enable the analysis of more complex and larger datasets. Such unified frameworks for analysis, and the continued co-development of increasingly involved experimental and computational techniques will help us to gradually unpack the fundamental question posed at the very beginning of this work: *How does the brain generate adaptive, intelligent behavior?*

A

Improving robustness against common corruptions by covariate shift adaptation

Distances and divergences for quantifying domain shift

Besides analyzing the performance drop when evaluating a model using source statistics on a target dataset, we consider the mismatch in model statistics directly. We first take an ImageNet trained model and adapt it to each of the 95 conditions in IN-C. To obtain a more exact estimate of the true statistics, we split the model into multiple stages with only few BN layers per stage and apply the following simple algorithm[1]:

- Start with image inputs $\mathbf{z}_n^0 \leftarrow \mathbf{x}_n$ from the validation set to adapt to, for each $n \in [50000]$.
- Split the model into multiple stages, $h(\mathbf{x}) = (f_m \circ \cdots \circ f_1)(\mathbf{x})$, where each module f_i can potentially contain one or multiple BN layers. We denote the number of BN layers in the i-th module as b_i.
- For each stage $i \in [m]$, repeat b_i times: $\mathbf{z}_n^i \leftarrow f_i(\mathbf{z}_n^{i-1})$ for each n, and update the BN statistics in module $f_i(\mathbf{z}_n^{i-1})$.
- Return h with adapted statistics.

[1]Note that for simplicity, we do not reset the statistics of the remaining $(b_i - i)$ BN layers. This could potentially be adapted in future work.

Using this scheme, we get source statistics μ_s and Σ_s for each layer and μ_t and Σ_t for each layer and corruption. In total, we get 96 different collections of statistics across network layers (for IN and the 95 conditions in IN-C). For simplicity, we will not further index the statistics. Note that all covariance matrices considered here are diagonal, which is a further simplification. We expect that our domain shift estimates could be improved by considering the full covariance matrices.

In the following, we will introduce three possible distances and divergences which can be applied between source and target statistics to quantify the effect of common corruptions induced covariate shift. We consider the Wasserstein distance, a normalized version of the Wasserstein distance, and the Jeffrey divergence.

The Wasserstein distance

Given a baseline ResNet-50 model with source statistics μ_s, Σ_s on IN, the Wasserstein distance (cf. Villani, 2008) between the train and test distribution with statistics μ_t, Σ_t is given as

$$W_2(p_s, p_t)^2 = \|\boldsymbol{\mu}_s - \boldsymbol{\mu}_t\|_2^2 + \operatorname{tr}\left(\boldsymbol{\Sigma}_s + \boldsymbol{\Sigma}_t - 2\left(\boldsymbol{\Sigma}_t^{1/2}\boldsymbol{\Sigma}_s\boldsymbol{\Sigma}_t^{1/2}\right)^{1/2}\right). \tag{A.1}$$

The source-normalized Wasserstein distance

When estimated for multiple layers across the network, the Wasserstein distance between source and target depends on the overall magnitude of the statistics. Practically, this means the metric is dominated by features with large magnitude (e.g. in the first layer of a neural network, which receives larger inputs).

To mitigate this issue, we normalize both statistics with the source statistics and define the normalized Wasserstein distance as

$$\widetilde{W}_2^2 = W_2^2\left(\boldsymbol{\Sigma}_s^{-1/2}\boldsymbol{\mu}_s, \mathbf{I}, \boldsymbol{\Sigma}_s^{-1/2}\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_s^{-1}\boldsymbol{\Sigma}_t\right) \tag{A.2}$$

$$= \operatorname{Tr}\left(\mathbf{I} + \boldsymbol{\Sigma}_t\boldsymbol{\Sigma}_s^{-1} - 2\boldsymbol{\Sigma}_t^{1/2}\boldsymbol{\Sigma}_s^{-1/2}\right) + (\boldsymbol{\mu}_t - \boldsymbol{\mu}_s)^T\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_s). \tag{A.3}$$

In the uni-variate case, the normalized Wasserstein distance $\widetilde{W}_2^2$ is equal to the Wasserstein distance W_2^2 between source and target statistics divided by σ_s^2:

$$\widetilde{W}_2^2 = W_2^2\left(\frac{\mu_s}{\sigma_s}, 1, \frac{\mu_t}{\sigma_s}, \frac{\sigma_t^2}{\sigma_s^2}\right) = 1 + \frac{\sigma_t^2}{\sigma_s^2} - 2\frac{\sigma_t}{\sigma_s} + \frac{(\mu_t - \mu_s)^2}{\sigma_s^2} = \frac{1}{\sigma_s^2}W_2^2(\mu_s, \sigma_s^2, \mu_t, \sigma_t^2). \tag{A.4}$$

The Jeffrey divergence

The Jeffrey divergence $J(p_s, p_t)$ between source distribution p_s and target distribution p_t is the symmetrized version of the Kullback-Leibler divergence D_{KL}:

$$J(p_s, p_t) = \frac{1}{2}\left(D_{KL}(p_s \| p_t) + D_{KL}(p_t \| p_s)\right) \tag{A.5}$$

The Kullback-Leibler divergence between the D-dimensional multivariate normal source and target distributions is defined as

$$D_{KL}(\mathcal{N}_t \| \mathcal{N}_s) = \frac{1}{2}\left(\mathrm{Tr}\left(\boldsymbol{\Sigma}_s^{-1}\boldsymbol{\Sigma}_t\right) + (\boldsymbol{\mu}_s - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_s^{-1}(\boldsymbol{\mu}_s - \boldsymbol{\mu}_t) - D + \ln\left(\frac{\det \boldsymbol{\Sigma}_s}{\det \boldsymbol{\Sigma}_t}\right)\right). \tag{A.6}$$

The Jeffrey divergence between the D-dimensional multivariate normal source and target distributions then follows as

$$J(\mathcal{N}_t, \mathcal{N}_s) = \frac{1}{4}\left(\mathrm{Tr}\left(\boldsymbol{\Sigma}_s^{-1}\boldsymbol{\Sigma}_t\right) + \mathrm{Tr}\left(\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}_s\right) + (\boldsymbol{\mu}_s - \boldsymbol{\mu}_t)^\top \left(\boldsymbol{\Sigma}_s^{-1} + \boldsymbol{\Sigma}_t^{-1}\right)(\boldsymbol{\mu}_s - \boldsymbol{\mu}_t) - 2D\right). \tag{A.7}$$

Summary statistics and quantification of covariate shift between different IN-C conditions

Given the 95 distances/divergences between the baseline (IN) statistics and 95 IN-C conditions, we first perform a layer-wise analysis of the statistics and depict the results in Figure A.1. The unnormalized Wasserstein distance is sensitive to the magnitude of the source statistics and hence differs qualitatively from the results on the normalized Wasserstein distance and Jeffrey Divergence. We appreciate that the most notable difference between source and target domains is visible in the ResNet-50 downsampling layers. All three metrics suggest that the shift is mainly present in the first and final layers of the network, supporting the hypothesis that within the common corruption dataset, we have both superficial covariate shift which can be corrected by simple means (such as brightness or contrast variations) in the first layers, and also more "high-level" domain shifts which can only be corrected in the later layers of the network.

In Figure A.2, we more closely analyze this relationship for different common corruptions. We can generally appreciate the increased measures as the corruption severity increases.

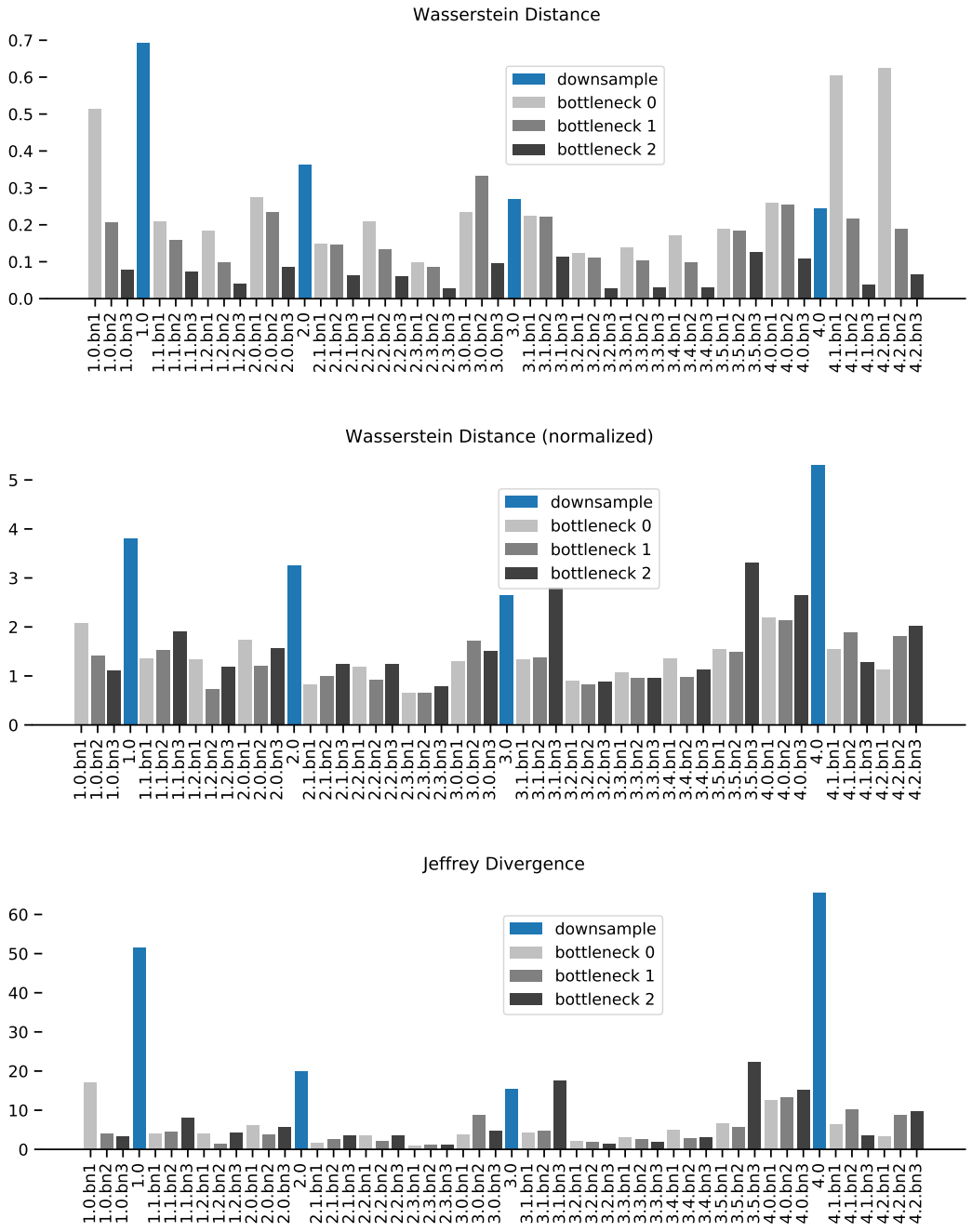


Figure A.1: Wasserstein distance, normalized Wasserstein distance and Jeffrey divergence estimated among source and target statistics between different network layers. We report the respective metric w.r.t. to the difference between baseline (IN) and target (IN-C) statistics and show the value averaged across all corruptions. We note that for a ResNet-50 model, downsampling layers contribute most to the overall error.

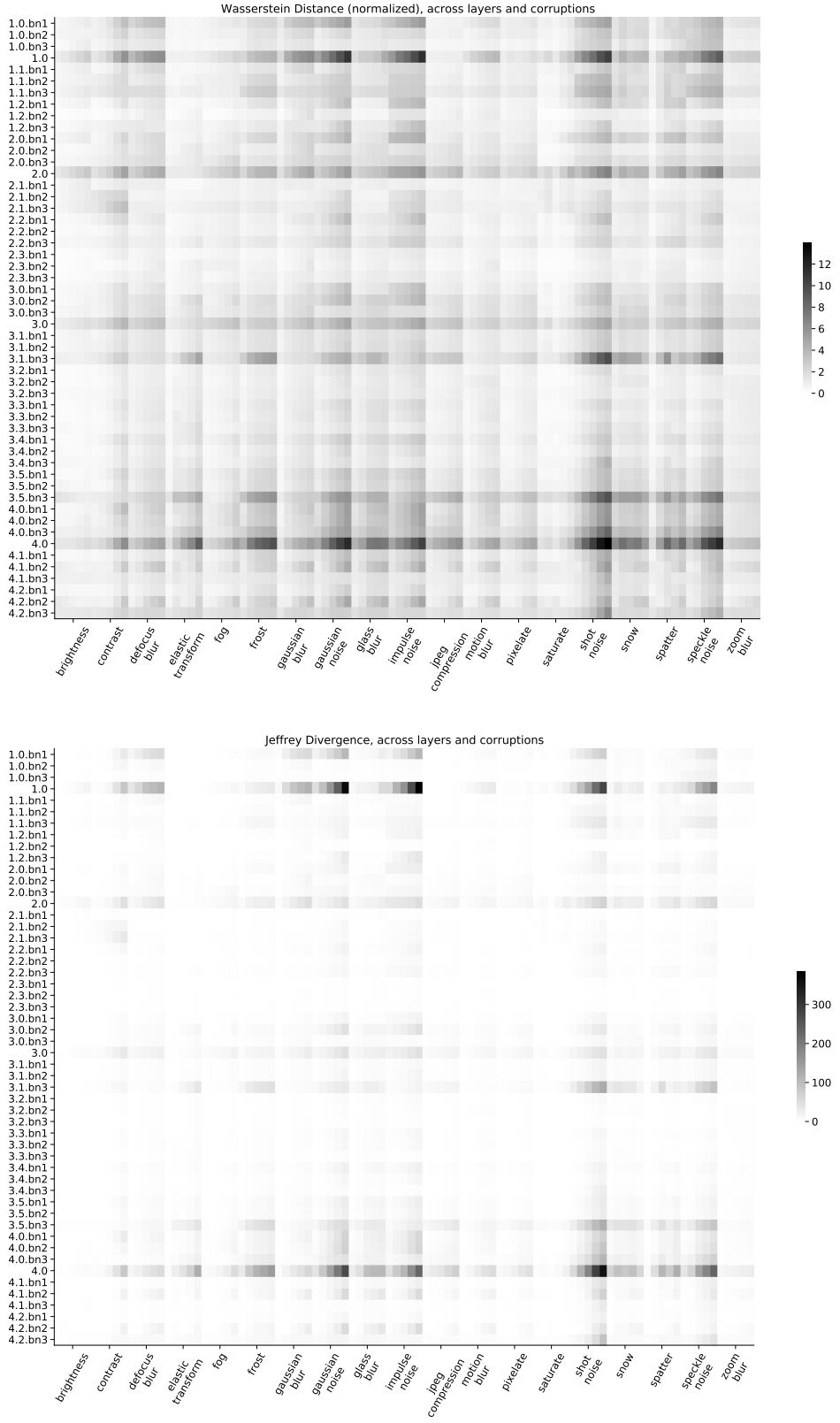


Figure A.2: Normalized Wasserstein distance and Jeffrey divergence across corruptions and layers in a ResNet-50.

Notes on the experimental setup

Practical considerations for implementing the method

Our method is conceptually very easy to implement. We generally recommend to first explore the easier variant of the algorithm where $N = 0$, i.e., no source statistics are used. As shown in our experiments, this setting works well if 100 or more target samples are available.

In this case, implementing the method boils down to enabling the training mode for all BN layers across the network. We will discuss this option along with two variants important for application to practical problems: Using exponential moving averaging (EMA) to collect target statistics across multiple batches, and using the source statistics as a prior.

Example implementation in PyTorch and caveats We encourage authors of robust models to always evaluate their models, and in particular baseline algorithms on both the train and test set statistics. Implementation in both PyTorch, Tensorflow and other machine learning libraries is straightforward and adds only minimal overhead. For PyTorch, adaptation is possible by simply adding

```
def use_test_statistics(module):
  if isinstance(module, nn._BatchNorm):
    module.train()
model.eval()
model.apply(use_test_statistics)
```

before starting a model evaluation. For the adaptation to a full dataset, we provide a reference implementation with the source code release of this paper. Also, in contrast to the convention of not shuffling examples during test time, *make sure to enable dataset shuffling also during test time* in order to compute the correct statistics marginalized over class assignment.

Exponential moving averaging In practice, it might be beneficial to keep track of samples already encountered and use a running mean and variance on the test set to normalize new samples. We can confirm that this technique closely matches the full-dataset adaptation case even when evaluating with batch size 1 and is well suited for settings with less powerful hardware, or in general settings where access to the full batch of samples is not possible. Variants of this technique include the adaptation of the decay factor to discard statistics of samples encountered in the past (e.g. when the data domain slowly drifts over time).

Notes on models

Note that we only re-evaluate existing model checkpoints, and hence do not perform any hyperparameter tuning or adaptations to model training except for selecting the

pseudo batchsize N for the source domain. Depending on the batch size and the architecture, model evaluations are done on one to eight Nvidia RTX 2080 GPUs (i.e., using 12 to 96 GB of memory) or up to four Nvidia V100 GPUs (128 GB of memory). Since we merely re-evaluate trained models, it is also possible to work on less powerful hardware with less memory. In these cases, the aggregation of batch normalization statistics has to be done across several batches using a variant of EMA.

Hyperparameter tuning

Our method is generally parameter-free if only target statistics should be considered for normalization. This approach is generally preferred for larger batch sizes n and should also be adapted in practice when a sufficient amount of samples is available. For tuning N, we consider the pre-defined holdout corruptions in IN-C, including speckle noise, saturation, Gaussian blur and spatter using a grid search across different values for N.

Notes on datasets

In the main paper, we have used several datasets and provide more relevant information here:

ImageNet-C (IN-C) For the evaluation on IN-C, we use the JPEG compressed images from github.com/hendrycks/robustness as is advised by the authors to ensure reproducibility. We note that Ford et al. (2019) report a decrease in performance when the compressed JPEG files are used as opposed to applying the corruptions directly in memory without compression artefacts.

ObjectNet (ON) We find that there are 9 classes with multiple possible mappings from ON to IN (see the list in Table A.1); we discard these classes in our evaluation. Models trained on IN experience a large performance drop on the order of 40–45% when tested on ON. ON is an interesting test case for unsupervised domain adaptation since IN and ON are likely sampled from different distributions. ON intentionally shows objects from new viewpoints on new backgrounds.

ImageNet-V2 (IN-V2) There are three test sets in IN-V2 that differ in *selection frequencies* of the MTurk workers. The selection frequency is given by the fraction of MTurk workers who selected an image for its target class. For the "MatchedFrequency" dataset, images were sampled according to the estimated selection frequency of sampling of the original IN validation dataset. For the "Thresholdo.7" variant of IN-V2, images were sampled with a selection frequency of at least 0.7. The "TopImages" was sampled from images with the highest selection frequency. Although all three test sets were sampled from the same Flickr candidate pool and were labeled correctly and selected by more than

70% of MTurk workers, the model accuracies on these datasets vary by 14%. The authors observe a systematic accuracy drop when comparing model performance on the original IN validation set and IN-V2 and attribute it to the distribution gap between their datasets and the original IN dataset. They quantify the distribution gap by how much the change from the original distribution to the new distribution affects the considered model. Engstrom et al. analyze the creation process of IN-V2 and identify statistical bias resulting from noisy readings of the selection frequency statistic as a main source of dropping performance (Engstrom et al., 2020). After correcting the bias, (Engstrom et al., 2020) find that the accuracy drop between IN and IN-V2 measures only 3.6% ± 1.5% of the original 11.7% ± 1.0%.

Table A.1: Mapping between 9 ambiguous ON classes and the possible correspondences in IN. Different IN classes are separated with a semicolon.

ON class	IN classes
wheel	wheel; paddlewheel, paddle wheel
helmet	football helmet; crash helmet
chair	barber chair; folding chair; rocking chair, rocker
still_camera	Polaroid camera, Polaroid Land camera; reflex camera
alarm_clock	analog clock; digital clock
tie	bow tie, bow-tie, bowtie; Windsor tie
pen	ballpoint, ballpoint pen, ballpen, Biro; quill, quill pen; fountain pen
bicycle	mountain bike, all-terrain bike, off-roader; bicycle-built-for-two, tandem bicycle, tandem
skirt	hoopskirt, crinoline; miniskirt, mini; overskirt

Overview of models in torchvision

In Table A.2, we provide a list of the models we evaluate in the main paper, along with numbers of trainable parameters and BN parameters. Note that the fraction of BN parameters is at most at 1% compared to all trainable parameters in all considered models.

Baseline corruption errors

In Table A.3, we report the scores used for converting top-1 error into the mean corruption error (mCE) metric proposed by Hendrycks and Dietterich (2019a).

Software stack

We use various open source software packages for our experiments, most notably Docker (Merkel, 2014), scipy and numpy (Virtanen et al., 2020), GNU parallel (Tange, 2011), Tensorflow (Abadi et al., 2016), PyTorch (Paszke et al., 2017) and torchvision (Marcel & Rodriguez, 2010).

Table A.2: Overview of different models with parameter counts. We show the total number of BN parameters, which is a sum of affine parameters.

Model	Parameter Count	BN Parameters	Fraction (%)
densenet121	7.98×10^6	8.36×10^4	0.010
densenet161	2.87×10^7	2.20×10^5	0.008
densenet169	1.41×10^7	1.58×10^5	0.011
densenet201	2.00×10^7	2.29×10^5	0.011
googlenet	1.30×10^7	1.51×10^4	0.001
inception-v3	2.72×10^7	3.62×10^4	0.001
mnasneto-5	2.22×10^6	2.06×10^4	0.009
mnasneto-75	3.17×10^6	2.98×10^4	0.009
mnasnet1-0	4.38×10^6	3.79×10^4	0.009
mnasnet1-3	6.28×10^6	4.88×10^4	0.008
mobilenet-v2	3.50×10^6	3.41×10^4	0.010
resnet101	4.45×10^7	1.05×10^5	0.002
resnet152	6.02×10^7	1.51×10^5	0.003
resnet18	1.17×10^7	9.60×10^3	0.001
resnet34	2.18×10^7	1.70×10^4	0.001
resnet50	2.56×10^7	5.31×10^4	0.002
resnext101-32x8d	8.88×10^7	2.03×10^5	0.002
shufflenet-v2-x0-5	1.37×10^6	7.95×10^3	0.006
shufflenet-v2-x1-0	2.28×10^6	1.62×10^4	0.007
shufflenet-v2-x1-5	3.50×10^6	2.34×10^4	0.007
shufflenet-v2-x2-0	7.39×10^6	3.37×10^4	0.005
vgg11-bn	1.33×10^8	5.50×10^3	4.142×10^{-5}
vgg13-bn	1.33×10^8	5.89×10^3	4.425×10^{-5}
vgg16-bn	1.38×10^8	8.45×10^3	6.106×10^{-5}
vgg19-bn	1.44×10^8	1.10×10^4	7.662×10^{-5}
wide-resnet101-2	1.27×10^8	1.38×10^5	0.001
wide-resnet50-2	6.89×10^7	6.82×10^4	0.001

Table A.3: AlexNet top1 errors on ImageNet-C

Category	Corruption	top1 error
Noise	Gaussian Noise	0.886428
	Shot Noise	0.894468
	Impulse Noise	0.922640
Blur	Defocus Blur	0.819880
	Glass Blur	0.826268
	Motion Blur	0.785948
	Zoom Blur	0.798360
Weather	Snow	0.866816
	Frost	0.826572
	Fog	0.819324
	Brightness	0.564592
	Contrast	0.853204
Digital	Elastic Transform	0.646056
	Pixelate	0.717840
	JPEG Compression	0.606500
Hold-out Noise	Speckle Noise	0.845388
Hold-out Digital	Saturate	0.658248
Hold-out Blur	Gaussian Blur	0.787108
Hold-out Weather	Spatter	0.717512

Table A.4: After converting the checkpoints from TensorFlow to Pytorch, we notice a slight degradation in performance on the IN val set.

IN val top-1 accuracy in %.

Model	TF	PyTorch
SimCLRv2 ResNet50	76.3	75.6
SimCLRv2 ResNet101	78.2	77.5
SimCLRv2 ResNet152	79.3	78.6

Table A.5: Adaptation improves the performance of the ResNet50 and the ResNet101 model but hurts the performance of the ResNet152 model.

ImageNet-C (n=4096), mCE.

Model, adaptation:	base	adapt	Δ
SimCLRv2 ResNet50	72.4	68.0	-4.2
SimCLRv2 ResNet101	66.6	65.1	-0.9
SimCLRv2 ResNet152	63.7	64.2	+0.5

Additional results

Performance of SimCLRv2 models

We evaluate the performance of 3 models from the SimCLRv2 framework with and without batchnorm adaptation. We test a ResNet50, a ResNet101 and a ResNet152, finetuned on 100% of IN training data. Since our code-base is in PyTorch, we use the Pytorch-SimCLR-Converter (Lin, 2020 (accessed October 21, 2020)) to convert the provided checkpoints from Tensorflow to PyTorch. We notice a slight decline in performance when comparing the top-1 accuracy on the IN validation set, see Table A.4. For preprocessing, we disable the usual PyTorch normalization and use the PIL.Image.BICUBIC interpolation for resizing because this interpolation is used in the TensorFlow code (instead of the default PIL.Image.BILINEAR in PyTorch).

The BN adaptation results for the converted models are shown in Table A.5. Adaptation improves the performance of the ResNet50 and the ResNet101 model, but hurts the performance of the ResNet152 model.

Relationship between parameter count and IN-C improvements

In addition to Fig. 3 in the main paper, we show the relationship between parameter count and IN-C mCE. In general, we see that the parameter counts correlates with corruption robustness since larger models have smaller mCE values.

Per-corruption results on IN-C

We provide more detailed results on the individual corruptions of IN-C for the most important models considered in our study in Fig. A.4. The results are shown for models where the BN parameters are adapted on the full test sets. The adaptation consistently improves the error rates on all corruptions for both vanilla and AugMix.

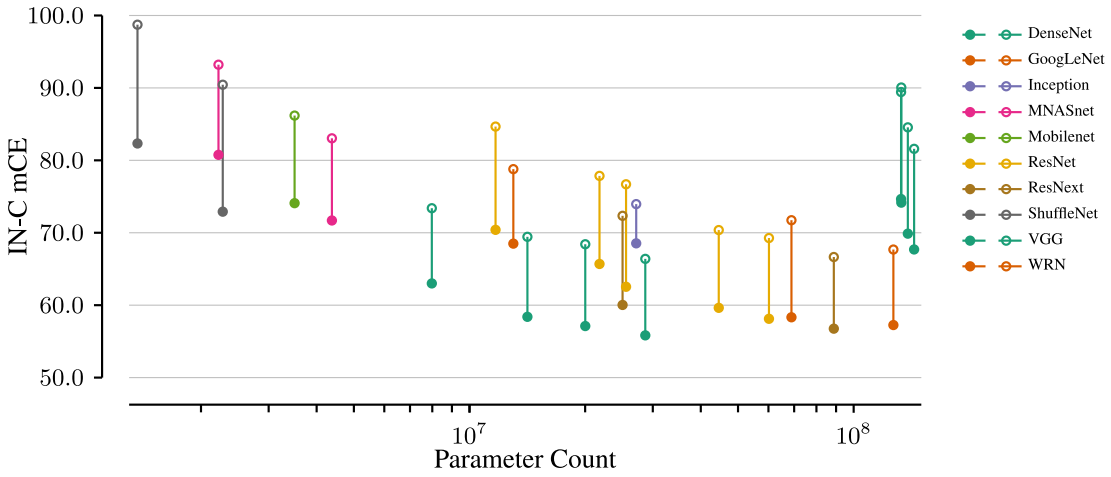


Figure A.3: Adaptation (•) improves baseline (○) mCE across all 25 model architectures in the `torchvision` library, often on the order of 10% points. Best viewed in color.

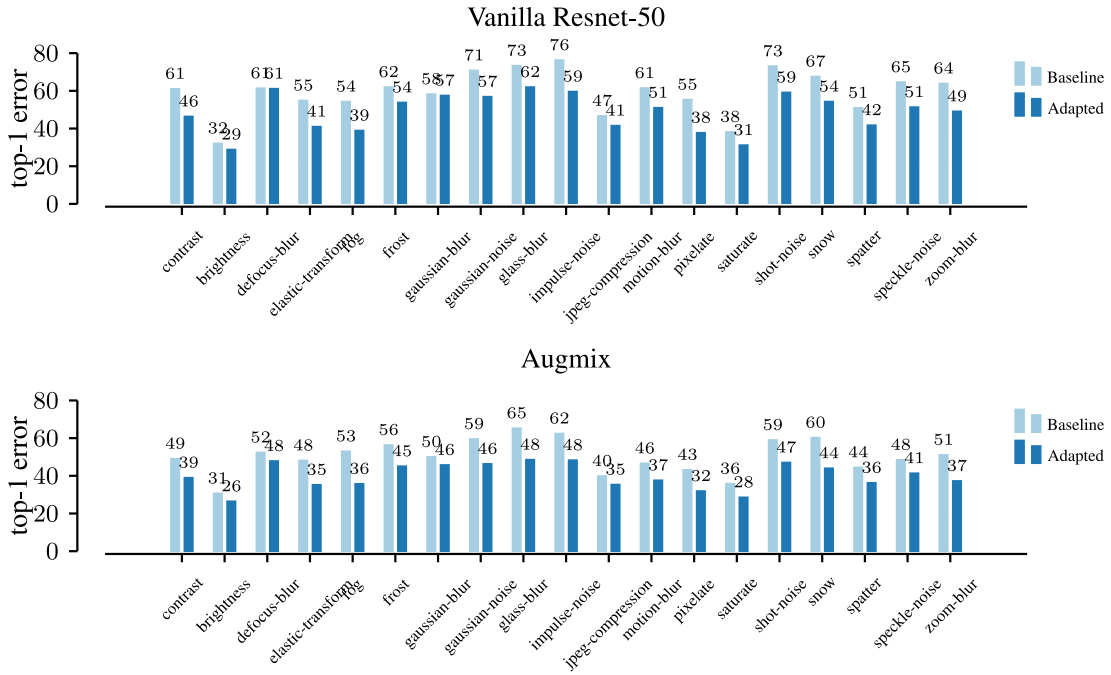


Figure A.4: Results on the individual corruptions of IN-C for the vanilla trained ResNet-50 and the AugMix model with and without adaptation. Adaptation reduces the error on all corruptions.

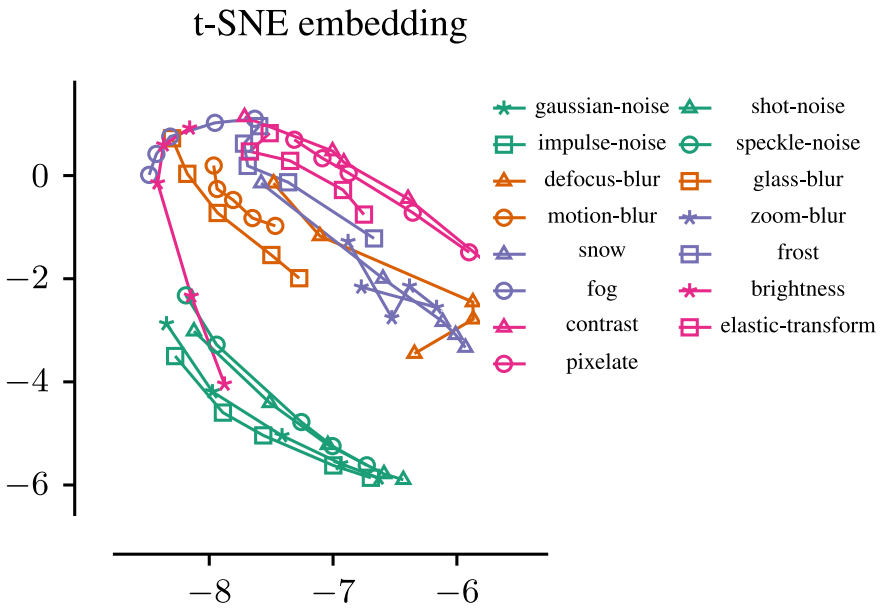


Figure A.5: tSNE embeddings of the Wasserstein distances between BN statistics adapted on the different corruptions. This plot shows evidence on the similarities between different corruption types.

Qualitative analysis of similarities between common corruptions

In this analysis, we compute a tSNE embedding of the Wasserstein distances between the adapted models and the non-adapted model from Fig. 2.3 of the main paper. The results are displayed in Fig. A.5. We observe that the different corruption categories indicated by the different colors are grouped together except for the 'digital' category (pink). This visualization shows that corruption categories mostly induce similar shifts in the BN parameters. This might be an explanation why training a model on Gaussian noise generalizes so well to other noise types as has been observed by Rusak et al. (2020): By training on Gaussian noise, the BN statistics are adapted to the Gaussian noise corruption and from Fig. A.5, we observe that these statistics are similar to the BN statistics of other noises.

Error prediction based on the Wasserstein distance

In Fig. 2.3(i), we observe that the relationship between the Wasserstein distance and the top-1 error on IN-C is strikingly linear in the considered range of the Wasserstein distance. Similar corruptions and corruption types (indicated by color) exhibit similar slope, allowing to approximate the expected top-1 error rate without any information about the test domain itself. Using the split of the 19 corruptions into 15 test and 4 holdout corruptions (Hendrycks & Dietterich, 2019a), we compute a linear regression model on the five data points we get for each of the holdout corruptions (corresponding to the five severity levels), and use this model to predict the expected top-1 error rates for the remaining corruptions within the corruption family. This scheme works

particularly for the "well defined" corruption types such as noise and digital (4.14% points absolute mean deviation from the real error. The full results are depicted in Table A.6.

Table A.6: Estimating top-1 error of unseen corruptions within the different corruption classes. We note that especially for well defined corruptions (like noise or digital corruptions), the estimation scheme works well. We follow the categorization originally proposed by Hendrycks and Dietterich (2019a).

	test error			holdout (train) error			model	
	true	pred	$\lvert\Delta\rvert$	true	pred	$\lvert\Delta\rvert$	coef	intercept
Fig. 2.3 (i)								
blur	64.89	54.53	11.04	58.13	58.13	3.24	37.59	-0.70
digital	54.37	51.96	6.97	38.08	38.08	0.60	37.20	6.39
noise	73.29	69.68	5.84	64.51	64.51	0.65	24.66	1.68
weather	53.87	42.92	11.21	50.84	50.84	5.48	25.80	6.33
Fig. 2.3 (ii)								
blur	55.68	53.28	5.65	57.38	57.38	4.01	42.74	-9.51
digital	41.53	39.80	4.14	31.05	31.05	0.34	23.44	11.09
noise	58.43	55.04	4.14	51.24	51.24	1.01	18.13	5.06
weather	43.84	36.16	7.80	41.63	41.63	4.32	17.80	10.91
Fig. 2.3 (iii)								
blur	57.10	69.84	13.43	74.01	74.01	3.96	43.50	5.93
digital	46.16	38.06	12.97	36.22	36.22	10.52	4.94	32.01
noise	93.60	85.84	13.08	81.10	81.10	3.52	22.56	23.65
weather	43.74	36.90	8.98	44.05	44.05	6.20	23.29	3.87

Training details on the models trained with Fixup initialization and GroupNorm

In Table 2.3, we consider IN models trained with GroupNorm and Fixup initialization. For these models, we consider the original reference implementations provided by the authors. We train ResNet-50, ResNet-101 and ResNet-152 models with stochastic gradient descent with momentum (learning rate 0.1, momentum 0.9), with batch size 256 and weight decay 1×10^{-4} for 100 epochs.

We show the full results for considering different choices of N for ResNet-50, Augmix, ANT, ANT+SIN and SIN models and display the result in Fig. A.7. We observe a characteristic shape which we believe can be attributed to the way statistics are estimated. We provide evidence for this view by proposing an analytical model which we discuss in §A.

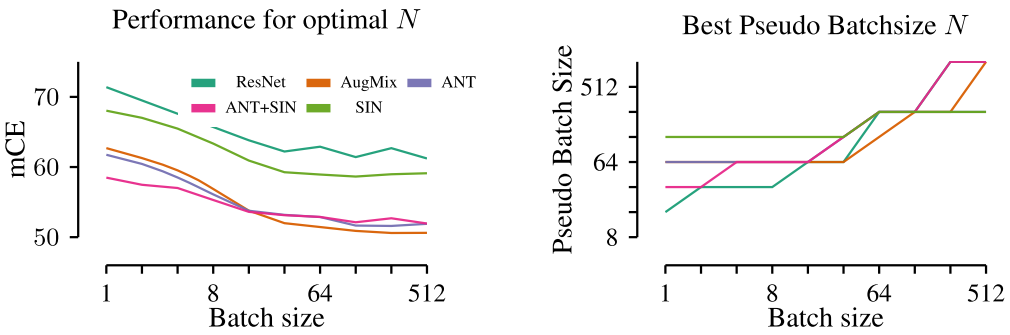


Figure A.6: Left: Performance for all the considered ResNet-50 variants based on the sample batch size. The optimal N is chosen according to the mCE on the holdout corruptions. Right: Best choice for N depending on the input batchsize n. Note that in general for high values n, the model is generally more robust to the choice of N.

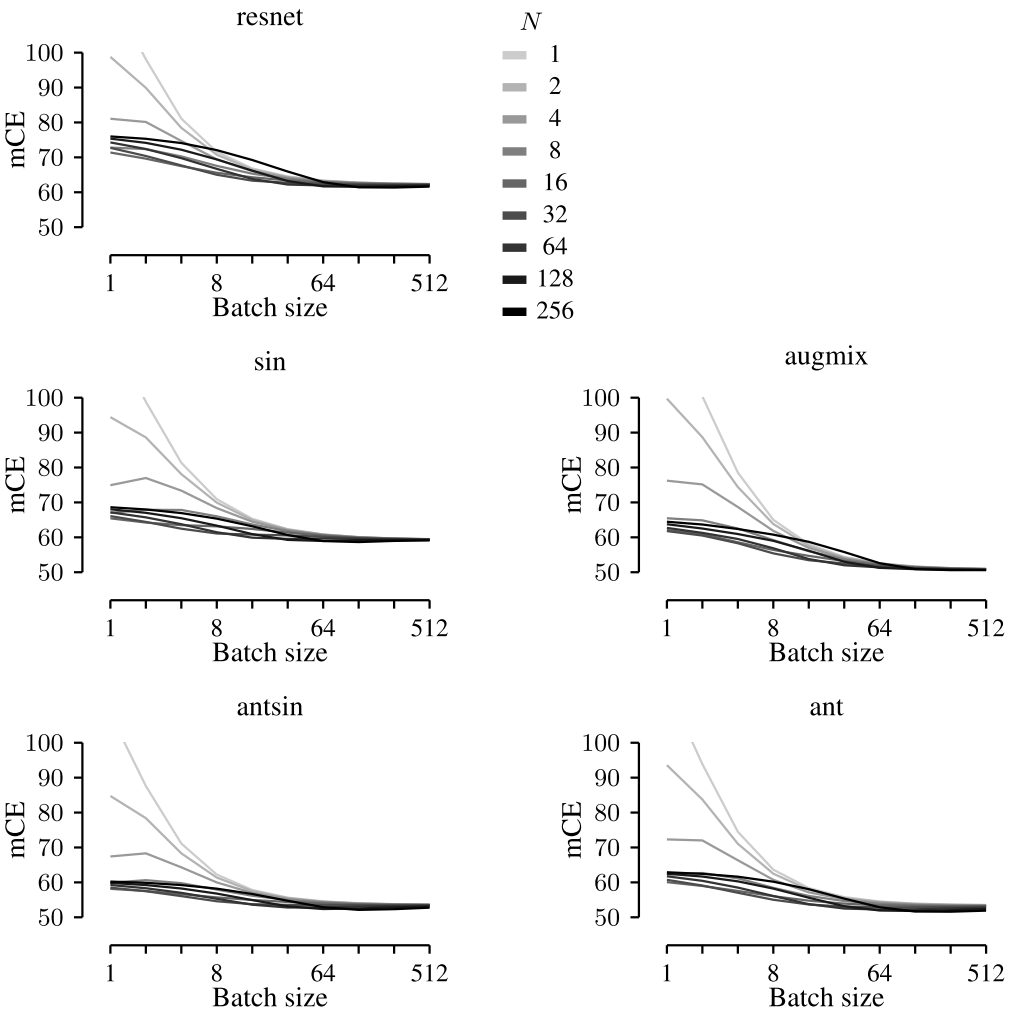


Figure A.7: Effects of batch size n and pseudo batch size N for the various considered models. We report mCE averaged across 15 test corruptions.

Table A.7: Test mCE for various batch sizes (rows) vs. pseudo batch sizes (columns), part 1/3.

ResNet-50	1	2	4	8	16	32	64	128	256
1	117.76	98.78	81.06	72.80	71.39	72.72	74.28	75.36	75.99
2	98.11	89.92	80.13	72.36	69.63	70.39	72.39	74.16	75.32
4	81.10	78.45	74.70	70.27	67.48	67.69	69.77	72.19	74.10
8	71.56	70.74	69.44	67.56	65.60	65.02	66.70	69.41	72.07
16	66.82	66.52	66.06	65.32	64.29	63.32	63.81	66.19	69.24
32	64.51	64.39	64.19	63.87	63.38	62.72	62.21	63.22	65.94
64	63.33	63.28	63.19	63.05	62.81	62.43	61.95	61.68	62.90
128	62.78	62.75	62.69	62.62	62.50	62.29	62.00	61.56	61.42
256	62.51	62.49	62.44	62.41	62.32	62.22	62.01	61.73	61.35
512	62.36	62.36	62.33	62.29	62.26	62.17	62.06	61.90	61.62
AugMix	1	2	4	8	16	32	64	128	256
1	122.56	99.72	76.23	65.46	62.08	61.78	62.70	63.75	64.47
2	100.39	88.69	75.16	64.86	60.93	60.51	61.28	62.52	63.67
4	78.55	74.41	68.69	62.52	58.58	58.30	59.53	60.94	62.39
8	65.02	63.81	61.86	59.21	56.39	55.40	56.87	59.00	60.77
16	58.02	57.55	56.96	56.02	54.69	53.44	53.78	56.15	58.71
32	54.37	54.20	53.99	53.68	53.21	52.50	51.99	53.01	55.78
64	52.55	52.50	52.38	52.24	52.07	51.83	51.39	51.25	52.59
128	51.64	51.60	51.54	51.47	51.38	51.26	51.10	50.88	50.89
256	51.18	51.17	51.12	51.08	51.02	50.95	50.86	50.76	50.60
512	50.96	50.95	50.93	50.90	50.86	50.80	50.72	50.65	50.61

Table A.8: Test mCE for various batch sizes (rows) vs. pseudo batch sizes (columns), part 2/3.

ANT	1	2	4	8	16	32	64	128	256
1	116.10	93.58	72.31	62.28	60.07	60.73	61.75	62.48	62.90
2	93.88	83.74	72.01	62.69	58.97	59.10	60.44	61.67	62.44
4	74.51	71.06	66.34	61.15	57.55	57.03	58.51	60.29	61.64
8	63.65	62.50	60.74	58.43	56.04	55.02	56.10	58.22	60.20
16	58.37	57.87	57.14	56.11	54.77	53.67	53.76	55.61	58.06
32	55.78	55.54	55.20	54.66	53.91	53.06	52.50	53.18	55.35
64	54.51	54.41	54.21	53.88	53.42	52.84	52.23	51.94	52.87
128	53.92	53.85	53.71	53.53	53.28	52.85	52.29	51.80	51.65
256	53.66	53.61	53.50	53.37	53.20	52.96	52.54	52.04	51.60
512	53.53	53.49	53.41	53.33	53.21	53.02	52.78	52.38	51.90
ANT+SIN	1	2	4	8	16	32	64	128	256
1	108.24	84.75	67.42	59.91	58.15	58.49	59.24	59.85	60.23
2	87.60	78.40	68.32	60.63	57.54	57.47	58.33	59.23	59.87
4	71.12	68.32	64.31	59.78	56.63	56.06	57.01	58.24	59.23
8	62.23	61.38	59.98	57.93	55.69	54.59	55.30	56.79	58.21
16	57.83	57.51	57.00	56.17	54.96	53.76	53.61	54.92	56.68
32	55.62	55.51	55.33	54.96	54.38	53.55	52.80	53.13	54.73
64	54.57	54.49	54.40	54.25	53.98	53.51	52.84	52.36	52.89
128	54.02	53.98	53.95	53.85	53.72	53.49	53.07	52.53	52.12
256	53.76	53.74	53.71	53.67	53.59	53.47	53.23	52.85	52.33
512	53.64	53.63	53.60	53.57	53.51	53.45	53.35	53.12	52.75

Table A.9: Test mCE for various batch sizes (rows) vs. pseudo batch sizes (columns), part 3/3. DAug = DeepAugment, AM=AugMix.

SIN	1	2	4	8	16	32	64	128	256
1	119.11	94.43	74.93	67.03	65.43	66.08	67.16	68.04	68.62
2	98.85	88.62	76.99	67.88	64.23	64.42	65.72	67.02	67.99
4	81.35	78.10	73.38	67.84	63.49	62.47	63.76	65.48	66.94
8	70.92	69.94	68.38	66.02	63.14	61.09	61.45	63.35	65.35
16	65.29	64.97	64.48	63.68	62.39	60.78	59.90	60.92	63.16
32	62.34	62.25	62.08	61.80	61.36	60.55	59.55	59.26	60.65
64	60.84	60.80	60.74	60.61	60.47	60.15	59.67	58.96	58.93
128	60.07	60.04	60.02	59.96	59.87	59.77	59.57	59.18	58.64
256	59.68	59.66	59.64	59.62	59.59	59.53	59.43	59.27	58.97
512	59.48	59.47	59.46	59.44	59.42	59.40	59.33	59.26	59.11

DAug	1	2	4	8	16	32	64	128	256
8	65.37	63.87	61.37	58.11	54.48	52.17	52.33	54.18	56.36

DAug+AM	1	2	4	8	16	32	64	128	256
8	52.59	51.98	51.05	49.83	48.5	47.81	48.36	49.72	51.12

ResNext+DAug+AM	1	2	4	8	16	32	64	128	256
8	42.09	41.74	41.29	40.67	39.96	39.69	40.35	41.55	42.69

Analytical error model

We first consider a univariate model and later discuss a simple extension to the multivariate diagonal case. As highlighted in the main text, the model qualitatively explains the overall characteristics of our experimental data. Note that we assume a linear relationship between the Wasserstein distance and the error under domain shift, as suggested by our empirical findings.

Univariate model. We denote the source statistics as μ_s, σ_s^2, the true target statistics as μ_t, σ_t^2 and the estimated target statistics as $\hat{\mu}_t, \hat{\sigma}_t^2$. For normalization, we take a convex combination of the source statistics and estimated target statistics:

$$\bar{\mu} = \frac{N}{N+n}\mu_s + \frac{n}{N+n}\hat{\mu}_t,\ \bar{\sigma}^2 = \frac{N}{N+n}\sigma_s^2 + \frac{n}{N+n}\hat{\sigma}_t^2. \tag{A.8}$$

We now analyze the trade-off between using an estimate closer to the source or closer to the estimated target statistics. In the former case, the model will suffer under the covariate shift present between target and source distribution. In the latter case, small batch sizes n will yield unreliable estimates for the true target statistics, which might hurt the performance even more than the source-target mismatch. Hence, we aim to gain understanding in the trade-off between both options, and potential optimal choices of N for a given sample size n.

As a metric of domain shift with good properties for our following derivation, we leverage the Wasserstein distance. In the result section of the main paper and Table A.6, we already established an empirical link between domain shift measured in terms of the top-1 performance vs. the Wasserstein distance between model statistics and observed a linear relationship for case of common corruptions.

Proposition 1 (Bounds on the expected value of the Wasserstein distance between target and combined estimated target and source statistics)**.** *We denote the source statistics as μ_s, σ_s^2, the true target statistics as μ_t, σ_t^2 and the biased estimates of the target statistics as $\hat{\mu}_t, \hat{\sigma}_t^2$. For normalization, we take a convex combination of the source statistics and estimated target statistics as discussed in Eq. A.8. At a confidence level $1-\alpha$, the expectation value of the squared Wasserstein distance $W_2^2(\bar{\mu}, \bar{\sigma}, \mu_t, \sigma_t)$ between ideal and estimated target statistics w.r.t. to the distribution of sample mean $\hat{\mu}_t$ and sample variance $\hat{\sigma}_t^2$ is bounded from above and below with $L \leq \mathbb{E}[W_2^2] \leq U$, where*

$$\begin{aligned} L &= \left(\sigma_t - \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2}\right)^2 + \frac{N^2}{(N+n)^2}\left(\mu_t - \mu_s\right)^2 + \frac{n}{(N+n)^2}\sigma_t^2 \\ U &= L + \sigma_t^5 \frac{(n-1)}{2(N+n)^2}\left(\frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{1-\alpha/2,n-1}\sigma_t^2\right)^{-3/2} \end{aligned} \tag{A.9}$$

The quantity $\chi^2_{1-\alpha/2,n-1}$ denotes the left tail value of a chi square distribution with $n-1$ degrees

of freedom, defined as $P\left(X \leq \chi^2_{1-\alpha/2,n-1}\right) = \alpha/2$ *for* $X \sim \chi^2_{n-1}$.

Proof sketch

We are interested in the expected value of the Wasserstein distance defined in (A.1) between the target statistics μ_t, σ_t^2 and the mixed statistics $\bar{\mu}, \bar{\sigma}^2$ introduced above in equation (A.8), taken with respect to the distribution of the sample moments $\hat{\mu}_t$, $\hat{\sigma}_t^2$. The expectation value itself cannot be evaluated in closed form because the Wasserstein distance contains a term proportional to $\bar{\sigma}$ being the square root of the convex combination of target and source variance.

In Lemma 3, the square root term is bounded from above and below using Jensen's inequality and Hölder's defect formula which is reviewed in Lemma 2. After having bounded the problematic square root term, the proof of Proposition 1 reduces to inserting the expectation values of sample mean and sample variance reviewed in Lemma 1.

Prerequisites

Lemma 1 (Mean and variance of sample moments, following (Weisstein, 2020))**.** *The sample moments* $\hat{\mu}_t, \hat{\sigma}_t^2$ *are random variables depending on the sample size n.*

$$\hat{\mu}_t = \frac{1}{n}\sum_{j=1}^{n} x_j, \quad \hat{\sigma}_t^2 = \frac{1}{n}\sum_{j=1}^{n}(x_j - \hat{\mu}_t)^2 \;\; \textit{with } x_j \sim \mathcal{N}\left(\mu_t, \sigma_t^2\right). \tag{A.10}$$

For brevity, we use the shorthand $\mathbb{E}[\cdot]$ *for all expectation values with respect to the distribution of* $p(\hat{\mu}_t, \hat{\sigma}_t^2|n)$*. In particular, our computation uses mean and variance of* $\hat{\mu}_t$ *and* $\hat{\sigma}_t^2$ *which are well known for a normal target distribution:*

$$\hat{\mu}_t \sim \mathcal{N}\left(\mu_t, \frac{1}{n}\sigma_t^2\right), \; \mathbb{E}[\hat{\mu}_t] = \mu_t, \; \mathbb{V}[\hat{\mu}_t] = \frac{1}{n}\sigma_t^2 \tag{A.11}$$

$$\frac{\hat{\sigma}_t^2}{\sigma_t^2/n} \sim \chi^2_{n-1}, \; \mathbb{E}[\hat{\sigma}_t^2] = \frac{n-1}{n}\sigma_t^2, \; \mathbb{V}[\hat{\sigma}_t^2] = \frac{\sigma_t^4}{n^2}\mathbb{V}\left[\frac{\hat{\sigma}_t^2}{\sigma_t^2/n}\right] = \frac{\sigma_t^4}{n^2}\,2(n-1). \tag{A.12}$$

The derivation of the variance $\mathbb{V}[\hat{\sigma}_t^2]$ *in the last line uses the fact that the variance of a chi square distributed variable with* $(n-1)$ *degrees of freedom is equal to* $2(n-1)$.

Lemma 2 (Hölder's defect formula for concave functions in probabilistic notation, following Becker (2012))**.** *If the concave function* $f : [a, b] \to \mathbb{R}$ *is twice continuously differentiable and there are finite bounds m and M such that*

$$-M \leq f''(x) \leq -m \leq 0 \; \forall x \in [a, b], \tag{A.13}$$

then the defect between Jensen's inequality estimate $f\left(\mathbb{E}[X]\right)$ *for a random variable X taking values* $x \in [a, b]$ *and the true expectation value* $\mathbb{E}[f(X)]$ *is bounded from above by a term*

proportional to the variance of X:

$$f\left(\mathbb{E}[X]\right) - \mathbb{E}[f(X)] \leq \frac{1}{2} M \mathbb{V}[X]. \tag{A.14}$$

Lemma 3 (Upper and lower bounds on the expectation value of $\bar{\sigma}$)**.** *The expectation value of the square root of the random variable $\bar{\sigma}^2$ defined as*

$$\bar{\sigma}^2 = \frac{N}{N+n}\sigma_s^2 + \frac{n}{N+n}\hat{\sigma}_t^2, \tag{A.15}$$

is bounded from above and below at a confidence level $1-\alpha$ by

$$\sqrt{\mathbb{E}\left[\bar{\sigma}^2\right]} - \frac{1}{2} M \mathbb{V}[\bar{\sigma}^2] \leq \mathbb{E}\left[\sqrt{\bar{\sigma}^2}\right] \leq \sqrt{\mathbb{E}\left[\bar{\sigma}^2\right]} \tag{A.16}$$

$$\sqrt{\mathbb{E}\left[\bar{\sigma}^2\right]} = \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2}, \tag{A.17}$$

$$\frac{1}{2} M \mathbb{V}[\bar{\sigma}^2] = \frac{(n-1)}{4(N+n)^2}\sigma_t^4 \left(\frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{1-\alpha/2,n-1}\sigma_t^2\right). \tag{A.18}$$

The quantity $\chi^2_{1-\alpha/2,n-1}$ denotes the left tail value of a chi square distribution with $n-1$ degrees of freedom, defined as $P\left(X \leq \chi^2_{1-\alpha/2,n-1}\right) = \alpha/2$ for $X \sim \chi^2_{n-1}$.

Proof. The square root function is concave, therefore Jensen's inequality implies the upper bound

$$\mathbb{E}\left[\sqrt{\bar{\sigma}^2}\right] \leq \sqrt{\mathbb{E}[\bar{\sigma}^2]}. \tag{A.19}$$

The square root of the expectation value of $\bar{\sigma}^2$ is computed using the expectation value of the sample variance as given in Lemma 1.

$$\sqrt{\mathbb{E}[\bar{\sigma}^2]} = \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n}{N+n}\frac{n-1}{n}\sigma_t^2} = \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2}. \tag{A.20}$$

To state a lower bound, we use Hölder's defect formula in probabilistic notation stated in Lemma 2. Hölder's formula for concave functions requires that the random variable $\bar{\sigma}^2$ can take values in the compact interval $[a,b]$ and that the second derivative of the square root function $f(\bar{\sigma}^2) = \sqrt{\bar{\sigma}^2}$, exists and is strictly smaller than zero in $[a,b]$. Regarding the interval of $\bar{\sigma}^2$, we provide probabilistic upper and lower bounds. The ratio of sample variance and true variance divided by n follows a chi square distribution with $n-1$ degrees of freedom. At confidence level $1-\alpha$, this ratio lies

between $\chi^2_{1-\alpha/2,n-1}$ and $\chi^2_{\alpha/2,n-1}$ which are defined as follows:

$$\chi^2_{1-\alpha/2,n-1} \leq \frac{\hat{\sigma}_t^2}{\sigma_t^2/n} \leq \chi^2_{\alpha/2,n-1}, \tag{A.21}$$

$$Pr(X \leq \chi^2_{1-\alpha/2,n-1}) = \frac{\alpha}{2},\ Pr(X \geq \chi^2_{\alpha/2,n-1}) = \frac{\alpha}{2}. \tag{A.22}$$

Then at the same confidence level, the sample variance itself lies between the two quantiles multiplied by σ_t^2/n,

$$\chi^2_{1-\alpha/2,n-1}\frac{\sigma_t^2}{n} \leq \hat{\sigma}_t^2 \leq \chi^2_{\alpha/2,n-1}\frac{\sigma_t^2}{n}, \tag{A.23}$$

and the random variable $\bar{\sigma}^2$ lies in the interval

$$\bar{\sigma}^2 \in [a,b] \text{ with } a = \frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{1-\alpha/2,n-1}\sigma_t^2, \tag{A.24}$$

$$\text{and } b = \frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{\alpha/2,n-1}\sigma_t^2. \tag{A.25}$$

The variances and chi square values are all positive and therefore both a and b are positive as well, implying that the second derivative of the square root is strictly negative in the interval $[a,b]$.

$$f(\bar{\sigma}^2) = \sqrt{\bar{\sigma}^2},\ f'(\bar{\sigma}^2) = \frac{1}{2}(\bar{\sigma}^2)^{-1/2},\ f''(\bar{\sigma}^2) = -\frac{1}{4}(\bar{\sigma}^2)^{-3/2} < 0 \in [a,b]. \tag{A.26}$$

Consequently the second derivative is in the interval $[M,m]$ at the given confidence level:

$$-M \leq f''(\bar{\sigma}^2) \leq -m \leq 0 \text{ for } \bar{\sigma}^2 \in [a,b] \text{ with } M = \frac{1}{4}a^{-3/2},\ m = \frac{1}{4}b^{-3/2}. \tag{A.27}$$

The defect formula (Lemma 2) states that the defect is bounded by

$$\sqrt{\mathbb{E}[\bar{\sigma}^2]} - \mathbb{E}[\sqrt{\bar{\sigma}^2}] \leq \frac{1}{2}M\mathbb{V}[\bar{\sigma}^2]. \tag{A.28}$$

The constant M was computed above in (A.27), and the variance of $\bar{\sigma}^2$ is calculated in the next lines, using the first and second moment of the sample variance as stated in 1.

$$\begin{aligned}\mathbb{V}[\bar{\sigma}^2] &= \mathbb{E}[(\bar{\sigma}^2 - \mathbb{E}[\bar{\sigma}^2])^2] = \mathbb{E}\left[\left(\frac{n}{N+n}\hat{\sigma}_t^2 - \frac{n}{N+n}\frac{n-1}{n}\sigma_t^2\right)^2\right] \\ &= \frac{n^2}{(N+n)^2}\mathbb{E}\left[\left(\hat{\sigma}_t^2 - \mathbb{E}[\hat{\sigma}_t^2\right)^2\right] = \frac{n^2}{(N+n)^2}\mathbb{V}\left[\hat{\sigma}_t^2\right] \\ &= \frac{n^2}{(N+n)^2}\frac{2(n-1)}{n^2}\sigma_t^4 = \frac{2(n-1)}{(N+n)^2}\sigma_t^4.\end{aligned} \tag{A.29}$$

Inserting $\mathbb{V}[\bar{\sigma}^2]$ computed in (A.29) and M defined in (A.27) with a as defined in (A.24) into the defect formula (A.28) yields the lower bound:

$$\begin{aligned}
&\sqrt{\mathbb{E}[\bar{\sigma}^2]} - \frac{1}{2}M\mathbb{V}[\bar{\sigma}^2] \leq \mathbb{E}[\sqrt{\bar{\sigma}^2}] \\
&\sqrt{\mathbb{E}[\bar{\sigma}^2]} - \frac{1}{2}M\mathbb{V}[\bar{\sigma}^2] \\
&= \sqrt{\mathbb{E}[\bar{\sigma}^2]} - \frac{1}{2}\cdot\frac{1}{4}a^{-3/2}\frac{2(n-1)}{(N+n)^2}\sigma_t^4 \\
&= \sqrt{\mathbb{E}[\bar{\sigma}^2]} - \frac{(n-1)}{4(N+n)^2}\sigma_t^4\left(\frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{1-\alpha/2,n-1}\sigma_t^2\right)^{-3/2}.
\end{aligned} \tag{A.30}$$

Assuming that source and target variance are of the same order of magnitude σ, the defect will be of order of magnitude σ: The factor $\mathbb{V}[X]$ scales with σ^4 and M with σ^{-3}. □

Proof of Proposition 1

Proof. For two univariate normal distributions with moments μ_t, σ_t^2 and $\bar{\mu}, \bar{\sigma}^2$, the Wasserstein distance as defined in (A.1) reduces to

$$W_2^2 = \sigma_t^2 + \bar{\sigma}^2 - 2\bar{\sigma}\sigma_t + (\bar{\mu} - \mu)^2. \tag{A.31}$$

The expected value of the Wasserstein distance across many batches is given as

$$\begin{aligned}
\mathbb{E}[W_2^2] &= \sigma_t^2 + \mathbb{E}[\bar{\sigma}^2] - 2\mathbb{E}[\bar{\sigma}]\sigma_t + \mathbb{E}[(\mu_t - \bar{\mu})^2] \\
&= \sigma_t^2 + \frac{N}{N+n}\sigma_s^2 + \frac{n}{N+n}\frac{n-1}{n}\sigma_t^2 - 2\sigma_t\mathbb{E}\left[\sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n}{N+n}\hat{\sigma}_t^2}\right] \\
&\quad + \mathbb{E}\left[\left(\mu_t - \frac{N}{N+n}\mu_s - \frac{n}{N+n}\hat{\mu}_t\right)^2\right]
\end{aligned} \tag{A.32}$$

which can already serve as the basis for our numerical simulations. To arrive at a closed form analytical solution, we invoke Lemma 3 to bound the expectation value $\mathbb{E}[\bar{\sigma}]$ in equation (A.32).

$$-2\sigma_t\sqrt{\mathbb{E}[\bar{\sigma}^2]} \leq -2\sigma_t\mathbb{E}\left[\sqrt{\bar{\sigma}^2}\right] \leq -2\sigma_t\sqrt{\mathbb{E}[\bar{\sigma}^2]} - 2\sigma_t\left(-\frac{1}{2}M\mathbb{V}[\bar{\sigma}^2]\right) \tag{A.33}$$

Apart from the square root term bounded in equation (A.33) above, the expectation value of the Wasserstein distance can be computed exactly. Hence the bounds on $\mathbb{E}[\bar{\sigma}]$ multiplied by a factor of $(-2\sigma_t^2)$ coming from equation (A.32) determine lower and upper bounds L and U on the expected value of W_2^2:

$$L \leq \mathbb{E}\left[W_2^2\right] \leq U = L + \sigma_t M\mathbb{V}[\bar{\sigma}^2] \tag{A.34}$$

In the next lines, the lower bound is calculated:

$$
\begin{aligned}
L = \; & \sigma_t^2 + \frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2 - 2\sigma_t\sqrt{\mathbb{E}\left[\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2\right]} \\
& + \left(\mu_t - \frac{N}{N+n}\mu_s\right)^2 - 2\left(\mu_t - \frac{N}{N+n}\mu_s\right)\frac{n}{N+n}\mathbb{E}[\hat{\mu}_t] + \frac{n^2}{(N+n)^2}\left(\mathbb{V}[\hat{\mu}_t] + (\mathbb{E}[\hat{\mu}_t])^2\right) \\
= \; & \sigma_t^2 + \frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2 - 2\sigma_t\sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2} \\
& + \left(\mu_t - \frac{N}{N+n}\mu_s\right)^2 - 2\left(\mu_t - \frac{N}{N+n}\mu_s\right)\frac{n}{N+n}\mu_t + \frac{n^2}{(N+n)^2}\left(\frac{1}{n}\sigma_t^2 + \mu_t^2\right) \\
= \; & \left(\sigma_t - \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2}\right)^2 + \left(\mu_t - \frac{N}{N+n}\mu_s - \frac{n}{N+n}\mu_t\right)^2 + \frac{n}{(N+n)^2}\sigma_t^2 \\
= \; & \left(\sigma_t - \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2}\right)^2 + \frac{N^2}{(N+n)^2}\left(\mu_t - \mu_s\right)^2 + \frac{n}{(N+n)^2}\sigma_t^2
\end{aligned}
\tag{A.35}
$$

After having derived the lower bound, the upper bound is the sum of the lower bound and the defect term as computed in Lemma 3.

$$
\begin{aligned}
\mathbb{E}[W^2] \geq U &= L + \sigma_t M \mathbb{V}[\bar{\sigma}^2] \\
&= L + \sigma_t \frac{1}{4}\left(\frac{N}{N+n}\sigma_s^2 + \frac{n}{N+n}\chi^2_{1-\alpha/2,n-1}\frac{\sigma_t^2}{n}\right)^{-3/2}\frac{2(n-1)}{(N+n)^2}\sigma_t^4 \\
&= L + \left(\frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{1-\alpha/2,n-1}\sigma_t^2\right)^{-3/2}\frac{(n-1)}{2(N+n)^2}\sigma_t^5.
\end{aligned}
\tag{A.36}
$$

□

Based on choices of the model parameters, the model qualitatively matches our experimental results. We plot different choices in Fig. A.8.

Extension to multivariate distributions.

We now derive a multivariate variant that can be fit to data from a DNN. Due to the estimation of running statistics in the network, we have access to a diagonal approximation of the true covariance matrix.

We denote the diagonal covariance matrices with matrix elements σ_i^2 as

$$
(\Sigma_t)_{ii} = (\sigma_t^2)_i,\; (\hat{\Sigma}_t)_{ii} = (\hat{\sigma}_t^2)_i,\; (\Sigma_s)_{ii} = (\sigma_s^2)_i \tag{A.37}
$$

and extend our definition of the statistics used for normalization to $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\Sigma}}$:

$$
\bar{\boldsymbol{\mu}} = \frac{N}{N+n}\boldsymbol{\mu}_s + \frac{n}{N+n}\hat{\boldsymbol{\mu}}_t,\; \bar{\boldsymbol{\Sigma}} = \frac{N}{N+n}\boldsymbol{\Sigma}_s + \frac{n}{N+n}\hat{\boldsymbol{\Sigma}}_t. \tag{A.38}
$$

Figure A.8: Overview of different parametrizations of the model. We denote each plot with $(\mu_t - \mu_s, \sigma_t/\sigma_s)$ and report the lower bound $\sqrt{L}$ on the Wasserstein distance. Parametrizations in columns four to seven produce qualitatively similar results we observed in our experiments, assuming a linear relationship between the Wasserstein distance and the error rate.

The Wasserstein distance between $\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}$ and $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$ is then defined as

$$\begin{aligned} W_2^2 =& \operatorname{Tr} \boldsymbol{\Sigma}_t + \bar{\boldsymbol{\Sigma}} - 2\boldsymbol{\Sigma}_t^{1/2}\bar{\boldsymbol{\Sigma}}^{1/2} + (\boldsymbol{\mu}_t - \bar{\boldsymbol{\mu}})^T(\boldsymbol{\mu}_t - \bar{\boldsymbol{\mu}}) \\ =& \sum_{i=1}^{D} (\sigma_t^2)_i + (\bar{\sigma}^2)_i - 2(\bar{\sigma})_i(\sigma_t)_i + \left((\mu_t)_i - (\bar{\mu}_t)_i\right)^2 = \sum_{i=1}^{D} (W_2^2)_i \end{aligned} \tag{A.39}$$

Every component $(W_2^2)_i$ in the sum above is bounded by the univariate bound discussed above. The multivariate Wasserstein distance which sums over the diagonal covariance matrix entries is then bounded by the sums over the individual bounds L_i and U_i given in (A.9).

$$L_i \leq (W_2^2)_i \leq U_i \Rightarrow \sum_{i=1}^{D} L_i \leq W_2^2 \leq \sum_{i=1}^{D} U_i. \tag{A.40}$$

Limits of Proposition 1

Limit $n \to \infty$ In the limit of infinite batch size $n \to \infty$, upper and lower bounds on the expected Wasserstein distance between $\bar{\mu}, \bar{\sigma}^2$ and μ_t, σ_t^2 both go to zero.

$$\begin{aligned} \lim_{n\to\infty} L =& \lim_{n\to\infty} \left(\sigma_t - \sqrt{\frac{N}{N+n}\sigma_s^2 + \frac{n-1}{N+n}\sigma_t^2} \right)^2 + \frac{N^2}{(N+n)^2} (\mu_t - \mu_s)^2 + \frac{n}{(N+n)^2}\sigma_t^2 \\ =& (\sigma_t - \sigma_t)^2 = 0 \\ \lim_{n\to\infty} U =& \lim_{n\to\infty} L + \lim_{n\to\infty} \sigma_t^5 \frac{(n-1)}{2(N+n)^2} \left(\frac{N}{N+n}\sigma_s^2 + \frac{1}{N+n}\chi^2_{1-\alpha/2,n-1}\sigma_t^2 \right)^{-3/2} = 0. \end{aligned} \tag{A.41}$$

The intuition behind this limit is that if a large number of samples from the target domain is given, $\hat{\mu}$ and $\hat{\sigma}^2$ approximate the true target statistics very well. As $\hat{\mu}$ and $\hat{\sigma}^2$ dominate $\bar{\mu}$ and $\bar{\sigma}^2$ for large n, the expected Wasserstein distance has to vanish.

Limit $N \to \infty$ In the opposite limit $N \to \infty$, the expected value of the Wasserstein distance reduces to the Wasserstein distance between source and target statistics.

$$\lim_{N\to\infty} \bar{\mu} = \mu_s, \ \lim_{N\to\infty} \bar{\sigma}^2 = \sigma_s^2, \tag{A.42}$$

$$\Rightarrow \lim_{N\to\infty} \mathbb{E}[W_2^2] = \sigma_t^2 + \sigma_s^2 - 2\sigma_t\sigma_s + (\mu_t - \mu_s)^2 = W_2^2\left(\mu_s, \sigma_s^2, \mu_t, \sigma_t^2\right). \tag{A.43}$$

Limiting case $\mu_t = \mu_s$ and $\sigma_t^2 = \sigma_s^2$ When source and target domain coincide, and the statistics $\sigma_s^2 = \sigma_t^2$ and $\mu_s = \mu_t$ are known, then the source target mismatch is not an error source.

However, one might assume that source and target domain are different even though they actually coincide. In this case, proceeding with our proposed strategy and using the statistics $\bar{\mu}$ and $\bar{\sigma}^2$, the bounds on the expected Wasserstein distance follow

from setting σ_t^2 to σ_s^2 and μ_t to μ_s in Proposition 1.

$$\begin{aligned}
&\bar{\mu} = \frac{N}{N+n}\mu_t + \frac{n}{N+n}\hat{\mu}_t,\ \bar{\sigma}^2 = \frac{N}{N+n}\sigma_t^2 + \frac{n}{N+n}\hat{\sigma}_t^2,\ L \leq \mathbb{E}[W_2^2] \leq U \\
&L = \sigma_t^2 \left(\frac{2N^2 + 4Nn - N + 2n^2}{(N+n)^2} - 2\sqrt{1 - \frac{1}{N+n}} \right), \\
&U = L + \sigma_t^2 \frac{n-1}{2(N+n)^2} \left(\frac{N + \chi^2_{1-\alpha/2,n-1}}{N+n} \right)^{-3/2}.
\end{aligned} \tag{A.44}$$

It could also be the case that the equality of source and target statistics is known but the concrete values of the statistics are unknown. In our model, this amounts to setting the number of pseudo samples N to zero and assuming that source and target statistics are equal. Setting $N = 0$ in equation (A.44) and keeping n finite yields

$$L = 2\sigma_t^2 \left(1 - \sqrt{1 - \frac{1}{n}} \right),\ U = L + \sigma_t^2 \frac{n-1}{2n^2} \left(\frac{\chi^2_{1-\alpha/2,n-1}}{n} \right)^{-3/2}. \tag{A.45}$$

Bounds on the normalized Wasserstein distance

The Wasserstein distance (A.1) between the interpolating statistics $\bar{\mu}$, $\bar{\sigma}^2$ and the target statistics can also be normalized by a factor of σ_s^{-2}. Because σ_s^{-2} is constant, the bounds on the expectation value of the unnormalized Wasserstein distance discussed in the previous subsections just have to be multiplied by σ_s^{-2} to obtain bounds on the normalized Wasserstein distance (A.3):

$$\frac{L}{\sigma_s^2} \leq \widetilde{W}_2^2 = W_2^2 \left(\frac{\bar{\mu}}{\sigma_s}, , \frac{\bar{\sigma}^2}{\sigma_s^2}, \frac{\mu_t}{\sigma_s}, \frac{\sigma_t^2}{\sigma_s^2} \right) = \frac{1}{\sigma_s^2} W_2^2(\bar{\mu}, \bar{\sigma}^2, \mu_t, \sigma_t^2) \leq \frac{U}{\sigma_s^2}. \tag{A.46}$$

Full list of models evaluated on IN

The following lists contains all models we evaluated on various datasets with references and links to the corresponding source code.

Torchvision models trained on IN

Weights were taken from https://github.com/pytorch/vision/tree/master/torchvision/models

1. `alexnet` (Krizhevsky et al., 2012a)
2. `densenet121` (Huang et al., 2017)
3. `densenet161` (Huang et al., 2017)
4. `densenet169` (Huang et al., 2017)
5. `densenet201` (Huang et al., 2017)
6. `densenet201` (Huang et al., 2017)
7. `googlenet` (Szegedy et al., 2015)
8. `inception_v3` (Szegedy et al., 2016)
9. `mnasnet0_5` (Tan et al., 2019)
10. `mnasnet1_0` (Tan et al., 2019)
11. `mobilenet_v2` (Sandler et al., 2018a)
12. `resnet18` (He et al., 2016c)
13. `resnet34` (He et al., 2016c)
14. `resnet50` (He et al., 2016c)
15. `resnet101` (He et al., 2016c)
16. `resnet152` (He et al., 2016c)
17. `resnext50_32x4d` (Xie et al., 2017)
18. `resnext101_32x8d` (Xie et al., 2017)
19. `shufflenet_v2_x0_5` (Ma et al., 2018)
20. `shufflenet_v2_x1_0` (Ma et al., 2018)
21. `vgg11_bn` (Simonyan & Zisserman, 2015)
22. `vgg13_bn` (Simonyan & Zisserman, 2015)
23. `vgg16_bn` (Simonyan & Zisserman, 2015)
24. `vgg19_bn` (Simonyan & Zisserman, 2015)
25. `wide_resnet101_2` (Zagoruyko & Komodakis, 2016)
26. `wide_resnet50_2` (Zagoruyko & Komodakis, 2016)

Robust ResNet50 models

1. `resnet50 AugMix` (Hendrycks et al., 2020b) https://github.com/google-research/augmix
2. `resnet50 SIN+IN` (Geirhos et al., 2019) https://github.com/rgeirhos/texture-vs-shape
3. `resnet50 ANT` (Rusak et al., 2020) https://github.com/bethgelab/game-of-noise
4. `resnet50 ANT+SIN` (Rusak et al., 2020) https://github.com/bethgelab/game-of-noise
5. `resnet50 DeepAugment` (Hendrycks et al., 2020a) https://github.com/hendrycks/imagenet-r
6. `resnet50 DeepAugment+AugMix` (Hendrycks et al., 2020a) https://github.com/hendrycks/imagenet-r

SimCLRv2 models (Chen et al., 2020c)

We used the checkpoints from https://github.com/google-research/simclr and converted them from TensorFlow to PyTorch with https://github.com/tonylins/simclr-converter, commit ID: 139d3cb0bd0c64b5ad32aab810e0bd0a0ddd aae0.

1. `resnet50` FT100 SK=0 width=1
2. `resnet101` FT100 SK=0 width=1
3. `resnet152` FT100 SK=0 width=1

Robust ResNext models (Xie et al., 2017)

Note that the baseline `resnext50_32x4d` model trained on ImageNet is available as part of the `torchvision` library.

1. `resnext50_32x4d WSL` (Mahajan et al., 2018) https://github.com/facebookresearch/WSL-Images/blob/master/hubconf.py
2. `resnext101_32x4d WSL` (Mahajan et al., 2018) https://github.com/facebookresearch/WSL-Images/blob/master/hubconf.py
3. `resnext101_32x8d Deepaugment+AugMix` (Hendrycks et al., 2020a) https://github.com/hendrycks/imagenet-r

ResNet50 with Group Normalization (Wu & He, 2018)

Model weights and training code was taken from https://github.com/ppwwyyxx/GroupNorm-reproduce

1. `resnet50 GroupNorm`
2. `resnet101 GroupNorm`
3. `resnet152 GroupNorm`

ResNet50 with Fixup initialization (Zhang et al., 2019)

Model weights and training code was taken from https://github.com/hongyi-zhang/Fixup/tree/master/imagenet. For training, we keep all hyperparameters at their default values and note that in particular the batchsize of 256 is a sensitive parameter.

1. `resnet50 FixUp`
2. `resnet101 FixUp`
3. `resnet152 FixUp`

B

If your data distribution shifts, use self-learning

A two-point model of self-learning

Proof of Proposition 1

Learning dynamics with stop gradient. Computing the stop gradient evolution defined in (3.7) explicitly yields

$$\begin{aligned}\dot{\mathbf{w}}^s &= -\nabla_{\mathbf{w}^s}\mathcal{L} = \frac{1}{\tau_s}\sum_{i=1}^{N}\left(\sigma_t(\mathbf{x}_i^\top\mathbf{w}^t)\sigma_s(-\mathbf{x}_i^\top\mathbf{w}^s) - \sigma_t(-\mathbf{x}_i^\top\mathbf{w}^t)\sigma_s(\mathbf{x}_i^\top\mathbf{w}^s)\right)\mathbf{x}_i \\ \dot{\mathbf{w}}^t &= \alpha(\mathbf{w}^s - \mathbf{w}^t)\end{aligned} \tag{B.1}$$

The second equality uses the well-known derivative of the sigmoid function, $\partial_z\sigma(z) = \sigma(z)\sigma(-z)$.

The equation system of $2d$ nonlinear, coupled ODEs for $\mathbf{w}^s \in \mathbb{R}^d$ and $\mathbf{w}^t \in \mathbb{R}^d$ in (B.1) is analytically difficult to analyze. Instead of studying the ODEs directly, we act on them with the data points $\mathbf{x}_k^\top$, $k = 1, \ldots, N$, and investigate the dynamics of the components $\mathbf{x}_k^\top\mathbf{w}^{s,t} \equiv y_k^{s,t}$:

$$\begin{aligned}\dot{y}_k^s &= \frac{1}{\tau_s}\sum_{i=1}^{N}\left(\mathbf{x}_i^\top\mathbf{x}_k\right)\left(\sigma_t(y_i^t)\sigma_s(-y_i^s) - \sigma_t(-y_i^t)\sigma_s(y_i^s)\right) \\ \dot{y}_k^t &= \alpha(y_k^s - y_k^t).\end{aligned} \tag{B.2}$$

The learning rate of each mode y_k^s is scaled by $(\mathbf{x}_k^\top\mathbf{x}_i)$ which is much larger for $i = k$

than for $i \neq k$ in high-dimensional spaces. In the two-point approximation, we consider only the two (in absolute value) largest terms $i = k, l$ for a given k in the sum in (B.2). Any changes that $y_k^{s,t}(t)$ and $y_l^{s,t}(t)$ might induce in other modes $y_i^{s,t}(t)$ are neglected, and so we are left with only four ODEs:

$$\begin{aligned}
\dot{y}_k^s =& \frac{1}{\tau_s}\|\mathbf{x}_k\|^2 \left(\sigma_t(y_k^t)\sigma_s(-y_k^s) - \sigma_t(-y_k^t)\sigma_s(y_k^s)\right) \\
&+ \frac{1}{\tau_s}(\mathbf{x}_k^\top \mathbf{x}_l) \left(\sigma_t(y_l^t)\sigma_s(-y_l^s) - \sigma_t(-y_l^t)\sigma_s(y_l^s)\right), \\
\dot{y}_l^s =& \frac{1}{\tau_s}\|\mathbf{x}_l\|^2 \left(\sigma_t(y_l^t)\sigma_s(-y_l^s) - \sigma_t(-y_l^t)\sigma_s(y_l^s)\right) \\
&+ \frac{1}{\tau_s}(\mathbf{x}_k^\top \mathbf{x}_l) \left(\sigma_t(y_k^t)\sigma_s(-y_k^s) - \sigma_t(-y_k^t)\sigma_s(y_k^s)\right) \\
\dot{y}_k^t =& \alpha(y_k^s - y_k^t),\ \dot{y}_l^t = \alpha(y_l^s - y_l^t).
\end{aligned} \tag{B.3}$$

The fixed points of (B.3) satisfy

$$\dot{y}_k^s = \dot{y}_l^s = \dot{y}_k^t = \dot{y}_l^t = 0. \tag{B.4}$$

For $\alpha > 0$, requiring $\dot{y}_k^t = \dot{y}_l^t = 0$ implies that $y_k^s = y_k^t$ and $y_l^s = y_l^t$. For $\tau_s = \tau_t$, the two remaining equations $\dot{y}_k^s = \dot{y}_l^s = 0$ vanish automatically so that there are no non-trivial two-point learning dynamics. For $\tau_s \neq \tau_t$, there is a fixed point at $y_k^{s,t} = y_l^{s,t} = 0$ since at this point, each bracket in (B.3) vanishes individually:

$$\sigma_t(y_{k,l})\sigma_s(-y_{k,l}) - \sigma_s(-y_{k,l})\sigma_t(y_{k,l})\Big|_{y_{k,l}=0} = \frac{1}{4} - \frac{1}{4} = 0. \tag{B.5}$$

At the fixed point $y_k^{s,t} = y_l^{s,t} = 0$, $\mathbf{w}^s$ and $\mathbf{w}^t$ are orthogonal to both $\mathbf{x}_k$ and $\mathbf{x}_l$ and hence classification fails. If this fixed point is stable, $\mathbf{w}^s$ and $\mathbf{w}^t$ will stay at the fixed point once they have reached it, i.e. the model collapses. The fixed point is stable when all eigenvalues of the Jacobian J of the ODE system (B.3) evaluated at $y_k^{s,t} = y_l^{s,t} = 0$ are negative. Two eigenvalues $\lambda_\pm$ are always negative, whereas the two other eigenvalues $\tilde{\lambda}_\pm$ are positive if $\tau_s > \tau_t$:

$$\begin{aligned}
J\Big|_{y_k^{s,t}=y_l^{s,t}=0} &= \begin{pmatrix}
\frac{-\|\mathbf{x}_k\|^2}{4\tau_s^2} & \frac{-(\mathbf{x}_k^\top \mathbf{x}_l)}{4\tau_s^2} & \frac{\|\mathbf{x}_k\|^2}{4\tau_s\tau_t} & \frac{(\mathbf{x}_k^\top \mathbf{x}_l)}{4\tau_s\tau_t} \\
\frac{-(\mathbf{x}_k^\top \mathbf{x}_l)}{4\tau_s^2} & \frac{-\|\mathbf{x}_l\|^2}{4\tau_s^2} & \frac{(\mathbf{x}_k^\top \mathbf{x}_l)}{4\tau_s\tau_t} & \frac{\|\mathbf{x}_l\|^2}{4\tau_s\tau_t} \\
\alpha & 0 & -\alpha & 0 \\
0 & \alpha & 0 & -\alpha
\end{pmatrix}, \\
\lambda_\pm &= -\frac{1}{16\tau_s^2}\left(A_\pm + \sqrt{A_\pm^2 + 32\alpha\tau_s^2 B(\tau_s/\tau_t - 1)}\right) < 0, \\
\tilde{\lambda}_\pm &= \frac{1}{16\tau_s^2}\left(-A_\pm + \sqrt{A_\pm^2 + 32\alpha\tau_s^2 B(\tau_s/\tau_t - 1)}\right) > 0 \text{ if } \tau_s > \tau_t, \text{ where} \\
A_\pm &= 8\alpha\tau_s^2 \pm B, \quad B = \|\mathbf{x}_k\| + \|\mathbf{x}_l\| + \sqrt{(\|\mathbf{x}_k\| - \|\mathbf{x}_l\|)^2 + 4(\mathbf{x}_k^\top \mathbf{x}_l)^2}
\end{aligned} \tag{B.6}$$

To sum up, training with stop gradient and $\tau_s > \tau_t$ avoids a collapse of the two-point model to the trivial representation $y_k^{s,t} = y_l^{s,t} = 0$ since the fixed point is not stable in this parameter regime.

Learning dynamics without stop gradient Without stop gradient, we set $\mathbf{w}^t = \mathbf{w}^s \equiv \mathbf{w}$ which leads to an additional term in the gradient:

$$\begin{aligned}\dot{\mathbf{w}} = -\nabla_{\mathbf{w}}\mathcal{L} =& \frac{1}{\tau_s}\sum_{i=1}^{N}\left(\sigma_t(\mathbf{x}_i^\top\mathbf{w})\sigma_s(-\mathbf{x}_i^\top\mathbf{w}) - \sigma_t(-\mathbf{x}_i^\top\mathbf{w})\sigma_s(\mathbf{x}_i^\top\mathbf{w})\right)\mathbf{x}_i \\ &+ \frac{1}{\tau_t}\sum_{i=1}^{N}\sigma_t(\mathbf{x}_i^\top\mathbf{w})\sigma_t(-\mathbf{x}_i^\top\mathbf{w})\underbrace{\left(\log\sigma_s(\mathbf{x}_i^\top\mathbf{w}) - \log\sigma_s(-\mathbf{x}_i^\top\mathbf{w})\right)}_{=\log\left((1+e^{y_i/\tau_s})/(1+e^{-y_i/\tau_s})\right)=y_i/\tau_s}\mathbf{x}_i.\end{aligned} \tag{B.7}$$

As before, we focus on the evolution of the two components $y_k = \mathbf{w}^\top\mathbf{x}_k$ and $y_l = \mathbf{w}^\top\mathbf{x}_l$.

$$\begin{aligned}\dot{y}_k =& \|\mathbf{x}_k\|^2\left(\frac{1}{\tau_s}\left(\sigma_t(y_k)\sigma_s(-y_k) - \sigma_t(-y_k)\sigma_s(y_k)\right) + \frac{1}{\tau_t}\sigma_t(y_k)\sigma_t(-y_k)y_k\right) \\ &+ (\mathbf{x}_k^\top\mathbf{x}_l)\left(\frac{1}{\tau_s}\left(\sigma_t(y_l)\sigma_s(-y_l) - \sigma_t(-y_l)\sigma_s(y_l)\right) + \frac{1}{\tau_s\tau_t}\sigma_t(y_l)\sigma_t(-y_l)y_l\right) \\ \dot{y}_l =& \|\mathbf{x}_l\|^2\left(\frac{1}{\tau_s}\left(\sigma_t(y_l)\sigma_s(-y_l) - \sigma_t(-y_l)\sigma_s(y_l)\right) + \frac{1}{\tau_t}\sigma_t(y_l)\sigma_t(-y_l)y_l\right) \\ &+ (\mathbf{x}_k^\top\mathbf{x}_l)\left(\frac{1}{\tau_s}\left(\sigma_t(y_k)\sigma_s(-y_k) - \sigma_t(-y_k)\sigma_s(y_k)\right) + \frac{1}{\tau_s\tau_t}\sigma_t(y_k)\sigma_t(-y_k)y_k\right)\end{aligned} \tag{B.8}$$

There is a fixed point at $y_k = y_l = 0$ where each bracket in (B.8) vanishes individually,

$$\frac{1}{\tau_s}\left(\sigma_t(y_{k,l})\sigma_s(-y_{k,l}) - \sigma_t(-y_{k,l})\sigma_s(y_{k,l})\right) + \frac{1}{\tau_s\tau_t}\sigma_t(y_{k,l})\sigma_t(-y_{k,l})y_{k,l}\Big|_{y_{k,l}} = 0. \tag{B.9}$$

The Jacobian of the ODE system in (B.8) and its eigenvalues evaluated at the fixed point are given by

$$\begin{aligned}J\Big|_{y_k=y_l=0} &= \begin{pmatrix}\frac{\|\mathbf{x}_k\|^2}{4\tau_s}\left(\frac{2}{\tau_t}-\frac{1}{\tau_s}\right) & \frac{(\mathbf{x}_k^\top\mathbf{x}_l)}{4\tau_s}\left(\frac{2}{\tau_t}-\frac{1}{\tau_s}\right) \\ \frac{(\mathbf{x}_k^\top\mathbf{x}_l)}{4\tau_s}\left(\frac{2}{\tau_t}-\frac{1}{\tau_s}\right) & \frac{\|\mathbf{x}_l\|^2}{4\tau_s}\left(\frac{2}{\tau_t}-\frac{1}{\tau_s}\right)\end{pmatrix} \\ \lambda_{1,2} &= \frac{1}{8\tau_s}\left(\frac{2}{\tau_t}-\frac{1}{\tau_s}\right)\underbrace{\left(\pm\underbrace{\sqrt{\|\mathbf{x}_k\|^4 + \|\mathbf{x}_l\|^4 - 2\|\mathbf{x}_k\|^2\|\mathbf{x}_l\|^2 + 4(\mathbf{x}_k^\top\mathbf{x}_l)^2}}_{\leq\|\mathbf{x}_k\|^2+\|\mathbf{x}_l\|^2} + \|\mathbf{x}_k\|^2 + \|\mathbf{x}_l\|^2\right)}_{\geq 0 \text{ with equality if } \mathbf{x}_k=\pm\mathbf{x}_l}.\end{aligned} \tag{B.10}$$

Hence the fixed point is unstable when $\tau_s > \tau_t/2$ and thus the model without stop gradient does not collapse onto $y_k = y_l = 0$ in this regime, concluding the proof.

Simulation of the two-point model

For visualization purposes in the main paper, we set $\mathbf{w}^s = \mathbf{w}^t = [0.5, 0.5]^\top$ and train the model using instant gradient updates on the dataset with points $\mathbf{x}_1 = [1, 0]$ and $\mathbf{x}_2 = [0, -1]$ using SGD with learning rate 0.1 and momentum 0.9. We varied student and teacher temperatures on a log-scale with 250 points from 10^{-3} to 10. Qualitatively similar results can be obtained without momentum training, at higher learning rates (most likely due to the implicit learning rate scaling introduced by the momentum term).

Note that the temperature scales for observing the collapse effect depend on the learning rate, and the exact training strategy—lower learning rates can empirically prevent the model from collapsing and shift the convergence region. The result in Figure 3.2 will hence depend on the exact choice of learning rate (which is currently not considered in our continuous time evolution theory), while the predicted region without collapse is robust to details of the optimization.

To visualize the impact of different hyperparameters, we show variants of the two point model with different learning rates using gradient descent with (Figure B.1) and without momentum (Figure B.2), and with different start conditions (Figure B.3), which all influence the regions where the model degrades, but not the stable regions predicted by our theory.

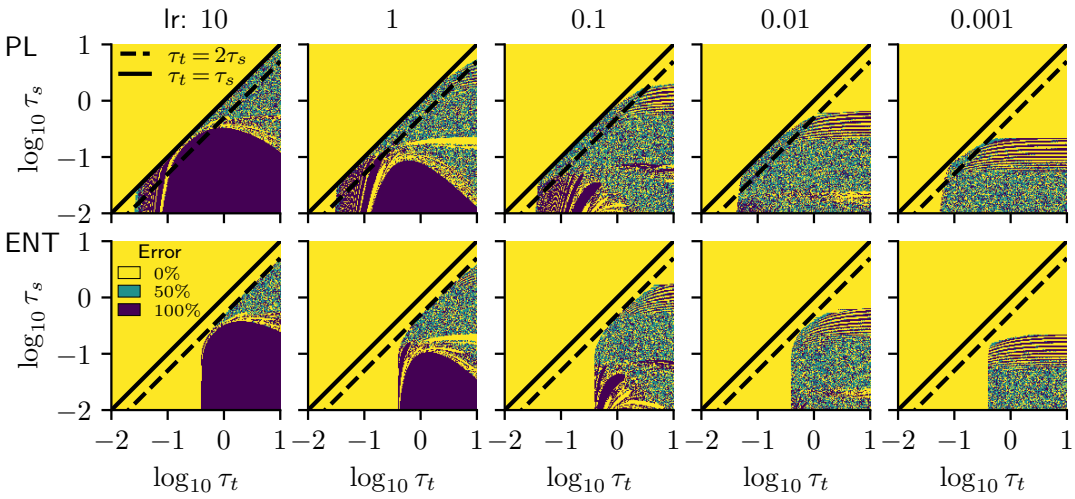


Figure B.1: Entropy minimization (top) Training two point model with momentum 0.9 and different learning rates with initialization $\mathbf{w}^s = \mathbf{w}^t = [0.5, 0.5]^\top$.

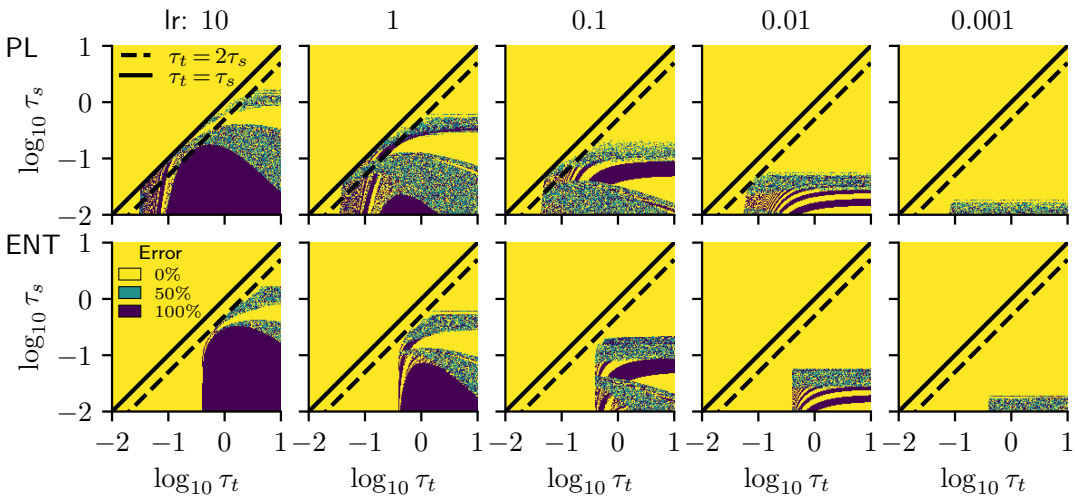


Figure B.2: Training a two point model without momentum and different learning rates with initialization $\mathbf{w}^s = \mathbf{w}^t = [0.5, 0.5]^\top$. Note that especially for lower learning rates, longer training would increase the size of the collapsed region.

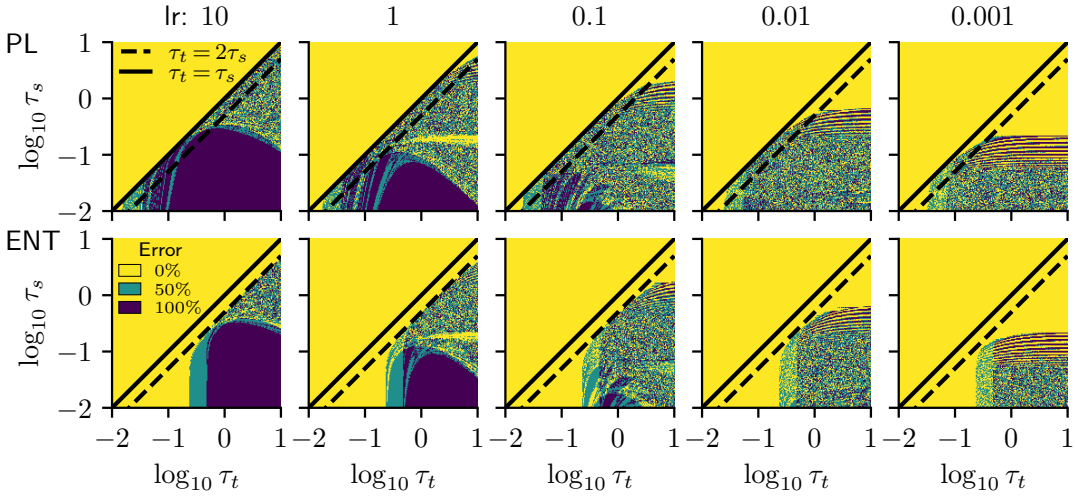


Figure B.3: Training a two point model with momentum 0.9 and different learning rates with initialization $\mathbf{w}^s = \mathbf{w}^t = [0.6, 0.3]^\top$.

Additional information on used models

Details on all hyperparameters we tested for different models

For all models except EfficientNet-L2, we adapt the batch norm statistics to the test domains following (Schneider et al., 2020a). We do not expect significant gains for combining EfficientNet-L2 with batch norm adaptation: as demonstrated in (Schneider et al., 2020a), models trained with large amounts of weakly labeled data do not seem to benefit from batch norm adaptation.

ResNet50 models (IN-C) We use a vanilla ResNet50 model and compare soft- and hard-labeling against entropy minimization and robust pseudo-labeling. To find optimal hyperparameters for all methods, we perform an extensive evaluation and test (i) three different adaptation mechanisms (ii) several learning rates 1.0×10^{-4}, 1.0×10^{-3}, 1.0×10^{-2} and 5.0×10^{-2}, (iii) the number of training epochs and (iv) updating the teacher after each epoch or each iteration. For all experiments, we use a batch size of 128. The hyperparameter search is performed on IN-C dev. We then use the optimal hyperparameters to evaluate the methods on the IN-C test set.

ResNeXt101 models The ResNeXt101 model is considerably larger than the ResNet50 model and we therefore limit the number of ablation studies we perform for this architecture. Besides a baseline, we include a state-of-the-art robust version trained with DeepAugment+Augmix (DAug+AM, Hendrycks et al., 2020a) and a version that was trained on 3.5 billion weakly labeled images (IG-3.5B, Mahajan et al., 2018). We only test the two leading methods on the ResNeXt101 models (ENT and RPL). We vary the learning rate in same interval as for the ResNet50 model but scale it down linearly to account for the smaller batch size of 32. We only train the affine batch normalization parameters because adapting only these parameters leads to the best results on ResNet50 and is much more resource efficient than adapting all model parameters. Again, the hyperparameter search is performed only on the development corruptions of IN-C. We then use the optimal hyperparameters to evaluate the methods on the IN-C test set.

EfficientNet-L2 models The current state of the art on IN, IN-C, IN-R and IN-A is an EfficientNet-L2 trained on 300 million images from JFT-300M (Chollet, 2017; Hinton et al., 2014) using a noisy student-teacher protocol (Xie et al., 2020a). We adapt this model for only one epoch due to resource constraints. During the hyperparameter search, we only evaluate three corruptions on the IN-C development set[1] and test the learning rates 4.6×10^{-2}, 4.6×10^{-3}, 4.6×10^{-4} and 4.6×10^{-5}. We use the optimal

[1]We compare the results of computing the dev set on the 1, 3 and 5 severities versus the 1, 2, 3, 4 and 5 severities on our ResNeXt101 model in the Supplementary material.

hyperparameters to evaluate ENT and RPL on the full IN-C test set (with all severity levels).

UDA-SS models We trained the models using the scripts from the official code base at https://github.com/yueatsprograms/uda_release. We used the provided scripts for the cases: (a) source: CIFAR10, target: STL10 and (b) source: MNIST, target: MNIST-M. For the case (c) source: CIFAR10, target: CIFAR10-C, we used the hyperparameters from case (a) since this case seemed to be the closest match to the new setting. We think that the baseline performance of the UDA-SS models can be further improved with hyperparameter tuning.

DANN models To train models with the DANN-method, we used the PyTorch implementation of this paper at https://github.com/fungtion/DANN_py3. The code base only provides scripts and hyperparameters for the case (b) source: MNIST, target: MNIST-M. For the cases (a) and (c), we used the same optimizer and trained the model for 100 epochs. We think that the baseline performance of the DANN models can be further improved with hyperparameter tuning.

Preprocessing For IN, IN-R, IN-A and IN-D, we resize all images to 256×256 px and take the center 224×224 px crop. The IN-C images are already rescaled and cropped. We center and re-scale the color values with $\mu_{RGB} = [0.485, 0.456, 0.406]$ and $\sigma_{RGB} = [0.229, 0.224, 0.225]$. For the EfficientNet-L2, we follow the procedure in Xie et al. (2020a) and rescale all inputs to a resolution of 507×507 px and then center-crop them to 475×475 px.

Full list of used models

ImageNet scale models ImageNet trained models (ResNet50, DenseNet161, ResNeXt) are taken directly from torchvision (Marcel & Rodriguez, 2010). The model variants trained with DeepAugment and AugMix augmentations (Hendrycks et al., 2020b, 2020a) are taken from https://github.com/hendrycks/imagenet-r. The weakly-supervised ResNeXt101 model is taken from the PyTorch Hub. For EfficientNet (Tan & Le, 2019), we use the PyTorch re-implementation available at https://github.com/rwightman/gen-efficientnet-pytorch. This is a verified re-implementation of the original work by Xie et al. (2020a). We verify the performance on ImageNet, yielding a 88.23% top-1 accuracy and 98.546% top-5 accuracy which is within 0.2% points of the originally reported result (Xie et al., 2020a). On ImageNet-C, our reproduced baseline achieves 28.9% mCE vs. 28.3% mCE originally reported by Xie et al. (2020a). As noted in the re-implementation, this offset is possible due to minor differences in the pre-processing. It is possible that our adaptation results would improve further when applied on the original codebase by Xie *et al.*.

Table B.1: Model checkpoints used for our experiments. References: [1]Croce et al. (2020), [2]He et al. (2016c), [3]Huang et al. (2017), [4]Xie et al. (2017), [5]Hendrycks et al. (2020a), [6]Mahajan et al. (2018), [7]Xie et al. (2020a), [8]Caron et al. (2021b)

Model	Source
WideResNet(28,10)[1]	https://github.com/RobustBench/robustbench/tree/master/robustbench
WideResNet(40,2)+AugMix[1]	https://github.com/RobustBench/robustbench/tree/master/robustbench
ResNet50[2]	https://github.com/pytorch/vision/tree/master/torchvision/models
ResNeXt101, 32×8d[2]	https://github.com/pytorch/vision/tree/master/torchvision/models
DenseNet[3]	https://github.com/pytorch/vision/tree/master/torchvision/models
ResNeXt101, 32×8d[4]	https://pytorch.org/hub/facebookresearch_WSL-Images_resnext/
ResNet50+DeepAugment+AugMix[5]	https://github.com/hendrycks/imagenet-r
ResNext101[5]	https://github.com/hendrycks/imagenet-r
ResNext101 32×8d IG-3.5B[6]	https://github.com/facebookresearch/WSL-Images/blob/master/hubconf.py
Noisy Student EfficientNet-L2[7]	https://github.com/rwightman/gen-efficientnet-pytorch
ViT-S/16[8]	https://github.com/facebookresearch/dino

Small scale models We train the UDA-SS models using the original code base at https://github.com/yueatsprograms/uda_release, with the hyperparameters given in the provided bash scripts. For our DANN experiments, we use the PyTorch implementation at https://github.com/fungtion/DANN_py3. We use the hyperparameters in the provided bash scripts.

The following Table B.1 contains all models we evaluated on various datasets with references and links to the corresponding source code.

Category	Corruption	top1 error
Noise	Gaussian Noise	0.886428
	Shot Noise	0.894468
	Impulse Noise	0.922640
Blur	Defocus Blur	0.819880
	Glass Blur	0.826268
	Motion Blur	0.785948
	Zoom Blur	0.798360
Weather	Snow	0.866816
	Frost	0.826572
	Fog	0.819324
	Brightness	0.564592
	Contrast	0.853204
Digital	Elastic Transform	0.646056
	Pixelate	0.717840
	JPEG Compression	0.606500
Hold-out Noise	Speckle Noise	0.845388
Hold-out Digital	Saturate	0.658248
Hold-out Blur	Gaussian Blur	0.787108
Hold-out Weather	Spatter	0.717512

Table B.2: AlexNet top1 errors on ImageNet-C

Detailed and additional Results on IN-C

Definition of the mean Corruption Error (mCE)

The established performance metric on IN-C is the mean Corruption Error (mCE), which is obtained by normalizing the model's top-1 errors with the top-1 errors of AlexNet across the C=15 test corruptions and S=5 severities:

$$\text{mCE(model)} = \frac{1}{C}\sum_{c=1}^{C}\frac{\sum_{s=1}^{S}\text{err}_{c,s}^{\text{model}}}{\sum_{s=1}^{S}\text{err}_{c,s}^{\text{AlexNet}}}. \tag{B.11}$$

The AlexNet errors used for normalization are shown in Table B.2.

Detailed results for tuning epochs and learning rates

We tune the learning rate for all models and the number of training epochs for all models except the EfficientNet-L2. In this section, we present detailed results for tuning these hyperparameters for all considered models. The best hyperparameters that we found in this analysis, are summarized in Table B.7.

Table B.3: mCE in % on the IN-C dev set for ENT and RPL for different numbers of training epochs when adapting the affine batch norm parameters of a ResNet50 model.

criterion	ENT			RPL		
lr	10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}
epoch						
0	60.2	60.2	60.2	60.2	60.2	60.2
1	54.3	**50.0**	72.5	57.4	51.1	52.5
2	52.4	50.9	96.5	55.8	49.6	57.4
3	51.5	51.0	112.9	54.6	49.2	64.2
4	51.0	52.4	124.1	53.7	49.0	71.0
5	50.7	53.5	131.2	52.9	**48.9**	76.3

Table B.4: mCE ($\searrow$) in % on the IN-C dev set for different learning rates for EfficientNet-L2. We favor $q = 0.8$ over $q = 0.7$ due to slightly improved robustness to changes in the learning rate in the worst case error setting.

lr (4.6 $\times$)	base	10^{-3}	10^{-4}	10^{-5}	10^{-6}
ENT	25.5	87.8	25.3	**22.2**	24.1
$\text{RPL}_{q=0.7}$	25.5	60.3	**21.3**	23.3	n/a
$\text{RPL}_{q=0.8}$	25.5	58.2	**21.4**	23.4	n/a

Detailed results for all IN-C corruptions

We outline detailed results for all corruptions and models in Table B.8. Performance across the severities in the dataset is depicted in Figure B.4. All detailed results presented here are obtained by following the model selection protocol outlined in the main text.

Detailed results for the CIFAR10-C and UDA adaptation

Ablation over the hyperparameter q for RPL

For RPL, we must choose the hyperparameter q. We performed an ablation study over q and show results in Table B.11, demonstrating that RPL is robust to the choice of q, with slight preference to higher values. Note: In the initial parameter sweep for this paper, we only compared $q = 0.7$ and $q = 0.8$. Given the result in Table B.11, it could be interesting to re-run the models in Table 1 of the main paper with $q = 0.9$, which could yield another (small) improvement in mCE.

Self-learning outperforms Test-Time Training (Sun et al., 2019b)

Sun et al. (2019b) use a ResNet18 for their experiments on ImageNet and only evaluate their method on severity 5 of IN-C. To enable a fair comparison, we trained a ResNet18 with both hard labeling and RPL and compare the efficacy of both methods to Test-Time Training in Table B.12. For both hard labeling and RPL, we use the hyperparameters

Table B.5: mCE in % on IN-C dev for entropy minimization for different learning rates and training epochs for ResNeXt101. (div.=diverged)

ENT	Baseline			IG-3.5B			DAug+AM		
lr 2.5 × epoch	1e-4	1e-3	5e-3	1e-4	1e-3	5e-3	1e-4	1e-3	5e-3
BASE	53.6	53.6	53.6	47.4	47.4	47.4	37.4	37.4	37.4
1	**43.0**	92.2	div.	40.9	40.4	58.6	**35.4**	46.4	div.
2	44.8	118.4	div.	39.8	41.5	69.5	35.5	90.8	div.
3	45.4	131.9	div.	39.3	42.6	76.1	35.5	122.5	div.
4	46.7	div.	div.	**39.1**	44.2	84.3	35.6	133.8	div.

Table B.6: mCE in % on IN-C dev for robust pseudo-labeling for different learning rates and training epochs for ResNeXt101. (div.=diverged)

RPL	Baseline			IG-3.5B			DAug+AM		
lr 2.5× epoch	1e-4	1e-3	5e-3	1e-4	1e-3	5e-3	1e-4	1e-3	5e-3
BASE	53.6	53.6	53.6	47.4	47.4	47.4	37.4	37.4	37.4
1	43.4	51.3	div.	45.0	39.9	43.6	35.3	35.1	79.1
2	42.3	63.2	div.	43.4	**39.3**	48.2	34.9	35.6	121.2
3	42.0	72.6	div.	42.4	39.4	52.9	34.7	40.1	133.5
4	**42.0**	72.6	div.	42.4	39.4	52.9	**34.7**	40.1	133.5

Table B.7: The best hyperparameters for all models that we found on IN-C. For all models, we fine-tune only the affine batch normalization parameters and use $q = 0.8$ for RPL. The small batchsize for the EfficientNet model is due to hardware limitations.

Model	Method	Learning rate	batch size	number of epochs
vanilla ResNet50	ENT	1×10^{-3}	128	1
vanilla ResNet50	RPL	1×10^{-3}	128	5
vanilla ResNeXt101	ENT	2.5×10^{-4}	128	1
vanilla ResNeXt101	RPL	2.5×10^{-4}	128	4
IG-3.5B ResNeXt101	ENT	2.5×10^{-4}	128	4
IG-3.5B ResNeXt101	RPL	2.5×10^{-3}	128	2
DAug+AM ResNeXt101	ENT	2.5×10^{-4}	128	1
DAug+AM ResNeXt101	RPL	2.5×10^{-4}	128	4
EfficientNet-L2	ENT	4.6×10^{-5}	8	1
EfficientNet-L2	RPL	4.6×10^{-4}	8	1

we found for the vanilla ResNet50 model and thus, we expect even better results for hyperparameters tuned on the vanilla ResNet18 model and following our general hyperparameter search protocol.

While all methods (self-learning and TTT) improve the performance over a simple vanilla ResNet18, we note that even the very simple baseline using hard labeling already outperfoms Test-Time Training; further gains are possible with RPL. The result highlights the importance of simple baselines (like self-learning) when proposing new domain adaptation schemes. It is likely that many established DA techniques more complex than the basic self-learning techniques considered in this work will even further improve over TTT and other adaptation approaches developed exclusively in robustness settings.

We report better accuracy numbers achieved with self-learning compared to online TTT, but note that they adapt to a single test sample while we make use of batches of data. Due to the single-image approach, they utilize GN instead of BN, and this may explain the performance gap to a certain degree as we show that BN is more effective for test-time adaptation compared to GN, even though the un-adapted BN models are

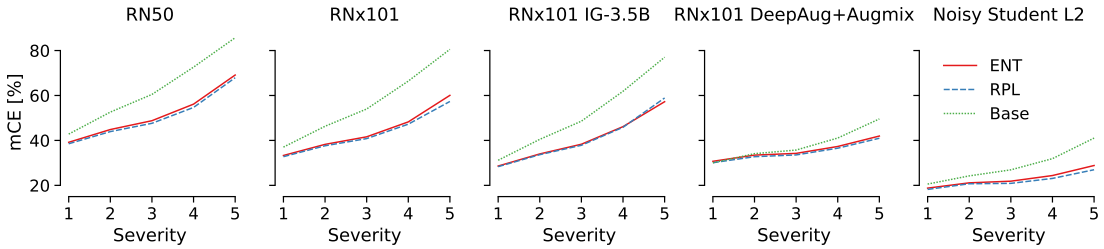


Figure B.4: Severity-wise mean corruption error (normalized using the *average* AlexNet baseline error for each corruption) for ResNet50 (RN50), ResNext101 (RNx101) variants and the Noisy Student L2 model. Especially for more robust models (DeepAugment+Augmix and Noisy Student L2), most gains are obtained across higher severities 4 and 5. For weaker models, the baseline variant (Base) is additionally substantially improved for smaller corruptions.

less robust compared to the un-adapted GN models, as also noted by Sun et al. (2019b). Further, Sun et al. (2019b) optimize all model parameters while we only optimize the affine BN parameters which works better.

Comparison to Meta Test-Time Training (Bartler et al., 2022)

Bartler et al. (2022) report an error of 24.4% as their top result on CIFAR10-C which is far worse than our top results of 8.5% with an AugMix trained model, or 13.3% with a vanilla trained model, but a different architecture: they used a WRN-26-1 while we report results with a WRN-40-2. Thus, to make the comparison more fair, we tested our approach on their model architecture.

A direct comparison is not straight-forward since Bartler et al. (2022) trained their models using Keras while our code-base is in PyTorch. Therefore, we first trained a baseline model in PyTorch on clean CIFAR10 using their architecture with the standard and widely used CIFAR10 training code available at https://github.com/kuangliu/pytorch-cifar (1.9k forks, 4.8k stars); the baseline test accuracy on clean CIFAR10 using the architecture of Bartler et al. (2022) is at 94.3%. As a second step, we adapted the baseline model with ENT and RPL on CIFAR10-C. Please see Table B.13 for our results. BN adaptation is not possible for the model used by Bartler et al. (2022) since they use GroupNorm instead of BatchNorm. Due to the usage of GroupNorm, it is not evident whether the affine GroupNorm parameters should be adapted, or all model parameters. Thus, we tested both, and report the results for both adaptation mechanisms.

We find that adaptation of affine GN layers works better than full model adaptation, consistent with our results for the adaptation of BN layers. As the gains due to ENT and RPL seem lower than for our WRN-40-2 architecture, we hypothesize that the issue lies in the GN layers as the presence of GN instead of BN layers is the main difference between our WRN-40-2 and the new WRN-26-1 model. Therefore, we trained another WRN-26-1 model with BN layers instead of GN, and adapted this model using BN, ENT and RPL. The baseline accuracy on clean CIFAR10 of WRN-26-1-BN is 95.04%. Indeed, we find that adapting the model with BN instead of GN layers leads to much larger

Table B.8: Detailed results for each corruption along with mean corruption error (mCE) as reported in Table 3.1 in the main paper. We show (unnormalized) top-1 error rate averaged across 15 test corruptions along with the mean corruption error (mCE: which is normalized). Hyperparameter selection for both ENT and RPL was carried out on the dev corruptions as outlined in the main text. Mismatch in baseline mCE for EfficientNet-L2 can be most likely attributed to pre-processing differences between the original tensorflow implementation (Xie et al., 2020a) and the PyTorch reimplementation we employ. We start with slightly weaker baselines for ResNet50 and ResNext101 than (Schneider et al., 2020a): ResNet50 and ResNext101 results are slightly worse than previously reported results (typically 0.1% points) due to the smaller batch size of 128 and 32. Smaller batch sizes impact the quality of re-estimated batch norm statistics when computation is performed on the fly (Schneider et al., 2020a), which is of no concern here due to the large gains obtained by pseudo-labeling.

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg	**mCE**
ResNet50																
Baseline (Schneider et al., 2020a)																62.2
Baseline (ours)	57.2	59.5	60.0	61.4	62.3	51.3	49.5	54.6	54.1	39.3	29.1	46.7	41.4	38.2	41.8	62.8
ENT	45.5	45.5	46.8	48.4	48.7	40.0	40.3	42.0	46.6	33.2	28.1	42.4	35.2	32.2	35.1	51.6
RPL	44.2	44.4	45.5	47.0	47.4	38.8	39.2	40.7	46.2	32.5	27.7	42.7	34.6	31.6	34.4	50.5
ResNeXt101 Baseline																
Baseline (Schneider et al., 2020a)																56.7
Baseline (ours)	52.8	54.1	54.0	55.4	56.8	46.7	46.6	48.5	49.4	36.6	25.4	42.8	37.8	32.5	36.7	56.8
ENT	40.5	39.5	41.4	41.6	43.0	34.1	34.5	35.0	39.4	28.5	24.0	33.8	30.3	27.2	30.5	44.3
RPL	39.4	38.9	39.8	40.3	41.0	33.4	33.8	34.6	38.7	28.0	23.7	31.4	29.8	26.8	30.0	43.2
ResNeXt101 IG-3.5B																
Baseline (Schneider et al., 2020a)																51.6
Baseline (ours)	50.7	51.5	53.1	54.2	55.5	45.5	44.7	41.7	42.0	28.1	20.1	33.8	35.4	27.8	33.9	51.8
ENT	38.6	38.3	40.4	41.4	41.5	33.8	33.6	32.2	34.6	24.1	19.7	26.3	27.6	24.2	27.9	40.8
RPL	39.1	39.2	40.8	42.1	42.4	33.7	33.5	31.8	34.7	23.9	19.6	26.1	27.5	23.8	27.5	40.9
ResNeXt101 DeepAug+Augmix																
Baseline (Schneider et al., 2020a)																38.0
Baseline (ours)	30.0	30.0	30.2	32.9	35.5	28.9	31.9	33.3	32.8	29.5	22.6	28.4	31.2	23.0	26.5	38.1
ENT	28.7	28.5	29.0	29.8	30.9	26.9	28.0	29.3	30.5	26.2	23.2	26.3	28.5	23.7	26.0	35.5
RPL	28.1	27.8	28.3	29.1	30.1	26.3	27.4	28.8	29.8	25.9	22.7	25.6	27.9	23.2	25.4	34.8
Noisy Student L2																
Baseline (Xie et al., 2020a)																28.3
Baseline (ours)	21.6	22.0	20.5	23.9	40.5	19.8	23.2	22.8	26.9	21.0	15.2	21.2	24.8	17.9	18.6	28.9
ENT	18.5	18.7	17.4	18.8	23.4	16.9	18.8	17.1	19.6	16.8	14.1	16.6	19.6	15.8	16.5	23.0
RPL	17.8	18.0	17.0	18.1	21.4	16.4	17.9	16.4	18.7	15.7	13.6	15.6	19.2	15.0	15.6	22.0

gains, which is consistent with findings by Schneider et al. (2020a) who found that BN adaptation outperforms non-adapted models trained with GN. Thus, we conclude that self-learning techniques work better with models which have BN layers and less well with models with GN layers. For both model types (with GN or with BN layers), ENT works better than RPL which is consistent with our other results on small-scale datasets.

Overall, our best result for the model architecture used by Bartler et al. (2022) (after replacing GN layers with BN layers) et al. is 13.1% which is much lower than their best result of 24.4%. Even when using GN layers, our best top-1 error is 18.0% which is significantly lower than the best result of Bartler et al. (2022).

Table B.9: Detailed results for each corruption along with mean error on CIFAR10-C as reported in Table 3.2 in the main paper.

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg	**avg**
WRN-28-10 vanilla																
Baseline	53.0	41.2	44.7	18.5	49.0	22.3	24.4	18.1	25.0	11.2	6.7	17.4	16.2	28.0	22.4	26.5
BN adapt	20.8	17.6	22.7	8.1	28.4	10.9	9.2	14.2	13.0	8.7	6.8	8.5	13.5	12.1	21.0	14.4
ENT	18.5	15.9	20.6	7.8	25.5	10.6	8.5	13.1	12.3	8.3	6.9	8.0	12.6	11.1	18.9	13.3
RPL	19.1	21.4	16.3	8.1	26.4	8.9	10.9	13.7	12.9	6.9	13.1	19.7	8.2	11.4	8.7	13.7
WRN-40-2 AM																
Baseline	19.1	14.0	13.3	6.3	17.1	7.9	7.0	10.4	10.6	8.5	5.9	9.7	9.2	16.8	11.9	11.2
BN adapt	14.1	11.9	13.9	7.2	17.6	8.7	7.9	10.8	10.6	9.0	6.8	9.0	10.9	10.1	14.0	10.8
ENT	10.8	9.1	10.9	6.0	13.4	7.2	6.3	8.4	7.8	7.1	5.7	7.1	9.2	7.4	11.2	8.5
RPL	11.5	11.6	9.6	6.2	14.2	6.5	7.4	8.8	8.2	6.0	9.5	11.9	7.9	8.0	7.6	9.0
WRN-26-16 UDA-SS																
Baseline	26.0	24.7	19.3	22.4	56.2	32.4	32.1	31.7	31.2	26.6	15.8	20.4	26.3	21.5	28.9	27.7
BN adapt	20.5	19.0	15.6	13.5	43.1	19.4	18.3	23.1	21.2	16.2	12.8	14.1	20.9	16.7	23.4	19.9
ENT	16.9	16.7	12.3	11.3	37.6	15.6	14.8	18.3	18.2	13.4	10.8	11.9	17.9	14.4	20.9	16.7
RPL	18.1	17.1	13.2	11.9	41.5	17.3	16.1	20.4	19.1	14.5	11.8	12.7	18.8	18.1	22.6	18.2
WRN-26-16 vanilla																
Baseline	50.8	46.9	39.3	15.9	44.2	20.8	18.8	14.9	17.8	4.8	13.6	20.4	19.0	25.0	10.0	24.2
BN adapt	18.6	20.4	15.6	6.1	24.9	7.4	8.3	11.6	10.8	5.3	11.4	18.9	6.8	9.9	6.8	12.2
ENT	16.7	18.4	13.9	5.7	23.0	6.8	7.9	10.8	9.9	5.0	10.8	17.3	6.4	9.1	6.4	11.2
RPL	16.1	18.0	13.6	6.6	23.2	8.7	9.3	11.9	11.1	6.4	11.7	17.0	7.4	9.0	7.2	11.8

Table B.10: Detailed results for the UDA methods reported in Table 3.2 of the main paper.

	Baseline	BN adapt	RPL	ENT
UDA CIFAR10→STL10, top1 error on target [%](↘)				
WRN-26-16 UDA-SS	28.7	24.6	22.9	21.8
WRN-26-16 DANN	25.0	25.0	24.0	23.9
UDA MNIST→MNIST-M, top1 error on target [%](↘)				
WRN-26-16 UDA-SS	4.8	3.9	2.4	2.0
WRN-26-2 DANN	11.4	6.2	5.2	5.1

Effect of batch size and linear learning rate scaling

How is self-learning performance affected by batch size constraints? We compare the effect of different batch sizes and linear learning rate scaling. In general, we found that affine adaptation experiments on ResNet50 scale can be run with batch size 128 on a Nvidia V100 GPU (16GB), while only batch size 96 experiments are possible on RTX 2080 GPUs.

The results in Table B.14 show that for a ResNet50 model, higher batch size yields a generally better performance.

Performance over different seeds in a ResNet50 on ImageNet-C

To limit the amount of compute, we ran RPL and ENT for our vanilla ResNet50 model three times with the optimal hyperparameters. The averaged results, displayed as

Table B.11: ImageNet-C dev set mCE in %, vanilla ResNet50, batch size 96. We report the best score across a maximum of six adaptation epochs.

q	0.5	0.6	0.7	0.8	0.9
mCE (dev)	49.5	49.3	49.2	49.2	49.1

Table B.12: Comparison of hard-pseudo labeling and robust pseudo-labeling to Test-Time Training (Sun et al., 2019b): Top-1 error for a ResNet18 and severity 5 for all corruptions. Simple hard pseudo-labeling already outperforms TTT, robust pseudo labeling over multiple epochs yields additional gains.

	gauss	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*snow*	*frost*	*fog*	*bright*	*contrast*	*elastic*	*pixelate*	*jpeg*	**Avg**
vanilla ResNet18	98.8	98.2	99.0	88.6	91.3	88.8	82.4	89.1	83.5	85.7	48.7	96.6	83.2	76.9	70.4	85.4
Test-Time Training	73.7	71.4	73.1	76.3	93.4	71.3	66.6	64.4	81.3	52.4	41.7	**64.7**	55.7	52.2	55.7	66.3
hard PL, (1 epoch)	73.2	70.8	73.6	76.5	75.6	63.9	56.1	59.0	**65.9**	48.4	39.7	85.2	50.4	47.0	51.5	62.5
ENT(1 epoch)	72.8	69.8	73.2	77.2	75.7	63.1	55.5	58.0	68.1	48.0	39.8	92.7	49.6	46.4	51.3	62.8
RPL (4 epochs)	**71.3**	**68.3**	**71.7**	**76.2**	**75.6**	**61.5**	**54.4**	**56.9**	67.1	**47.3**	**39.3**	93.2	**48.9**	**45.7**	**50.4**	**61.9**

"mean (unbiased std)" are:

Self-learning as continuous test-time adaptation

We test our method on continuous test-time adaptation where the model adapts to a continuous stream of data from the same domain. In Fig. B.5, we display the error of the Noisy Student L2 model while it is being adapted to ImageNet-C and ImageNet-R. The model performance improves as the model sees more data from the new domain. We differentiate continuous test-time adaptation from the online test-time adaptation setting (Zhang et al., 2021) where the model is adapted to each test sample individually, and reset after each test sample.

Table B.13: Detailed results for our comparison to MT3 (Bartler et al., 2022)

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg	**avg**
WRN-26-1-GN vanilla																
Baseline (Bartler et al., 2022)	50.5	47.2	56.1	23.7	51.7	24.3	26.3	25.6	34.4	28.1	13.5	25.0	27.4	55.8	29.8	34.6
adapted (Bartler et al., 2022)	30.1	29.5	41.8	15.6	33.7	22.8	18.7	20.2	18.8	24.1	13.8	22.4	23.7	27.6	22.7	24.4
WRN-26-1-GN vanilla																
Baseline [ours]	39.1	32.0	30.2	10.9	31.9	13.5	13.3	14.1	16.7	10.6	7.3	9.4	14.1	16.1	20.6	18.6
ENT [GN layers, ours]	41.6	30.7	32.8	9.7	30.9	11.8	11.4	13.5	14.9	10.0	7.2	8.9	13.2	13.3	19.8	18.0
RPL [GN layers, ours]	39.3	31.6	30.6	10.6	31.6	13.0	12.6	14.0	16.1	10.4	7.3	9.2	13.9	15.4	20.4	18.4
ENT [full model, ours]	61.2	46.0	37.6	9.1	30.2	11.1	10.4	13.1	14.4	10.0	7.2	8.7	13.1	11.9	19.7	20.3
RPL [full model, ours]	54.1	37.3	33.0	10.2	31.5	11.9	12.4	13.6	15.3	7.3	13.6	20.4	9.2	13.6	10.2	19.6
WRN-26-1-BN vanilla																
Baseline [ours]	55.7	44.8	43.1	15.5	44.2	20.5	21.1	17.2	20.7	6.6	15.6	21.3	23.6	24.7	11.8	25.8
BN adapt [ours]	28.0	28.9	24.2	10.5	31.5	12.7	14.0	18.4	18.1	9.4	17.1	26.9	13.2	15.3	12.7	18.7
ENT [BN layers, ours]	17.9	19.6	14.9	8.1	23.2	9.1	10.4	13.1	13.2	7.2	13.2	18.0	9.9	10.5	8.9	**13.1**
RPL [BN layers, ours]	21.3	22.0	17.9	9.3	26.6	10.5	11.8	15.1	15.0	7.8	14.7	20.8	12.1	11.8	10.4	15.1

Table B.14: ImageNet-C dev set mCE for various batch sizes with linear learning rate scaling. All results are computed for a vanilla ResNet50 model using RPL with $q = 0.8$, reporting the best score across a maximium of six adaptation epochs.

batch size	16	32	64	80	96	128
learning rate ($\times 10^{-3}$)	0.125	0.250	0.500	0.625	0.750	1
dev mCE	53.8	51.0	49.7	49.3	49.2	48.9

Table B.15: ImageNet-C performance for three seeds on a ResNet50 for ENT and RPL.

ResNet50 + self-learning	mCE on IN-C dev [%]	mCE on IN-C test [%]
ENT	50.0 (0.04)	51.6 (0.04)
RPL	48.9 (0.02)	50.5 (0.03)

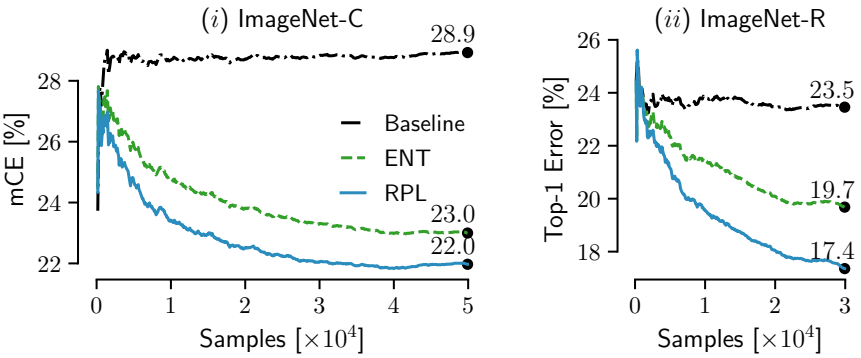


Figure B.5: Evolution of error during online adaptation for EfficientNet-L2.

Table B.16: Statistics of one-to-many mappings from IN-D to ImageNet.

Number of IN classes one IN-D class is mapped to	1	2	3	4	5	6	7	8	13	28	39	132
Frequency of these mappings	102	32	13	3	5	1	1	2	1	2	1	1

Detailed and additional Results on IN-D

Detailed protocol for label mapping from DomainNet to ImageNet

The mapping was first done by comparing the class labels in DomainNet and the synset labels on ImageNet. Afterwards, the resulting label maps were cleaned manually, because simply comparing class label strings resulted in imperfect matches. For example, images of the class "hot dog" in DomainNet were mapped to the class "dog" in IN. Another issue is that IN synset labels of different animal species do not contain the animal name in the text label, e.g., the class "orangutan, orang, orangutang, Pongo pygmaeus" does not contain the word "monkey" and we had to add this class to the hierarchical class "monkey" manually. We verified the mappings by investigating the class-confusion matrix of the true DomainNet class and the predicted ImageNet classes remapped to DomainNet on the "Real" domain, and checked that the predictions lay on the main diagonal, indicating that ImageNet classes have not been forgotten. The statistics for the mappings are shown in Table B.16. Most IN-D classes (102) are mapped to one single ImageNet class. A few IN-D classes are mapped to more than 20 ImageNet classes: the IN-D classes "monkey" and "snake" are mapped to 28 ImageNet monkey and snake species classes, the IN-D class "bird" is mapped to 39 ImageNet bird species classes, and the IN-D class "dog" is mapped to 132 ImageNet dog breed classes.

We have considered max-pooling predictions across all sub-classes according to the ImageNet class hierarchy following the approach of Taori et al. (2020) for Youtube-BB and ImageNet-Vid (Recht et al., 2020). However, Radford et al. (2021) note that the resulting mappings are sometimes "much less than perfect", thus, we decided to clean the mappings ourselves to increase the mappings' quality. The full dictionary of the mappings will be released alongside the code.

Evaluation protocol on IN-D

The domains in IN-D differ in terms of their difficulty for the studied models. Therefore, to calculate an aggregate score, we propose normalizing the error rates by the error achieved by AlexNet on the respective domains to calculate the mean error, following the approach in Hendrycks and Dietterich (2019a) for IN-C. This way, we obtain the aggregate score mean Domain Error (mDE) by calculating the mean over different

Table B.17: mDE in % on IN-D for different model selection strategies.

model	model selection	
	L1outCV	IN-C dev
ResNet50 $\text{RPL}_{q=0.8}$	81.3	76.1
ResNet50 ENT	82.4	77.3
EfficientNet-L2 ENT	69.2	66.8
EfficientNet-L2 $\text{RPL}_{q=0.8}$	69.1	67.2

domains,

$$\text{DE}_d^f = \frac{E_d^f}{E_d^{\text{AlexNet}}}, \qquad \text{mDE} = \frac{1}{D}\sum_{d=1}^{D} E_d^f, \tag{B.12}$$

where E_d^f is the top-1 error of a classifier f on domain d.

Leave-one-out-cross-validation For all IN-D results we report in this paper, we chose the hyperparameters on the IN-C dev set. We tried a different model selection scheme on IN-D as a control experiment with "Leave one out cross-validation" (L1outCV): with a round-robin procedure, we choose the hyperparameters for the test domain on all other domains. We select the same hyperparameters as when tuning on the "dev" set: For the ResNet50 model, we select over the number of training epochs (with a maximum of 7 training epochs) and search for the optimal learning rate in the set [0.01, 0.001, 0.0001]. For the EfficientNet-L2 model, we train only for one epoch as before and select the optimal learning rate in the set $[4.6 \times 10^{-3}, 4.6 \times 10^{-4}, 4.6 \times 10^{-5}, 4.6 \times 10^{-6}]$. This model selection leads to worse results both for the ResNet50 and the EfficientNet-L2 models, highlighting the robustness of our model selection process, see Table B.17.

Detailed results for robust ResNet50 models on IN-D

We show detailed results for all models on IN-D for vanilla evaluation (Table B.18) BN adaptation (Table B.19), $\text{RPL}_{q=0.8}$ (Table B.20) and ENT(Table B.21). For $\text{RPL}_{q=0.8}$ and ENT, we use the same hyperparameters that we chose on our IN-C 'dev' set. This means we train the models for 5 epochs with $\text{RPL}_{q=0.8}$ and for one epoch with ENT.

We evaluate the pre-trained and public checkpoints of SIN (Geirhos et al., 2019), ANT (Rusak et al., 2020), ANT+SIN (Rusak et al., 2020), AugMix (Hendrycks et al., 2020b), DeepAugment (Hendrycks et al., 2020a) and DeepAug+Augmix (Hendrycks et al., 2020a) in the following tables.

The summary results for all models are shown in Table B.22.

We show the top-1 error for the different IN-D domains versus training epochs for a vanilla ResNet50 in Fig. B.6. We indicate the epochs 1 and 5 at which we extract the errors with dashed black lines.

Table B.18: Top-1 error on IN-D in % as obtained by robust ResNet50 models. For reference, we also show the mCE on IN-C and the top-1 error on IN-R. See main test for model references.

Model	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	mDE	IN-C	IN-R
vanilla	76.0	89.6	65.1	99.2	40.1	82.0	88.2	76.7	63.9
SIN	71.3	88.6	62.6	97.5	40.6	77.0	85.6	69.3	58.5
ANT	73.4	88.9	63.3	99.2	39.9	80.8	86.9	62.4	61.0
ANT+SIN	68.4	88.6	60.6	95.5	40.8	70.3	83.1	60.7	53.7
AugMix	70.8	88.6	62.1	99.1	39.0	78.5	85.4	65.3	58.9
DeepAugment	72.0	88.8	61.4	98.9	39.4	78.5	85.6	60.4	57.8
DeepAug+Augmix	68.4	88.1	58.7	98.2	39.2	75.2	83.4	53.6	53.2

Table B.19: Top1 error on IN-D in % as obtained by state-of-the-art robust ResNet50 models and batch norm adaptation, with a batch size of 128. See main text for model references.

Model	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	mDE
vanilla	70.2	88.2	63.5	97.8	41.1	78.3	80.2
SIN	67.3	89.7	62.2	97.2	44.0	75.2	79.6
ANT	69.2	89.4	63.0	97.5	42.9	79.5	80.7
ANT+SIN	64.9	88.2	60.0	96.8	42.6	73.0	77.8
AugMix	66.9	88.1	61.2	97.1	40.4	75.0	78.4
DeepAugment	66.6	89.7	60.0	97.2	42.5	75.1	78.8
DeepAug+Augmix	61.9	85.7	57.5	95.3	40.2	69.2	74.9

Detailed results for the EfficientNet-L2 Noisy Student model on IN-D

We show the detailed results for the EfficientNet-L2 Noisy Student model on IN-D in Table B.23.

Detailed results on the error analysis on IN-D

Analysing frequently predicted classes We analyze the most frequently predicted classes on IN-D by a vanilla ResNet50 and show the results in Fig. B.7. The colors of the bars indicate whether the predicted class is part of the IN-D dataset: "blue" indicates that the class appear in the IN-D dataset, while "orange" means that the class is not present in IN-D.

We make several interesting observations: First, we find most errors interpretable: it makes sense that a ResNet50 assigns the label "comic book" to images from the "Clipart" or "Painting" domains, or "website" to images from the "Infograph" domain, or "envelope" to images from the "Sketch" domain. Second, on the hard domain "Quickdraw", the ResNet50 mostly predicts non-sensical classes that are not in IN-D, mirroring its almost chance performance on this domain. Third, we find no systematic errors on the "Real" domain which is expected since this domain should be similar to IN.

Analyzing the correlation between the performance on IN-C/IN-R and IN-D We show the Spearman's rank correlation coefficients for errors on ImageNet-D correlated to errors

Table B.20: Top-1 error on IN-D in % as obtained by state-of-the-art robust ResNet50 models and $RPL_{q=0.8}$. See main text for model references.

Model	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	mDE
vanilla	63.6	85.1	57.8	99.8	37.3	73.0	76.1
SIN	60.8	86.4	56.0	99.0	37.8	67.0	76.8
ANT	63.4	86.3	57.7	99.2	37.7	71.0	78.1
ANT+SIN	61.5	86.4	56.8	97.0	39.0	67.1	76.1
AugMix	59.7	83.4	54.1	98.2	35.6	70.1	74.6
DeepAugment	58.1	84.6	53.3	99.0	36.2	64.2	74.8
DeepAug+Augmix	57.0	83.2	53.4	99.1	36.5	61.3	72.6

Table B.21: Top-1 error on IN-D in % as obtained by state-of-the-art robust ResNet50 models and ENT. See main text for references to the used models.

Model	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	mDE
vanilla	65.1	85.8	59.2	98.5	38.4	75.8	77.3
SIN	62.1	87.0	57.3	99.1	39.0	68.6	75.5
ANT	64.2	86.9	58.7	97.1	38.8	72.8	76.5
ANT+SIN	62.2	86.8	57.7	95.8	40.1	68.7	75.2
AugMix	60.2	84.6	55.8	97.6	36.8	72.0	74.4
DeepAugment	59.5	85.7	54.4	98.0	37.1	66.4	73.3
DeepAug+Augmix	58.4	84.3	54.7	98.5	38.1	63.6	72.7

on ImageNet-R and ImageNet-C for robust ResNet50 models in Fig. B.7. For this correlation analysis, we take the error numbers from Table B.18. We find the correlation to be high between most domains in IN-D and IN-R which is expected since the distribution shift between IN-R and IN is similar to the distribution shift between IN-D and ImageNet. The only domain where the Spearman's rank correlation coefficient is higher for IN-C is the "Real" domain which can be explained with IN-C being closer to real-world data than IN-R. Thus, we find that the Spearman's rank correlation coefficient reflects the similarity between different datasets.

Filtering predictions on IN-D that cannot be mapped to ImageNet We perform a second analysis: We filter the predicted labels according to whether they can be mapped to

Table B.22: mDE on IN-D in % as obtained by robust ResNet50 models with a baseline evaluation, batch norm adaptation, $RPL_{q=0.8}$ and ENT. See main text for model references.

	mDE on IN-D (↘)			
Model	Baseline	BN adapt	$RPL_{q=0.8}$	ENT
vanilla	88.2	80.2	76.1	77.3
SIN	85.6	79.6	76.8	75.5
ANT	86.9	80.7	78.1	76.5
ANT+SIN	**83.1**	77.8	76.1	75.2
AugMix	85.4	78.4	74.6	74.4
DeepAugment	85.6	78.8	74.8	73.3
DeepAugment+Augmix	83.4	**74.9**	**72.6**	**72.7**

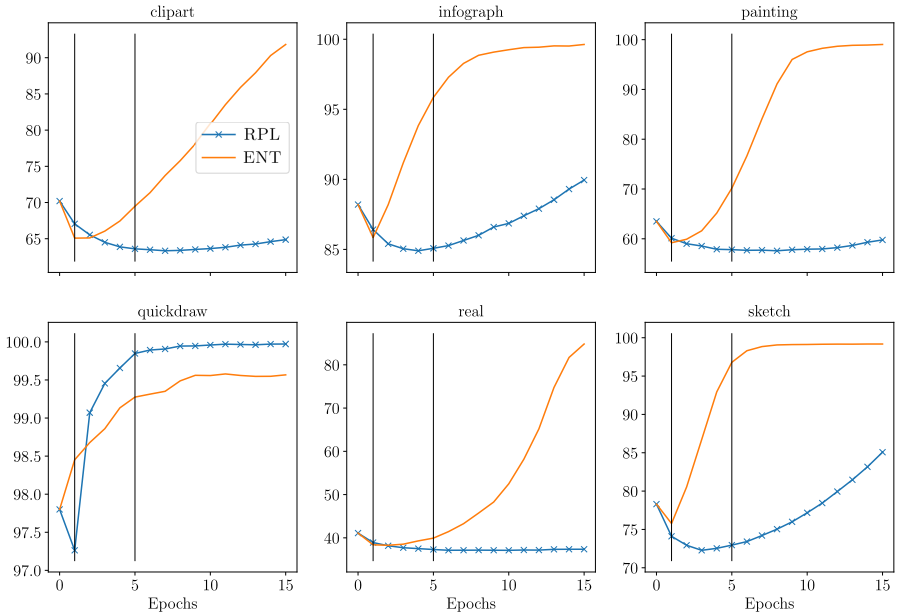


Figure B.6: Top-1 error for the different IN-D domains for a ResNet50 and training with $\text{RPL}_{q=0.8}$ and ENT. We indicate the epochs at which we extract the test errors by the dashed black lines (epoch 1 for ENT and epoch 5 for $\text{RPL}_{q=0.8}$).

Table B.23: Top-1 error (↘) on IN-D in % for EfficientNet-L2

Domain	Baseline	ENT	RPL
Clipart	45.0	39.8	**37.9**
Infograph	**77.9**	91.3	94.3
Painting	42.7	41.7	**40.9**
Quickdraw	**98.4**	99.4	99.4
Real	29.2	28.7	**27.9**
Sketch	56.4	**48.0**	51.5
mDE	67.2	**66.8**	67.2

IN-D and report the filtered top-1 errors as well as the percentage of filtered out inputs in Table B.24. We note that for the domains "infograph" and "quickdraw", the ResNet50 predicts labels that cannot be mapped to IN-D in over 70% of all cases, highlighting the hardness of these two domains.

Filtering labels and predictions on IN that cannot be mapped to ImageNet-D To test for possible class-bias effects, we test the performance of a ResNet50 model on ImageNet classes that can be mapped to IN-D and report the results in Table B.24.

First, we map IN labels to IN-D to make the setting as similar as possible to our experiments on IN-D and report the top-1 error (12.1%). This error is significantly lower compared to the top-1 error a ResNet50 obtains following the standard evaluation protocol (23.9%). This can be explained by the simplification of the task: While in IN there are 39 bird classes, these are all mapped to the same hierarchical class in

Table B.24: top-1 error on IN and different IN-D domains for different settings: left column: predicted labels that cannot be mapped to IN-D are filtered out, right column: percentage of filtered out labels.

Dataset	top-1 error on filtered labels in %	percentage of rejected inputs
IN val	13.4	52.7
IN-D real	17.2	27.6
IN-D clipart	59.0	59.0
IN-D infograph	59.3	74.6
IN-D painting	39.5	42.4
IN-D quickdraw	96.7	76.1
IN-D sketch	65.6	47.9

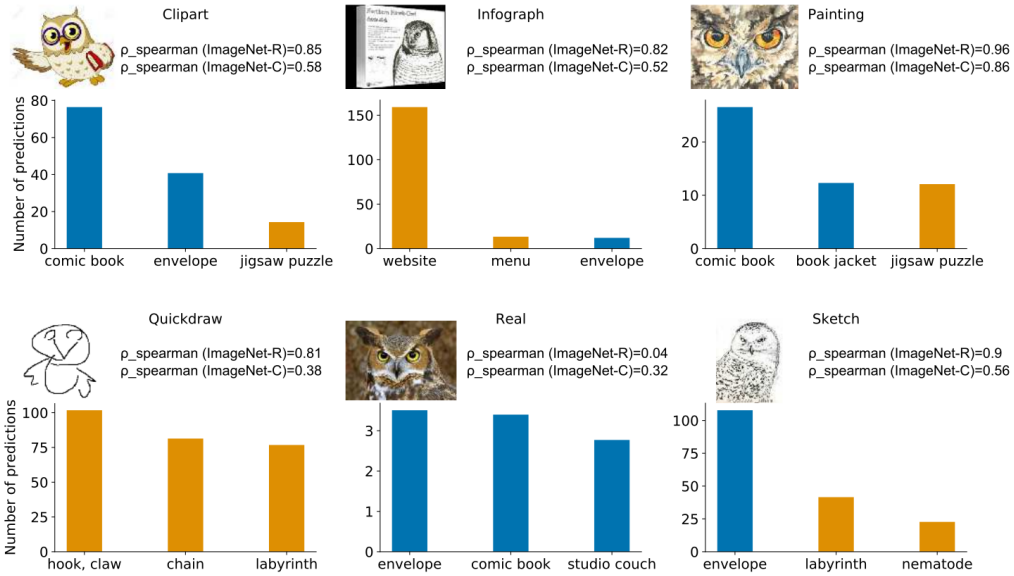


Figure B.7: Systematic predictions of a vanilla ResNet50 on IN-D for different domains. The colors of the bars indicate whether the predicted class is part of the IN-D dataset: "blue" indicates that the class appear in the IN-D dataset, while "orange" means that the class is not present in IN-D. The errors on IN-R are strongly correlated to errors on IN-D for most domains, except for the "Real" domain.

IN-D. Therefore, the classes in IN-D are more dissimilar from each other than in IN. Additionally, there are only 164 IN-D classes compared to the 1000 IN classes, raising the chance level prediction.

If we further only accept predictions that can be mapped to IN-D, the top-1 error is slightly increased to 13.4%. In total, about 52.7% of all images in the IN validation set cannot be mapped to IN-D.

Table B.25: top-1 error on IN-D by AlexNet which was used for normalization.

Dataset	top-1 error in %
IN-D real	54.887
IN-D clipart	84.010
IN-D infograph	95.072
IN-D painting	79.080
IN-D quickdraw	99.745
IN-D sketch	91.189

Top-1 error on IN-D for AlexNet

We report the top-1 error numbers on different IN-D as achieved by AlexNet in Table B.25. We used these numbers for normalization when calculating mDE.

Additional experiments

Beyond ImageNet classes: Self-learning on WILDS

The WILDS benchmark (Koh et al., 2021) is comprised of ten tasks to test domain generalization, subpopulation shift, and combinations thereof. In contrast to the setting considered here, many of the datasets in WILDS mix several 10s or 100s domains during test time.

The Camelyon17 dataset in WILDS contains histopathological images, with the labels being binary indicators of whether the central 32×32 region contains any tumor tissue; the domain identifies the hospital that the patch was taken from. Camelyon17 contains three different test splits with different domains and varying difficulty levels. For evaluation, we took the pretrained checkpoint[2] for a DenseNet121 model (Huang et al., 2017) and verified the reported baseline performance numbers. We adapt the models using ENT or RPL for a maximum of 10 epochs using learning rates $\{3 \times 10^{-5}, 3 \times 10^{-4}, \ldots 3 \times 10^{-1}\}$. The best hyperparameter is selected according to OOD Validation accuracy.

The RxRx1 dataset in WILDS contains RGB images of cells obtained by fluorescent microscopy, with the labels indicating which of the 1,139 genetic treatments (including no treatment) the cells received; the domain identifies the batch in which the imaging experiment was run. The RxRx1 dataset contains three test splits, however, unlike Camelyon17, in all of the splits the domains are mixed. For evaluation, we took the pretrained checkpoint[3] for a ResNet50 model and verified the reported baseline performance numbers. We adapt the models using ENT or RPL for a maximum of 10 epochs using base learning rates $\{6.25 \times 10^{-6}, 6.25 \times 10^{-5}, \ldots 6.25 \times 10^{-2}\}$, which are scaled to the admissible batch size for single GPU adaptation using linear scaling. The best hyperparameter is selected according to OOD Validation accuracy.

The FMoW dataset in WILDS contains RGB satellite images, with the labels being one of 62 building or land use categories; the domain specifies the year in which the image was taken and its geographical region (Africa, the Americas, Oceania, Asia, or Europe). The FMoW dataset contains four test splits for different time periods, for which all regions are mixed together. For evaluation, we took the pretrained checkpoint[4] for a DenseNet121 model and verified the reported baseline performance numbers. We adapt the models using ENT or RPL for a maximum of 10 epochs using learning rates $\{5.0 \times 10^{-6}, 5.0 \times 10^{-5}, \ldots 5.0 \times 10^{-2}\}$. The best hyperparameter is selected according to OOD Validation accuracy.

While we see improvements on Camelyon17, neither BN adaptation nor self-learning can improve performance on RxRx1 or FMoW. Initial experiments on PovertyMap and iWildsCam also do not show improvements with self-learning. We hypothesize that

[2] https://worksheets.codalab.org/worksheets/0x00d14c55993548a1823a710642f6d608, camelyon17_erm_densenet121_seed0

[3] https://worksheets.codalab.org/bundles/0x7d33860545b64acca5047396d42c0ea0

[4] https://worksheets.codalab.org/bundles/0x20182ee424504e4a916fe88c91afd5a2

Table B.26: Self-learning can improve performance on WILDS if a systematic shift is present—on Camelyon17, the ood validation and test sets are different hospitals, for example. On datasets like RxRx1 and FMoW, we do not see an improvement, most likely because the ood domains are shuffled, and a limited amount of images exist for each test domain.

	Top-1 accuracy [%]		
	Validation (ID)	Validation (OOD)	Test (OOD)
Camelyon17			
Baseline	81.4	88.7	63.1
BN adapt	97.8 (+16.4)	90.9 (+2.2)	88.0 (+24.9)
ENT	97.6 (+16.2)	92.7 (+4.0)	91.6 (+28.5)
RPL	97.6 (+16.2)	93.0 (+4.3)	91.0 (+27.9)
RxRx1			
Baseline	35.9	19.1	29.7
BN adapt	35.0 (-0.9)	19.1 (0.0)	29.4 (-0.3)
ENT	34.8 (-1.1)	19.2 (+0.1)	29.4 (-0.3)
RPL	34.8 (-1.1)	19.2 (+0.1)	29.4 (-0.3)
FMoW			
Baseline	60.5	59.2	52.9
BN adapt	59.9 (-0.6)	57.6 (-1.6)	51.8 (-1.1)
ENT	59.9 (-0.6)	58.5 (-0.7)	52.2 (-0.7)
RPL	59.8 (-0.7)	58.6 (-0.6)	52.1 (-0.8)

the reason lies in the mixing of the domains: Both BN adaptation and our self-learning methods work best on systematic domain shifts. These results support our claim that self-learning is effective, while showing the important limitation when applied to more diverse shifts.

Small improvements on BigTransfer models with Group normalization layers

We evaluated BigTransfer models (Kolesnikov et al., 2020) provided by the timm library (Wightman, 2019). A difference to the ResNet50, ResNeXt101 and EfficientNet models is the use of group normalization layers, which might influence the optimal method for adaptation—for this evaluation, we followed our typical protocol as performed on ResNet50 models, and used affine adaptation.

For affine adaptation, a distilled BigTransfer ResNet50 model improves from 49.6 % to 48.4 % mCE on the ImageNet-C development set, and from 55.0 % to 54.4 % mCE on the ImageNet-C test set when using RPL ($q = 0.8$) for adaptation, at learning rate 7.5×10^{-4} at batch size 96 after a single adaptation epoch. Entropy minimization did not further improve results on the ImageNet-C test set. An ablation over learning rates and epochs on the dev set is shown in Table B.27, the final results are summarized in Table B.28.

Can Self-Learning improve over Self-Learning based UDA?

An interesting question is whether test-time adaptation with self-learning can improve upon self-learning based UDA methods. To investigate this question, we build upon

Table B.27: mCE in % on the IN-C dev set for ENT and RPL for different numbers of training epochs when adapting the affine batch norm parameters of a BigTransfer ResNet50 model.

criterion	ENT			RPL		
lr, 7.5 × epoch	10^{-5}	10^{-4}	10^{-3}	10^{-5}	10^{-4}	10^{-3}
0	49.63	49.63	49.63	49.63	49.63	49.63
1	49.44	50.42	52.59	49.54	48.89	48.95
2	49.26	50.27	56.47	49.47	**48.35**	50.77
3	49.08	52.18	60.06	49.39	48.93	51.45
4	48.91	52.03	60.50	49.31	50.01	51.53
5	**48.80**	51.97	62.91	49.24	49.96	51.34
6	48.83	52.10	62.96	49.16	49.71	51.19
7	48.83	52.10	62.96	49.16	49.71	51.19

Table B.28: mCE in % on the IN-C dev set for ENT and RPL for different numbers of training epochs when adapting the affine batch norm parameters of a BigTransfer ResNet50 model.

	dev mCE	test mCE
Baseline	49.63	55.03
ENT	48.80	56.36
RPL	**48.35**	**54.41**

French et al. (2017) and their released code base at https://github.com/Britefury/self-ensemble-visual-domain-adapt. We trained the Baseline models from scratch using the provided shell scripts with the default hyperparameters and verified the reported performance. For adaptation, we tested BN adaptation, ENT, RPL, as well as continuing to train in exactly the setup of French et al. (2017), but without the supervised loss. For the different losses, we adapt the models for a maximum of 10 epochs using learning rates $\{1 \times 10^{-5}, 1 \times 10^{-4}, \ldots, 1 \times 10^{-1}\}$.

Note that for this experiment, in contrast to any other result in this paper, **we purposefully do not perform proper hyperparameter selection based on a validation dataset**—instead we report the best accuracy across all tested epochs and learning rates to give an upper bound on the achievable performance for test-time adaptation.

As highlighted in Table B.29, none of the four tested variants is able to meaningfully improve over the baseline, corroborating our initial hypothesis that self-learning within a full UDA setting is the optimal strategy, if dataset size and compute permits. On the other hand, results like the teacher refinement step in DIRT-T (Shu et al., 2018) show that with additional modifications in the loss function, it might be possible to improve over standard UDA with additional adaptation at test time.

Table B.29: Test-time adaptation marginally improves over self-ensembling.

	Baseline	BN adapt	ENT	RPL	Self-ensembling loss
MNIST→SVHN					
MT+TF	33.88	34.44	34.87	35.09	33.27
MT+CT*	32.62	34.11	34.25	34.21	33.36
MT+CT+TF	41.59	41.93	41.95	41.95	42.70
MT+CT+TFA	30.55	32.53	32.54	32.55	30.84
SVHN-specific aug.	97.05	96.82	96.91	96.87	97.12
MNIST→USPS					
MT+TF	98.01	97.91	97.96	97.91	98.16
MT+CT*	88.34	88.39	88.54	88.39	88.44
MT+CT+TF	98.36	98.41	98.41	98.41	98.50
MT+CT+TFA	98.45	98.45	98.45	98.45	98.61
SVHN→MNIST					
MT+TF	98.49	98.47	98.49	98.47	99.40
MT+CT*	88.34	88.36	88.36	88.36	89.36
MT+CT+TF	99.51	99.49	99.5	99.49	99.57
MT+CT+TFA	99.56	99.57	99.57	99.57	99.58
SVHN-specific aug.	99.52	99.49	99.5	99.49	99.65
USPS→MNIST					
MT+TF	92.79	92.62	92.62	92.66	93.08
MT+CT*	99.11	99.13	99.14	99.13	99.21
MT+CT+TF	99.41	99.42	99.45	99.42	99.52
MT+CT+TFA	99.48	99.54	99.57	99.54	99.54

Software stack

We use different open source software packages for our experiments, most notably Docker (Merkel, 2014), scipy and numpy (Virtanen et al., 2020), GNU parallel (Tange, 2011), Tensorflow (Abadi et al., 2016), PyTorch (Paszke et al., 2017), timm (Wightman, 2019), Self-ensembling for visual domain adaptation (French et al., 2017), the WILDS benchmark (Koh et al., 2021), and torchvision (Marcel & Rodriguez, 2010).

C

RDumb: A simple approach that questions our progress in continual test-time adaptation

2D Example Experiments and Analysis

In order to better understand collapse, we constructed a simple 2D Gaussian binary classification example.

Data. The 2D datasets are constructed as follows: the data is sampled by drawing sampled from two Gaussian blobs with identical variance corresponding to the two classes $\mathcal{N}(\mu_i, \Sigma)$. For the corrupted case, the data is then rotated by an angle θ and combined with additional additive Gaussian noise $\mathcal{N}(0, \Sigma_{\text{corrupt}})$. Finally, both in the clean and the corrupt data case, the data is rotated by an angle of $-\frac{\pi}{4}$ (which minimizes the effect of the batch normalization in the model):

$$
\begin{aligned}
Y &\sim \text{Ber}(0.5) \\
X &\sim \mathcal{N}(\mu_Y, \Sigma) \\
X_{\text{clean}} &= R_{-\frac{\pi}{4}} X \\
X_{\text{corrupt}} &\sim \mathcal{N}(R_{\theta_{c_1}} X_{\text{clean}}, R_{-\frac{\pi}{4}}^{\top} R_{\theta_{c_2}}^{\top} \tilde{\Sigma} R_{\theta_{c_2}} R_{-\frac{\pi}{4}}) \\
\text{with } \Sigma &= \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}, \quad \sigma_1 \gg \sigma_2
\end{aligned}
$$

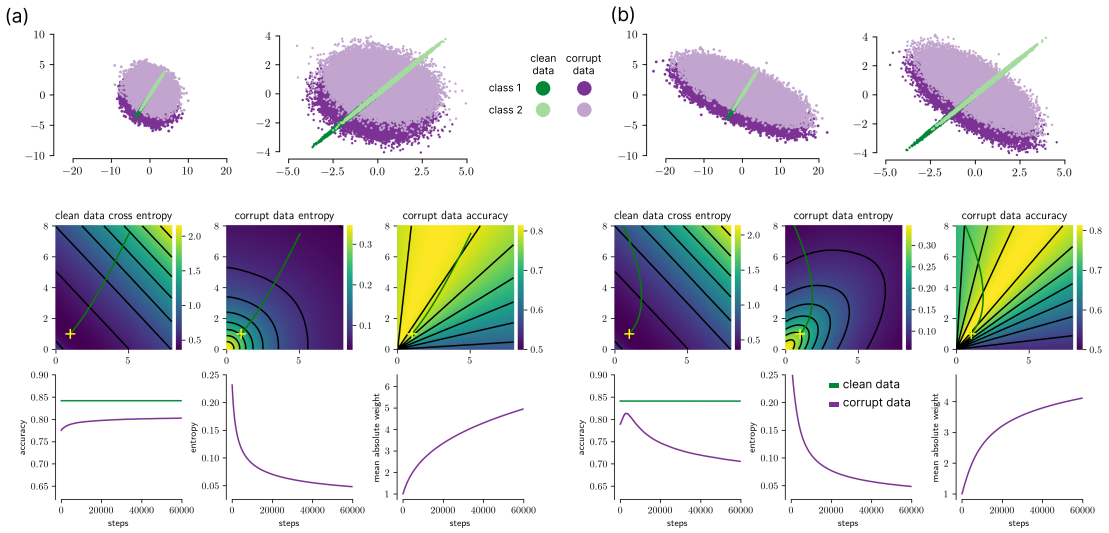


Figure C.1: Theoretical analysis of adaptation under distribution shift. Collapsing or non-collapsing behavior of entropy minimization can be reproduced with a simple 2d Gaussian binary classification example, a domain shift which slightly rotates the data and adds Gaussian noise, and a model which consists of a batch norm layer followed by logistic regression. **Top:** clean and corrupt data for two classes before (a) and after (b) batch norm. **Middle:** learning dynamics of entropy minimization in the 2d adaptation parameter space starting from initial parameters (yellow marker) over time. **Bottom:** accuracy, entropy, and size of adaptation weights over time.

$$\tilde{\Sigma} = \begin{pmatrix} \tilde{\sigma}_1 & 0 \\ 0 & \tilde{\sigma}_2 \end{pmatrix}, \quad \tilde{\sigma}_2 \geq \tilde{\sigma}_1$$

$$R_\theta = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

Note that for the clean data, we allocate most of the variance to the class dimension, σ_1. On the corrupted data, we add noise primarily perpendicular to the class dimension ($\tilde{\sigma}_2$), which is the main defining factor whether or not we observe collapsing behavior. $\theta_{c_1} \neq \theta_{c_2}$ serves to break the symmetry between signal and noise in the data, which results in Tent starting to deviate towards the noise direction.

In our example, we parametrize this model as follows: In condition 1, $\mu_i = (\pm 1, 0)$ and $\Sigma^2 = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0.03 \end{smallmatrix}\right)$, $\theta = -\frac{\pi}{9}$ and $\Sigma^2_{\text{corrupt}} = R_{\theta_c}^T \left(\begin{smallmatrix} 0.25 & 0 \\ 0 & 4 \end{smallmatrix}\right) R_{\theta_c}$, where R_{θ_c} is the rotation matrix for an angle of $\theta_c = -\frac{\pi}{6}$. In condition 2, everything is the same except for $\Sigma^2_{\text{corrupt}} = R_{\theta_c}^T \left(\begin{smallmatrix} 0.25 & 0 \\ 0 & 25 \end{smallmatrix}\right) R_{\theta_c}$. In both conditions, we sample 300 000 data points which are equally distributed over both classes.

Model. The classification model consumes the dataset of shape $N \times 2$ and consists of a batchnorm layer ($\epsilon = 0$) followed by a fully connected layer with two input channels, one output channels, no bias term and a sigmoid nonlinearity. Because our data is always centered, we do not learn the offset parameter of the affine adaptation in the

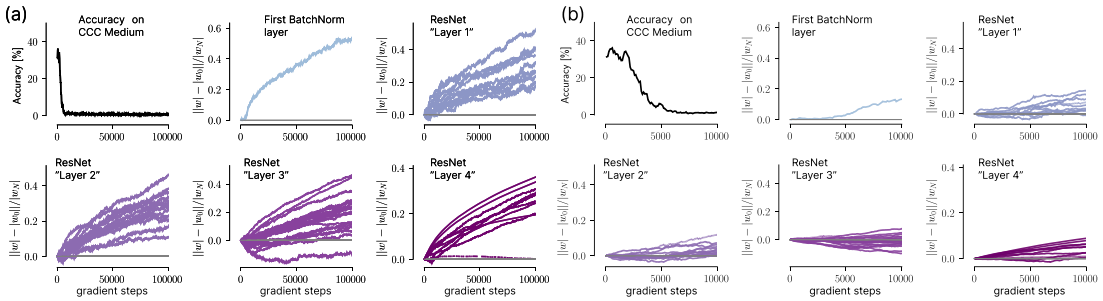


Figure C.2: Analysis of entropy minimization collapse on real data. (a) Consistent with the theoretical analysis in Figure C.1, we find that the adaptation weights in all layers increase over time continually, even long after the collapse as indicated by Accuracy on CCC-Medium has happened. (b) shows a zoomed-in view where this increase is not yet apparent, well after the collapse.

batchnorm layer, but only the scale parameter.

Model training. The model is trained to minimize the binary cross-entropy on the clean data. We use batch gradient descent on the whole 300,000 sample dataset with a learning rate of 0.1 and no momentum. We decay the learning rate by a factor of 0.1 after 1000 and 2000 steps and stop training after 3000 steps.

Model Learning and Adaptation. We adapt the model on the corrupted data using Tent. More precisely, we optimize the scale parameter of the batch norm layer to minimize the entropy of the predictions using SGD with a learning rate of 0.01 and no momentum. We process the whole dataset in one batch and adapt for 60 000 steps.

Results. In the toy model, simple cases emerge where the loss does not result in collapse, or vice versa (Figure C.1a and Figure C.1b, respectively), mainly depending on the relation of signal and noise variances and directions.

The toy example furthermore predicts that the adapted parameters of a model should grow on the long run and indeed we were able to find exactly this effect when running ETA on a ResNet50 on CCC-Medium (Figure C.2), suggesting that our minimal setup successfully reproduces the relevant aspects of the large scale case. However, the weight explosion becomes apparent only after the collapse happens, hence weight regularization is not enough to avoid the collapse (Figure C.2(b)).

In the Figure C.1(a) example, we find the direction of high target domain performance and stay there; in this case entropy minimization is stable. In the Figure C.1(b) experiment, the domain shift adds more noise nearly orthogonal to the signal direction, which entropy minimization tries to use for making high confidence predictions: we still initially find the direction of high target domain performance, but traverse this region and continue into a direction of low entropy and low accuracy. This shows

that even in a linear example, entropy minimization can show initial performance improvement and then collapse.

Path Finding Algorithm

Algorithm 1 describes the pseudo code of the algorithm used to generate CCC. The algorithm is based on a set of Calibration Matrices, similar to the one shown in Figure 4.2. There exists a matrix m for every (n_a, n_b) pair, such that $m[i][j]$ is equal to accuracy of a pretrained ResNet-50 on the comination of noises (n_a, n_b), and severities $(i/5, j/5)$. We will release the full set of matrices upon publication.

Additionally, Algorithm 2 uses a function $MinValidPath(s_1, s_2)$: this function returns the minimum path that starts at (s_1, s_2) and ends at $(0, s_j)$ for some s_k. The cost of a path is simply the average of all entries along the path. The minimum path is defined as the path with a cost closest to b_a in absolute terms. Lastly, a path is only valid if it starts with s_2 equal to 0, every transition either decreases s_1 by 0.25, or increases s_2 by 0.25, and stops once s_1 is equal to 0.

Algorithm 1 Algorithm used to generate each split of CCC

Require: ba, k, T ▷ Baseline accuracy, transition speed, and total split size.
$t = 0$ ▷ Initialize the total images generated counter
$c_1, c_2 \sim \text{Uniform}(\{1 \dots 15\})$ ▷ Initialize the first two corruptions.
path $\leftarrow$ CalculatePath(c_1, c_2, ba) ▷ Calculate path along the noise pair with an average accuracy closest to ba.
loop
  $s_1, s_2 \leftarrow$ path$[p]$
  Subset $\sim$ Uniform(ImageNetVal)
  apply (c_1, s_1, c_2, s_2) to Subset
  save Subset
  $t \leftarrow k$
  if $t >= T$ **then** return
  end if
  if $p = \text{len(path)} - 1$ **then**
    $c_1 \leftarrow c_2$
    $c_2 \sim \text{Uniform}(\{1 \dots 14\})$
    path $\leftarrow$ CalculatePath(c_1, c_2, ba) ▷ Calculate new path along the new noise pair.
    $p \leftarrow 0$
  else
    $p \leftarrow p + 1$ ▷ Move to the next severity combination
  end if
end loop

The resulting dataset features transitions between two different noises like the ones seen in Figure C.3 (a). We additionally plot the accuracy of a pretrained ResNet-50, alongisde the severities of the different noises in Figure C.3 (b).

We additionally share the following metadata about the length of traversals in CCC-Easy/Medium/Hard:

Algorithm 2 CalculatePath

Require: c_1, c_2, ba ▷ The 2 noises, and the baseline accuracy
1: $m \leftarrow \text{CalibrationMatrix}[c_1][c_2]$
2: $\text{MinPath}, \text{MinCost} \leftarrow \text{MinValidPath}(0, 0, m, ba)$
3: **for** $s_1 \in 0.25, 0.5, 0.75, \ldots, 5$ **do**
4: $\quad m \leftarrow \text{MinValidPath}(1, 0, m, ba)$
5: $\quad \text{MinPath}, \text{MinCost} \leftarrow \text{MinValidPath}(s_1, 0, m, ba)$
6: **end for**
7: **return** *MinPath*

Table C.1: CCC traversal length statistics, for each CCC split.

	Min	Max	Mean	Median
CCC-Easy	11	36	22.8	23
CCC-Medium	21	41	33.9	34
CCC-Hard	41	41	41	41

CCC Plots

EATA Implementation and Ablations

Our implementation of EATA differs from the official implementation. The reason for this is that the official implementation uses clean ImageNet validation images to calculate the Fisher vector matrix for its regularizer[1]. This stands in contradiction with the method, which should not have access to the training distribution at test time.

Instead of using 2,000 ImageNet validation images, we calculate the Fisher matrix using the first 2,000 images in our data stream. We conduct a hyperparameter search on the weight regularizer tradeoff parameter β:

Table C.2: Accuracy of EATA on CIN-C holdout noises for different values of the weight regularizer loss.

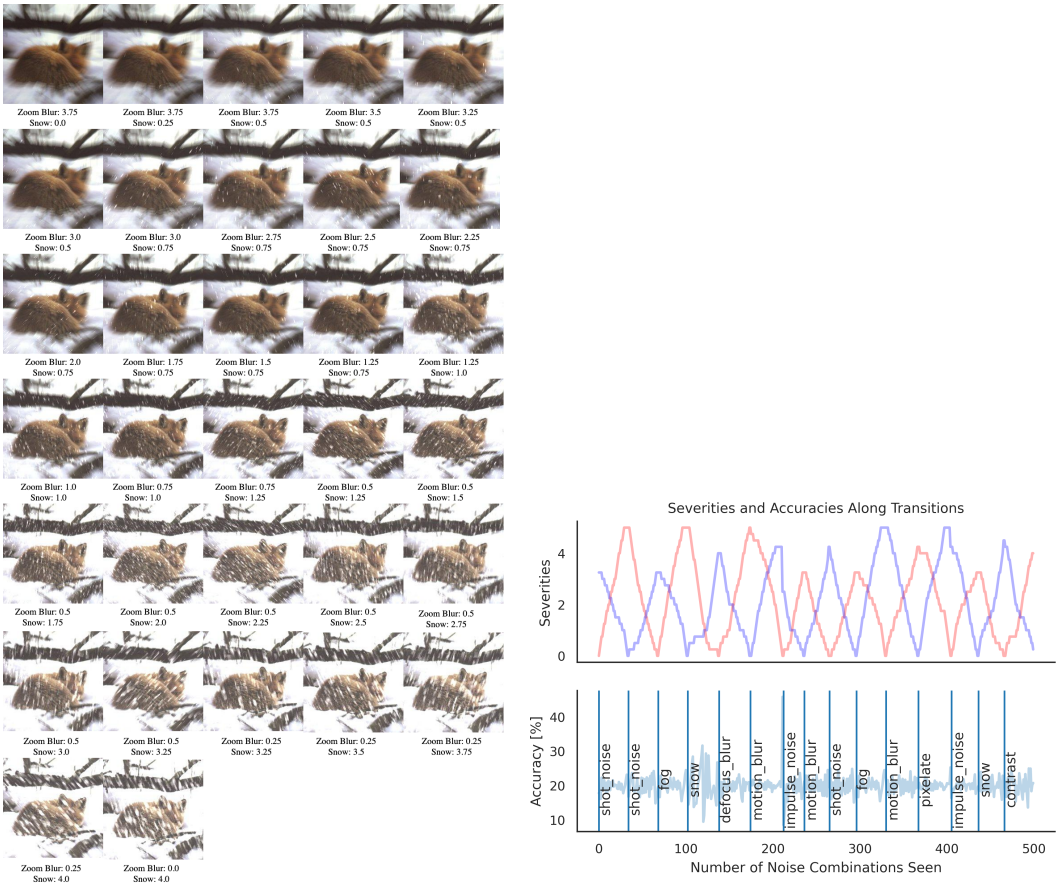
β	25	50	100	250	500	1000	1500	2000
Acc. [%]	46.5	46.9	**47.1**	46.7	46.1	45.6	44.8	44.0

Using the optimal value, 100 led to worse results than the default value, 2000, on CCC:

In the end, we used the original value of 2000, as that was optimal on the CCC dataset.

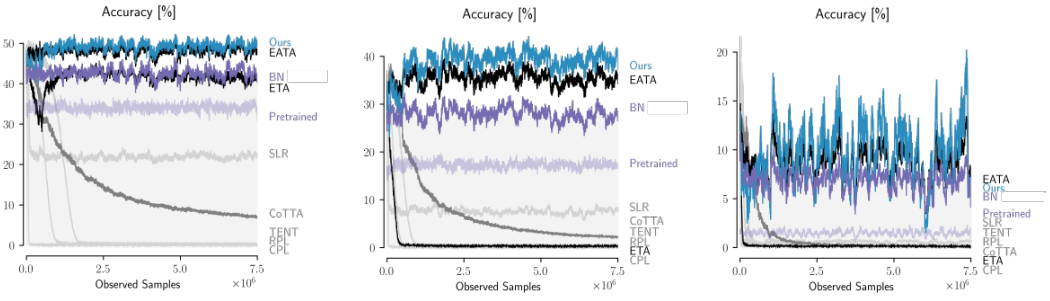
In addition, we conducted a hyperparameter search for EATA on a ViT backbone. As shown in Figure 4.4, EATA performs worse than a pretrained, non adapting baseline in this setting. To that end, we tried to stabilize the model by increasing the value of β,

[1] https://github.com/mr-eggplant/EATA/blob/f739b3668c/main.py#L144



(a) (b)

Figure C.3: **(a)** Visualization of CCC-Medium's smooth transition between Zoom Blur to Snow. Note: CCC additionally uses random flips and crops, which are not shown here. **(b)** As CCC transitions between noises, the severities of the first noise (red), and the second noise (blue) go up and down correspondingly, in order to keep the accuracy of a pretrained model stable.



(a) (b) (c)

Figure C.4: Adaptation performance of all evaluated models using a ResNet-50 backbone. (a) CCC Easy. (b) CCC Medium. (c) CCC Hard. For all subplots, model performances are averaged over the 9 runs of the respective difficulty level.

Table C.3: Accuracy of EATA on CIN-C holdout noises for different values of the Fisher alpha.

	CIN-C	CCC-Easy	CCC-Medium	CCC-Hard	CCC Avg
EATA-100	46.7	47.7	36.5	3.8	29.3
EATA-2000	41.8	48.2	35.4	8.7	30.8
Ours	46.5	49.3	38.9	9.6	32.6

the hyperparameter that controls the weight of the anti-forgetting regularizer. As with the previous experiment, the original value of 2000 is optimal.

Table C.4: Accuracy of EATA on CCC-Medium using a ViT backbone, for different values of the regularizer, β.

β	2000	3000	4000
Acc. [%]	**38.5**	27.8	16.5

Novelty of Resetting

Our work is the first to propose resetting to solve collapse in TTA methods. Notably, while prior work (Niu et al., 2022b; Wang et al., 2020b; Zhang et al., 2022) has briefly touched upon the concept of episodic resetting, the methodology and its application is significantly distinct and unrelated to collapse in TTA.

- **Tent** (Wang et al., 2020b) mentions episodic in the context of overfitting to a single sample in segmentation (similar to Zhang et al. (2022)'s overfitting to a single sample). Resetting here is unrelated to collapse, as the paper doesn't discuss collapse at all.

- Although **MEMO** (Zhang et al., 2022) uses resetting, it does so because it overfits to one image (and its augmentations) every step. MEMO doesn't discuss collapse or catastrophic forgetting. MEMO compares itself to a version of Tent that resets after every step (which they call Tent + episodic resetting), because MEMO without augmentations is similar to Tent + episodic resetting with a batch size of 1. (Note: MEMO is outperformed by BN/Tent/ETA when using the standard batch size of 64)

- **EATA** (Niu et al., 2022b) shows results for Tent + episodic resetting after every step in its tables, but provides no reasoning or discussion for doing this. Tent + episodic resetting is outperformed by regular Tent.

CIFAR10 Experiments

We conduct CIFAR10 experiments to show the need for ImageNet scale benchmarks. Tent, without an anti-collapse mechanism, does not collapse on CIFAR10-C, even after seeing 100 million images Figure C.5a,b. Like ImageNet, CIFAR10-C's noises also exhibit high variance in difficulty (Figure C.5c).

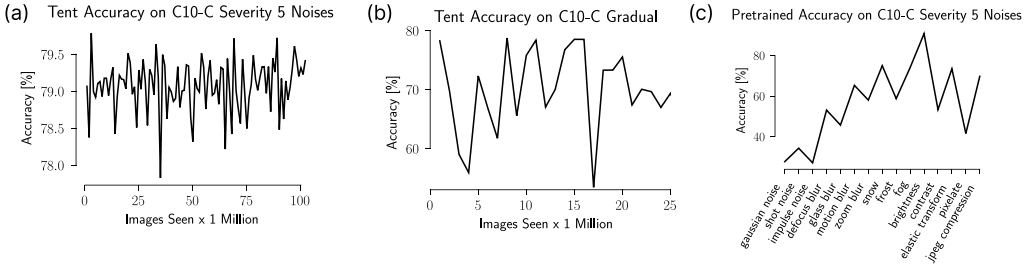


Figure C.5: **(a)** When tested on an infinite concatenation of severity 5 noises, Tent does not collapse even after seeing 100M CIFAR scale images. **(b)** Tent does not collapse to chance level when tested on a long term variant of CIFAR10-C gradual. **(c)** CIFAR10-C exhibits great variations between individual corruptions, similar to ImageNet-C.

Compute details

We conduct all experiments on Nvidia RTX 2080 TI GPUs with 12GB memory per device. All experiments except our study on larger models were conducted on a single GPU. For CoTTA experiments, we use data parallel training on 2 GPUs. A bulk of the compute spent for this work was on computing baseline accuracies on the calibration dataset, which contains 463M images.

Software and Dataset Licenses

Datasets

- ImageNet-C (Hendrycks & Dietterich, 2019b): Creative Commons Attribution 4.0 International,
  https://zenodo.org/record/2235448
- ImageNet-C (Hendrycks & Dietterich, 2019b), code for generating corruptions: Apache License 2.0
  https://github.com/hendrycks/robustness
- ImageNet-3D-CC (Kar et al., 2022): CC-BY-NC 4.0 License
  https://github.com/EPFL-VILAB/3DCommonCorruptions

Models

- PyTorch's (Paszke et al., 2019a) Backbones
  https://pytorch.org/vision/stable/models.html
- Adaptive BN (Nado et al., 2020; Schneider et al., 2020a):
  Apache License 2.0, https://github.com/bethgelab/robustness
- Tent (Wang et al., 2020b): MIT License,
  https://github.com/DequanWang/tent
- RPL (Rusak et al., 2021): Apache License 2.0,
  https://github.com/bethgelab/robustness
- CoTTA (Wang et al., 2022): MIT License,
  https://github.com/qinenergy/cotta
- CPL (Goyal et al., 2022): MIT License,
  https://github.com/locuslab/tta_conjugate
- EATA (Niu et al., 2022b): MIT License
  https://github.com/mr-eggplant/EATA

D

Pretraining boosts out-of-domain robustness for pose estimation

Additional information on the Horse-10 dataset

The table lists the following statistics: labeled frames, scale (nose-to-eye distance in pixels), and whether a horse was within domain (w.d) or out-of-domain (o.o.d.) for each shuffle.

Learning schedule cross validation

Because of the extensive resources required to cross validate all models, we only underwent the search on MobileNetV2s 0.35 and 1.0, ResNet 50, and EfficientNets B0, B3, and B5 for the pretraining and from scratch variants. For all other models, the parameters from the most similar networks were used for training (i.e. EfficientNet-B1 used the parameters for EfficientNet-B0). The grid search started with the highest possible initial learning rate that was numerically stable for each model; lower initial learning rates were then tested to fine tune the schedule. Zero and nonzero decay target levels were tested for each initial learning rate. In addition to the initial learning rates and decay targets, we experimented with shortening the cosine decay and incorporating restarts. All cross validation experiments were performed on the three splits with 50% of the data for training.

For training, a cosine learning rate schedule, as in (Kornblith et al., 2019b) with ADAM optimizer (Kingma & Ba, 2014) and batchsize 8 was used. For the learning schedules we use the following abbreviations: Initial Learning Rates (ILR) and decay target (DT).

Horse Identifier	samples	nose-eye dist	shuffle 1	shuffle 2	shuffle 3
BrownHorseinShadow	308	22.3	o.o.d	o.o.d	w.d.
BrownHorseintoshadow	289	17.4	o.o.d	o.o.d	o.o.d
Brownhorselight	306	15.57	o.o.d	w.d.	o.o.d
Brownhorseoutofshadow	341	16.22	o.o.d	w.d.	w.d.
ChestnutHorseLight	318	35.55	w.d.	w.d.	o.o.d
Chestnuthorseongrass	376	12.9	o.o.d	w.d.	w.d.
GreyHorseLightandShadow	356	14.41	w.d.	w.d.	o.o.d
GreyHorseNoShadowBadLight	286	16.46	w.d.	o.o.d	w.d.
TwoHorsesinvideobothmoving	181	13.84	o.o.d	o.o.d	w.d.
Twohorsesinvideoonemoving	252	16.51	w.d.	w.d.	w.d.
Sample1	174	24.78	o.o.d	o.o.d	o.o.d
Sample2	330	16.5	o.o.d	o.o.d	o.o.d
Sample3	342	16.08	o.o.d	o.o.d	o.o.d
Sample4	305	18.51	o.o.d	o.o.d	w.d.
Sample5	295	16.89	w.d.	o.o.d	o.o.d
Sample6	376	12.3	o.o.d	o.o.d	o.o.d
Sample7	262	18.52	w.d.	o.o.d	o.o.d
Sample8	388	12.5	w.d.	w.d.	o.o.d
Sample9	359	12.43	o.o.d	o.o.d	o.o.d
Sample10	235	25.18	o.o.d	o.o.d	o.o.d
Sample11	256	19.16	o.o.d	w.d.	o.o.d
Sample12	288	17.86	w.d.	o.o.d	w.d.
Sample13	244	25.78	w.d.	w.d.	w.d.
Sample14	168	25.55	o.o.d	o.o.d	o.o.d
Sample15	154	26.53	o.o.d	o.o.d	o.o.d
Sample16	212	15.43	o.o.d	o.o.d	o.o.d
Sample17	240	10.04	w.d.	o.o.d	o.o.d
Sample18	159	29.55	o.o.d	w.d.	o.o.d
Sample19	134	13.44	o.o.d	o.o.d	w.d.
Sample20	180	28.57	o.o.d	o.o.d	o.o.d
mean	270.47	18.89			
STD	73.04	6.05			

The tables below list the various initial learning rates explored during cross validation for each model with pretraining.

Model	ILR			
MobileNetV2-0.35	1e-2	5e-3	1e-3	5e-4
MobileNetV2-1.0	1e-2	5e-3	1e-3	5e-4
ResNet-50	1e-3	5e-4	1e-4	5e-5
EfficientNet-B0	2.5e-3	1e-3	7.5e-4	5e-4
EfficientNet-B3	1e-3	5e-4	1e-4	5e-5
EfficientNet-B5	5e-4	1e-4		

For the ImageNet pretrained case, the learning rate schedule without restarts was optimal on out of domain data, and the resulting optimal parameters are as follows:

Models	ILR & DT	
MobileNetV2s 0.35, 0.5	1e-2	0
MobileNetV2s 0.75, 1.0	1e-2	1e-4
ResNets 50, 101	1e-4	1e-5
EfficientNets B0, B1	5e-4	1e-5
EfficientNets B2,B3,B4	5e-4	0
EfficientNets B5,B6	5e-4	1e-5

The initial learning rates explored for the from scratch models during cross validation are as follows:

Model	ILR			
MobileNetV2 0.35	1e-2	5e-3	1e-3	5e-4
MobileNetV2 1.0	1e-1	1e-2	1e-3	1e-4
ResNet 50	1e-3	5e-4	1e-4	5e-5
EfficientNet-B0	1e-3	5e-4	1e-4	5e-5
EfficientNet-B3	1e-3	5e-4	1e-4	5e-5

For models trained from scratch, we found that using restarts lead to the best performance on out of domain data. The optimal learning rates found during the search are as follows:

Models	ILR & DT	
MobileNetV2s 0.35, 0.5	5e-2	5e-3
MobileNetV2s 0.75, 1.0	1e-2	0
ResNet 50	5e-4	5e-5
EfficientNets B0, B3	1e-3	0

Baseline Performance on Horse-30

For comparison to Horse-10, we provide the train and test normalized errors for models trained on Horse-30. Here, Horse-30 was split into 3 shuffles each containing a train/test split of 50% of the horse images. Compared to Horse-10, we train these models for twice as long (60,000 iterations) but with the same cross-validated cosine schedules from Horse-10. Errors below are averaged over the three shuffles.

	Horse-10 Errors		Horse-30 Errors	
Models	Train	Test	Train	Test
MobileNetV2 0.35	0.1342	0.1390	0.1545	0.1595
ResNet 50	0.0742	0.0815	0.0772	0.0825
EfficientNet-B4	0.0598	0.0686	0.0672	0.0750

Performance (PCK per bodypart) for all networks on Horse-10

The tables below show the PCK for several bodyparts for all backbones that we considered. They complete the abridged tables in the main text (Table 2 and 3) Thereby the bodyparts are abbreviated as follows: (FF=front foot; HF = Hind foot; HH = Hind Hock).

Table D.1: PCK@0.3 (%) for several bodyparts and all evaluated architectures on within domain horses.

	Nose	Eye	Shoulder	Wither	Elbow	NearFF	OffFF	Hip	NearHH	NearHF	OffHF
MobileNetV2 0.35	90.7	94.1	97.6	96.9	96.7	92.3	93.7	96.4	94.1	94.2	92.5
MobileNetV2 0.5	94.1	96.1	99.2	98.3	98.0	93.8	95.4	96.7	97.2	97.2	97.0
MobileNetV2 0.75	96.0	97.5	99.2	98.0	99.0	96.6	96.8	98.8	97.6	98.0	97.4
MobileNetV2 1.0	97.7	98.8	99.7	99.1	99.0	97.6	97.3	99.4	98.4	98.5	98.9
ResNet 50	99.9	100.0	99.8	99.9	99.8	99.8	99.6	99.9	99.9	99.6	99.8
ResNet 101	99.9	100.0	99.9	99.8	99.9	99.8	99.7	99.8	99.9	99.7	99.9
EfficientNet-B0	99.7	99.9	100.0	99.9	100.0	99.6	99.5	100.0	99.9	99.7	99.7
EfficientNet-B1	99.8	99.9	100.0	99.8	99.9	99.5	99.8	100.0	99.8	99.8	99.8
EfficientNet-B2	99.9	99.9	100.0	99.9	100.0	99.8	99.7	99.9	99.8	99.7	99.7
EfficientNet-B3	99.9	99.9	99.9	99.9	99.9	99.7	99.6	99.7	99.8	99.6	99.9
EfficientNet-B4	100.0	100.0	99.9	99.8	99.9	99.6	99.7	99.9	99.7	99.8	99.8
EfficientNet-B5	99.9	99.9	100.0	99.9	100.0	99.7	99.8	99.6	99.8	99.8	99.9
EfficientNet-B6	99.9	99.9	99.9	99.8	100.0	99.8	99.9	99.8	99.8	99.7	99.8

Table D.2: PCK@0.3 (%) for several bodyparts and all architectures on out-of-domain horses.

	Nose	Eye	Shoulder	Wither	Elbow	NearFF	OffFF	Hip	NearHH	NearHF	OffHF
MobileNetV2 0.35	45.6	53.1	65.5	68.0	69.1	56.4	57.6	65.9	65.9	60.5	62.5
MobileNetV2 0.5	52.7	61.0	76.7	69.7	78.3	62.9	65.4	73.6	70.8	68.1	69.7
MobileNetV2 0.75	54.2	65.6	78.3	73.2	80.5	67.3	68.9	80.0	74.1	70.5	70.2
MobileNetV2 1.0	59.0	67.2	83.8	79.7	84.0	70.1	72.1	82.0	79.9	76.0	76.7
ResNet 50	68.2	73.6	85.4	85.8	88.1	72.6	70.2	89.2	85.7	77.0	74.1
ResNet 101	67.7	72.4	87.6	86.0	89.0	79.9	78.0	92.6	87.2	83.4	80.0
EfficientNet-B0	60.3	62.5	84.9	84.6	87.2	77.0	75.4	86.7	86.7	79.6	79.4
EfficientNet-B1	67.4	71.5	85.9	85.7	89.6	80.0	81.1	86.7	88.4	81.8	81.6
EfficientNet-B2	68.7	74.8	84.5	85.2	89.2	79.7	80.9	88.1	88.0	82.3	81.7
EfficientNet-B3	71.7	76.6	88.6	88.7	92.0	80.4	81.8	90.6	90.8	85.0	83.6
EfficientNet-B4	71.1	75.8	88.1	87.4	91.8	83.3	82.9	90.8	90.3	86.7	85.5
EfficientNet-B5	74.8	79.5	89.6	89.5	93.5	82.2	84.1	91.8	90.9	86.6	85.2
EfficientNet-B6	74.7	79.7	90.3	89.8	92.8	83.6	84.4	92.1	92.1	87.8	85.3

CKA analysis of training & trained vs. from scratch networks

Figure D.1 shows a linear centered kernel alignment (CKA) (Kornblith et al., 2019a) comparison of representations for task-training vs. ImageNet trained (no task training) for ResNet-50

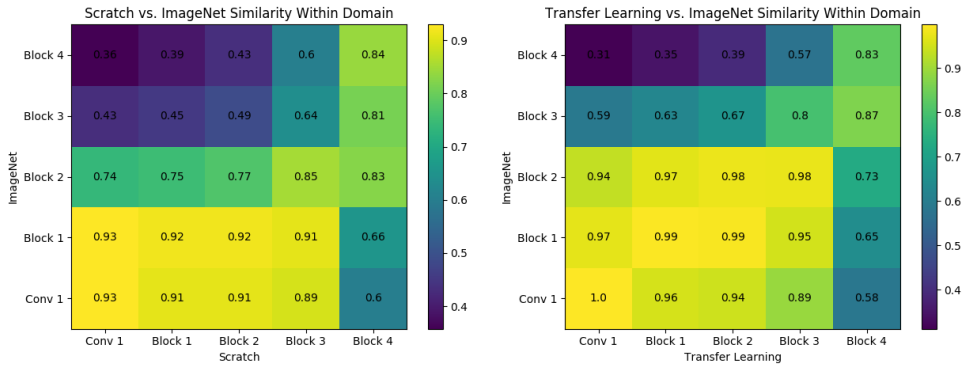


Figure D.1: CKA comparison of representations for task-training vs. ImageNet trained (no task training) for ResNet-50. Left: Linear CKA on within domain horses (used for training) when trained from scratch vs. plain ImageNet trained (no horse pose estimation task training). Right: Same, but for Transfer Learning vs. from ImageNet. Matrices are the averages over the three splits. In short, task training changes representations.

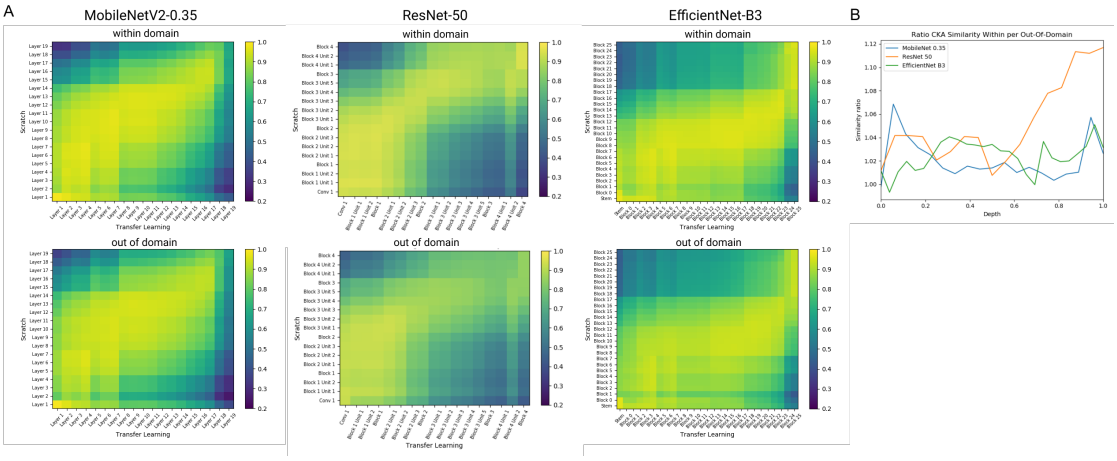


Figure D.2: CKA comparison of representations when trained from scratch vs. from ImageNet initialization. **A:** Top: Linear CKA between layers of individual networks of different depths on within domain horses (used for training the models). Bottom: Same, but for out-of-domain horses (not used in training). Matrices are the averages over the three splits. **B:** Quantification of similarity ratio plotted against depth of the networks.

Results of within domain performance on Animal Pose

In Main Figure 4 we show the performance when we train on only 1 species and testing on another (or all without cow/sheep vs. sheep.cow). Here, as a baseline we report the performance within domain, i.e. for each out-of-domain test species (cow and sheep) we trained on 90% of the cow data and tested on 10% cow data (see tables D.3 and D.4).

Table D.3: Test performance on cow when trained on 90% of cow data

	Normalized Error
MobileNetV2 0.35	0.136
MobileNetV2 1.0	0.093
ResNet 50	0.062
EfficientNet-B0	0.060
EfficientNet-B3	0.054

Table D.4: Test performance on sheep when trained on 90% of sheep data

	Normalized Error
MobileNetV2 0.35	0.385
MobileNetV2 1.0	0.248
ResNet 50	0.186
EfficientNet-B0	0.124
EfficientNet-B3	0.159

Full results on Horse-C

We show the full set of results for the Horse-C benchmark. We compute the corruptions proposed by Hendrycks et al. (Hendrycks & Dietterich, 2019b) using the image corruptions library proposed by Michaelis et al. (Michaelis et al., 2019a).

The original Horse-30 dataset is processed once for each of the corruptions and severities. In total, Horse-C is comprised of 75 evaluation settings with 8,114 images each, yielding a total of 608,550 images. For a visual impression of the impact of different corruptions and severities, see Figures D.3–D.6.

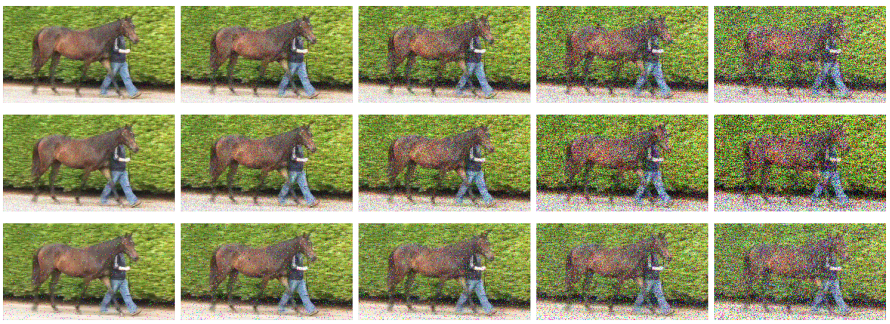


Figure D.3: Noise corruptions for all five different severities (1 to 5, left to right). Top to bottom: Gaussian Noise, Shot Noise, Impulse Noise.

Figure D.4: Blur corruptions for all five different severities (1 to 5, left to right). Top to bottom: Defocus Blur, Motion Blur, Zoom Blur

For evaluation, we consider MobileNetV2-0.35, MobileNetV2-1.0, ResNet-50 and the B0 and B3 variants of EfficientNet. All models are either trained on Horse-10 from scratch or pre-trained on ImageNet and fine-tuned to Horse-10, using the three validation splits used throughout the paper. In contrast to our other experiments, we now fine-tune the BatchNorm layers for these models. For both the w.d. and o.o.d. settings, this yields comparable performance, but enables us to use the batch adaptation technique proposed by Schneider, Rusak et al. (Schneider et al., 2020b) during evaluation on Horse-C, allowing a better estimate of model robustness.

On the clean data, using batch norm adaptation yields slightly improved perfor-

mance for MobileNetV2s on clean within-domain data and deteriorates performance for EfficientNet models. Performance on clean ood. data is improved of all model variants when training from scratch, and improved for MobileNets and ResNets when using pre-trained weights.

We evaluate the normalized errors for the non-adapted model (Base) and after estimating corrected batch normalization statistics (Adapt). The corrected statistics are estimated for each horse identity and corruption as proposed in (Schneider et al., 2020b). We average the normalized metrics across shuffles (and horses as usual). We present the full results for a pre-trained ResNet50 model for all four corruption classes in Tables D.7 and D.8 and contrast this to the within-domain/out-of-domain evaluation setting in Table D.9.

For the ResNet50 model considered in detail, we find that batch normalization helps most for noise and weather corruptions, where we typically found improvements of $60 - 90\%$ and of $30 - 70\%$, respectively. In contrast, blur corruptions and digital corruptions (apart from contrast, defocus blur) saw more modest improvements. It is notable that some of the corruptions—such as elastic transform or pixelation—likely also impact the ground truth posture.

Batch norm adaptation slightly improves the prediction performance when evaluating on different horse identities, but fails to close the gap between the w.d. and ood. setting. In contrast, batch adaptation considerably improves prediction performance on all considered common corruptions.

In summary, we provide an extensive suite of benchmarks for pose estimation and our experiments suggest that domain shift induced by different individuals is difficult in nature (as it is difficult to fix). This further highlights the importance of benchmarks such as Horse-10. Full results for other model variants are depicted in Table D.5 and Table D.6. We report average scores on Horse-C all models in the main text.

Table D.5: Summary results for evaluation of all models on the Horse-C dataset. Results are averaged across all five severities and three validation splits of the data. Adaptive batch normalization (adapt) is crucial for attaining good performance compared to fixing the statistics during evaluation (base). Best viewed in the digital version.

Net Type	mobilenet_v2_0.35				mobilenet_v2_1.0				resnet_50				efficientnet-b0				efficientnet-b3			
Pretrained	False		True		False		True		False		True		False		True		False		True	
Condition	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base
Corruption																				
brightness	0.34	1.76	0.29	0.87	0.27	1.67	0.21	0.94	0.33	1.29	0.17	0.24	0.40	1.40	0.19	0.72	0.33	1.38	0.19	0.70
contrast	0.47	8.02	0.30	4.78	0.36	8.63	0.22	4.93	0.38	8.57	0.18	2.41	0.41	6.95	0.20	4.01	0.37	7.94	0.20	3.43
defocus_blur	0.81	3.22	0.69	2.20	0.61	3.51	0.66	3.35	0.83	3.44	0.60	1.54	0.67	2.05	0.51	2.54	0.71	2.64	0.54	2.01
elastic_transform	0.38	0.96	0.36	0.83	0.32	0.96	0.32	0.92	0.35	0.50	0.26	0.29	0.39	0.91	0.29	0.76	0.38	0.90	0.29	0.77
fog	1.57	6.62	0.41	1.55	1.17	7.20	0.30	2.23	1.09	7.51	0.26	0.56	1.63	5.31	0.27	1.15	1.28	6.80	0.25	1.11
frost	2.27	6.74	1.10	3.78	1.97	6.81	1.00	3.84	1.68	6.44	0.60	1.80	1.39	6.44	0.71	2.91	1.43	7.04	0.67	2.43
gaussian_noise	2.65	5.90	1.68	6.98	2.11	6.19	1.91	7.51	0.97	3.53	0.82	5.25	1.65	5.13	1.22	5.77	1.71	5.82	1.25	5.89
glass_blur	0.60	2.03	0.63	1.57	0.50	2.01	0.67	2.34	0.54	1.69	0.50	0.95	0.53	1.35	0.53	1.56	0.56	1.45	0.59	1.39
impulse_noise	2.36	5.75	1.73	6.88	1.86	6.07	1.91	7.46	0.83	3.47	0.81	5.56	1.45	4.83	0.89	5.46	1.46	5.80	0.86	5.71
jpeg_compression	0.64	1.32	0.52	1.12	0.50	1.49	0.48	1.30	0.39	0.62	0.34	0.39	0.51	1.10	0.43	1.06	0.47	1.13	0.45	1.00
motion_blur	0.83	2.81	0.68	1.84	0.73	2.99	0.68	2.69	0.80	2.29	0.56	1.08	0.68	1.88	0.56	1.72	0.66	2.07	0.56	1.66
none	0.30	0.88	0.26	0.72	0.25	0.87	0.20	0.73	0.27	0.40	0.17	0.19	0.33	0.84	0.18	0.66	0.30	0.83	0.18	0.66
pixelate	0.34	0.99	0.33	0.84	0.28	0.96	0.28	0.99	0.31	0.47	0.23	0.28	0.35	0.89	0.27	0.78	0.33	0.86	0.27	0.76
shot_noise	2.27	5.31	1.29	6.52	1.65	5.57	1.29	6.95	0.72	2.83	0.63	4.40	1.28	4.55	0.82	4.94	1.32	5.63	0.80	4.90
snow	0.89	4.14	0.82	2.55	0.75	4.38	0.76	3.55	0.71	4.89	0.46	1.69	0.70	3.51	0.53	1.75	0.63	4.32	0.51	1.79
zoom_blur	0.98	2.34	0.82	1.74	0.88	2.58	0.89	2.39	0.93	2.16	0.69	1.11	0.93	1.75	0.70	1.60	1.02	1.95	0.71	1.56

Figure D.5: Weather corruptions for all five different severities (1 to 5, left to right). Top to bottom: Snow, Frost, Fog, Brightness

Figure D.6: Digital corruptions for all five different severities (1 to 5, left to right). Top to bottom: Contrast, Elastic Transform, Pixelate, Jpeg Compression

Table D.6: Full result table on Horse-C. All results are averaged across the three validation splits. "none" denotes the uncorrupted Horse-10 dataset. Best viewed in the digital version.

	Net Type	mobilenet_v2_0.35				mobilenet_v2_1.0				resnet_50				efficientnet-b0				efficientnet-b3			
	Pretrained	False		True		False		True		False		True		False		True		False		True	
	Condition	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base	adapt	base
Corruption	Severity																				
brightness	1	0.30	1.02	0.26	0.75	0.25	0.96	0.20	0.78	0.29	0.44	0.16	0.20	0.34	0.92	0.18	0.67	0.30	0.86	0.17	0.68
	2	0.32	1.26	0.26	0.80	0.25	1.20	0.20	0.84	0.31	0.60	0.16	0.21	0.36	1.01	0.18	0.69	0.31	0.95	0.18	0.69
	3	0.33	1.77	0.27	0.87	0.26	1.67	0.20	0.91	0.32	0.99	0.17	0.23	0.39	1.32	0.19	0.71	0.33	1.20	0.18	0.69
	4	0.35	2.20	0.30	0.93	0.27	2.14	0.22	1.02	0.34	1.71	0.18	0.26	0.42	1.67	0.20	0.74	0.34	1.61	0.20	0.70
	5	0.40	2.55	0.35	1.00	0.30	2.38	0.24	1.17	0.39	2.74	0.19	0.32	0.46	2.07	0.22	0.78	0.38	2.25	0.21	0.72
contrast	1	0.31	5.23	0.26	0.96	0.25	5.25	0.20	1.02	0.27	5.64	0.17	0.27	0.33	2.50	0.18	0.80	0.30	3.87	0.18	0.73
	2	0.32	6.91	0.27	1.38	0.25	7.93	0.20	1.77	0.28	7.86	0.17	0.39	0.34	5.22	0.18	1.03	0.31	7.36	0.18	0.87
	3	0.35	8.63	0.27	3.50	0.27	9.51	0.20	4.67	0.30	9.03	0.17	1.05	0.35	7.88	0.19	2.20	0.32	9.21	0.18	1.70
	4	0.48	9.54	0.29	8.13	0.36	10.22	0.22	8.17	0.38	10.05	0.18	3.90	0.40	9.48	0.20	6.85	0.36	9.60	0.20	5.66
	5	0.88	9.77	0.38	9.92	0.68	10.25	0.29	9.03	0.69	10.28	0.22	6.43	0.63	9.68	0.26	9.17	0.56	9.66	0.25	8.16
defocus_blur	1	0.36	1.11	0.32	0.89	0.30	1.09	0.26	1.00	0.33	0.64	0.24	0.31	0.38	0.97	0.23	0.78	0.34	0.91	0.23	0.77
	2	0.41	1.48	0.39	1.14	0.35	1.39	0.31	1.50	0.39	1.11	0.29	0.40	0.42	1.11	0.28	1.02	0.39	1.05	0.28	0.91
	3	0.65	3.21	0.59	1.92	0.50	3.08	0.52	3.23	0.63	3.23	0.47	0.94	0.58	1.78	0.44	2.31	0.58	2.06	0.44	1.56
	4	1.03	4.59	0.87	2.96	0.76	5.30	0.86	4.97	1.09	5.36	0.82	2.26	0.84	2.69	0.66	3.69	0.92	3.65	0.69	2.75
	5	1.60	5.69	1.28	4.11	1.12	6.69	1.36	6.06	1.72	6.83	1.19	3.77	1.13	3.68	0.96	4.88	1.30	5.52	1.05	4.05
elastic_transform	1	0.32	0.92	0.29	0.75	0.27	0.90	0.23	0.79	0.30	0.43	0.20	0.22	0.35	0.87	0.22	0.69	0.33	0.86	0.21	0.69
	2	0.34	0.94	0.31	0.77	0.29	0.92	0.26	0.83	0.32	0.46	0.22	0.24	0.36	0.88	0.24	0.72	0.35	0.88	0.24	0.72
	3	0.38	0.96	0.35	0.82	0.31	0.95	0.31	0.91	0.35	0.50	0.25	0.28	0.39	0.91	0.29	0.75	0.37	0.90	0.28	0.76
	4	0.40	0.99	0.39	0.86	0.34	0.99	0.36	0.99	0.37	0.53	0.29	0.32	0.41	0.93	0.33	0.79	0.40	0.92	0.33	0.80
	5	0.44	1.01	0.44	0.93	0.38	1.04	0.43	1.11	0.41	0.58	0.33	0.38	0.44	0.96	0.38	0.85	0.44	0.96	0.39	0.87
fog	1	0.87	5.11	0.34	0.96	0.65	5.27	0.24	1.11	0.60	5.78	0.20	0.28	0.89	3.46	0.21	0.81	0.71	4.76	0.21	0.76
	2	1.26	6.42	0.36	1.21	0.94	6.93	0.26	1.57	0.82	7.33	0.22	0.33	1.30	4.89	0.23	0.96	1.03	6.53	0.22	0.86
	3	1.74	7.10	0.41	1.59	1.29	7.82	0.29	2.38	1.18	8.03	0.26	0.48	1.80	5.91	0.26	1.20	1.40	7.45	0.24	1.06
	4	1.81	6.97	0.43	1.64	1.34	7.67	0.32	2.47	1.24	7.97	0.28	0.56	1.89	5.82	0.28	1.21	1.47	7.35	0.26	1.15
	5	2.17	7.49	0.51	2.33	1.65	8.29	0.40	3.60	1.60	8.46	0.36	1.13	2.27	6.45	0.35	1.58	1.77	7.93	0.32	1.74
frost	1	1.02	4.11	0.46	1.32	0.73	3.81	0.37	1.30	0.65	3.24	0.26	0.37	0.58	2.94	0.34	0.88	0.56	3.52	0.33	0.82
	2	1.98	6.45	0.86	2.99	1.63	6.32	0.75	2.97	1.30	6.39	0.45	1.15	1.08	6.01	0.57	1.86	1.14	7.02	0.55	1.57
	3	2.59	7.56	1.23	4.41	2.30	7.71	1.14	4.50	1.92	7.39	0.65	2.09	1.58	7.56	0.77	3.34	1.63	8.05	0.73	2.73
	4	2.75	7.66	1.31	4.70	2.41	7.92	1.21	4.81	2.06	7.48	0.71	2.30	1.67	7.60	0.84	3.69	1.77	8.16	0.79	3.02
	5	3.01	7.92	1.63	5.50	2.77	8.32	1.54	5.61	2.46	7.71	0.90	3.07	2.03	8.12	1.01	4.77	2.05	8.47	0.95	3.99
gaussian_noise	1	0.93	3.08	0.51	3.73	0.59	2.99	0.50	4.81	0.34	0.61	0.27	0.68	0.58	1.55	0.41	1.36	0.53	1.73	0.40	1.45
	2	1.61	5.26	0.80	6.21	1.00	5.42	0.82	6.78	0.42	1.12	0.37	2.67	0.85	3.03	0.57	3.09	0.80	3.91	0.56	3.50
	3	2.63	6.60	1.41	7.96	1.87	6.91	1.56	8.31	0.63	3.13	0.57	6.12	1.45	5.25	0.93	6.59	1.43	6.37	0.91	6.66
	4	3.62	7.15	2.29	8.48	2.99	7.55	2.65	8.75	1.16	5.62	1.02	8.02	2.23	7.35	1.55	8.70	2.39	7.96	1.60	8.54
	5	4.47	7.43	3.40	8.53	4.10	8.06	4.00	8.92	2.28	7.17	1.88	8.77	3.12	8.46	2.62	9.13	3.40	9.14	2.77	9.29
glass_blur	1	0.33	0.98	0.30	0.82	0.27	0.94	0.25	0.85	0.31	0.50	0.22	0.27	0.35	0.89	0.21	0.71	0.33	0.85	0.21	0.71
	2	0.38	1.11	0.37	0.95	0.32	1.11	0.33	1.07	0.35	0.68	0.28	0.35	0.39	0.98	0.28	0.82	0.36	0.93	0.30	0.81
	3	0.53	1.54	0.58	1.38	0.47	1.45	0.62	1.83	0.51	1.19	0.48	0.67	0.52	1.14	0.49	1.15	0.52	1.13	0.55	1.13
	4	0.65	2.38	0.72	1.79	0.57	2.06	0.83	2.93	0.59	1.84	0.60	1.06	0.58	1.42	0.66	1.61	0.63	1.48	0.74	1.51
	5	1.09	4.16	1.18	2.91	0.88	4.51	1.34	5.04	0.95	4.26	0.93	2.41	0.83	2.33	1.01	3.50	0.98	2.89	1.17	2.80
impulse_noise	1	0.92	2.86	0.56	3.69	0.58	2.80	0.56	4.72	0.38	0.76	0.36	1.43	0.63	1.66	0.33	1.12	0.56	2.08	0.31	1.05
	2	1.53	5.19	0.85	6.26	1.00	5.37	0.94	6.95	0.47	1.50	0.47	3.89	0.89	3.17	0.47	2.92	0.82	4.55	0.43	3.50
	3	2.10	6.26	1.21	7.45	1.44	6.53	1.42	7.95	0.57	2.69	0.58	5.74	1.18	4.45	0.62	5.56	1.12	5.87	0.57	5.83
	4	3.18	7.06	2.36	8.42	2.61	7.55	2.67	8.75	0.99	5.38	0.97	7.98	1.90	6.82	1.12	8.60	1.93	7.60	1.06	8.76
	5	4.07	7.38	3.64	8.58	3.68	8.07	3.95	8.93	1.76	6.99	1.68	8.78	2.67	8.07	1.90	9.10	2.84	8.90	1.94	9.38
jpeg_compression	1	0.45	1.05	0.35	0.88	0.37	1.09	0.33	0.96	0.31	0.45	0.24	0.28	0.40	0.92	0.29	0.77	0.37	0.93	0.29	0.77
	2	0.53	1.12	0.41	0.95	0.42	1.18	0.37	1.05	0.35	0.48	0.27	0.31	0.45	0.95	0.34	0.83	0.41	0.98	0.35	0.83
	3	0.66	1.16	0.51	1.03	0.49	1.25	0.43	1.14	0.37	0.50	0.30	0.33	0.48	1.01	0.39	0.91	0.44	1.07	0.39	0.91
	4	0.75	1.45	0.61	1.25	0.58	1.66	0.56	1.49	0.42	0.70	0.38	0.44	0.57	1.23	0.49	1.17	0.53	1.27	0.53	1.11
	5	0.80	1.81	0.71	1.51	0.62	2.26	0.70	1.86	0.51	0.95	0.52	0.57	0.63	1.37	0.65	1.62	0.58	1.39	0.68	1.38
motion_blur	1	0.39	1.15	0.34	0.86	0.34	1.11	0.30	0.98	0.37	0.60	0.26	0.35	0.41	1.02	0.28	0.80	0.38	0.97	0.27	0.81
	2	0.51	1.54	0.44	1.02	0.45	1.49	0.41	1.31	0.48	0.95	0.36	0.48	0.49	1.20	0.38	0.98	0.46	1.16	0.38	0.96
	3	0.73	2.69	0.61	1.47	0.63	2.59	0.60	2.27	0.70	1.90	0.51	0.77	0.64	1.67	0.53	1.41	0.61	1.69	0.53	1.36
	4	1.10	3.94	0.88	2.46	0.96	4.39	0.91	3.90	1.05	3.47	0.74	1.46	0.85	2.46	0.73	2.29	0.84	2.77	0.73	2.17
	5	1.44	4.73	1.11	3.41	1.26	5.38	1.17	4.97	1.38	4.52	0.95	2.32	1.03	3.06	0.88	3.14	1.03	3.75	0.88	3.01
pixelate	1	0.30	0.92	0.27	0.74	0.25	0.89	0.21	0.78	0.28	0.41	0.18	0.21	0.33	0.85	0.20	0.69	0.31	0.84	0.19	0.69
	2	0.31	0.93	0.28	0.75	0.25	0.89	0.23	0.80	0.28	0.42	0.18	0.22	0.33	0.86	0.23	0.70	0.31	0.84	0.21	0.69
	3	0.33	0.98	0.31	0.82	0.26	0.93	0.26	0.91	0.29	0.46	0.22	0.27	0.35	0.88	0.24	0.73	0.32	0.86	0.25	0.73
	4	0.37	1.04	0.36	0.91	0.30	1.02	0.34	1.13	0.32	0.50	0.26	0.34	0.37	0.91	0.31	0.83	0.35	0.88	0.31	0.80
	5	0.38	1.09	0.41	0.98	0.32	1.08	0.38	1.33	0.35	0.56	0.29	0.38	0.39	0.94	0.38	0.96	0.36	0.90	0.37	0.89
shot_noise	1	0.69	1.98	0.44	2.43	0.43	1.86	0.40	3.21	0.31	0.52	0.23	0.41	0.46	1.14	0.32	0.97	0.43	1.24	0.30	0.98
	2	1.19	4.19	0.65	5.61	0.68	4.14	0.62	5.99	0.37	0.88	0.31	1.52	0.66	2.10	0.44	1.90	0.60	2.97	0.41	2.15
	3	1.97	5.93	1.02	7.70	1.19	6.31	1.01	7.93	0.50	1.89	0.46	4.28	1.00	4.06	0.63	4.49	0.93	5.95	0.59	4.58
	4	3.38	7.07	1.87	8.41	2.49	7.55	1.84	8.77	0.92	4.68	0.86	7.45	1.82	7.17	1.10	8.33	1.89	8.46	1.09	7.82
	5	4.12	7.41	2.50	8.44	3.47	7.98	2.56	8.84	1.51	6.16	1.29	8.34	2.46	8.30	1.60	9.01	2.74	9.55	1.63	8.98
snow	1	0.49	1.51	0.47	1.09	0.41	1.67	0.41	1.30	0.41	1.21	0.30	0.44	0.45	1.20	0.36	0.87	0.42	1.30	0.33	0.81
	2	0.74	3.60	0.71	2.38	0.63	4.19	0.70	3.51	0.61	5.65	0.43	1.38	0.60	2.60	0.51	1.42	0.55	3.73	0.48	1.34
	3	0.85	3.89	0.77	2.16	0.72	4.32	0.70	3.12	0.66	4.64	0.45	1.27	0.69	2.78	0.51	1.38	0.61	3.71	0.50	1.44
	4	1.20	5.37	1.06	3.23	1.02	5.56	0.98	4.61	0.94	6.10	0.59	2.53	0.91	4.58	0.63	2.13	0.81	5.61	0.62	2.42
	5	1.15	6.35	1.07	3.92	0.97	6.18	1.03	5.23	0.91	6.85	0.54	2.82	0.83	6.39	0.63	2.93	0.75	7.23	0.63	2.93
zoom_blur	1	0.67	1.69	0.58	1.16	0.59	1.65	0.58	1.50	0.63	1.20	0.45	0.56	0.64	1.28	0.45	1.04	0.67	1.29	0.46	1.05
	2	0.85	2.05	0.78	1.51	0.76	2.13	0.85	2.05	0.80	1.70	0.64	0.87	0.81	1.51	0.63	1.37	0.88	1.58	0.65	1.35
	3	0.95	2.39	0.78	1.64	0.84	2.60	0.83	2.32	0.92	2.13	0.66	1.04	0.90	1.72	0.67	1.50	0.99	1.91	0.67	1.45
	4	1.11	2.65	0.93	1.98	1.00	3.02	1.05	2.87	1.08	2.62	0.85	1.40	1.06	1.94	0.84	1.86	1.19	2.27	0.85	1.83
	5	1.34	2.95	1.01	2.41	1.21	3.48	1.15	3.23	1.24	3.12	0.87	1.68	1.21	2.27	0.93	2.24	1.36	2.70	0.92	2.14
none (Horse-10)	0	0.30	0.88	0.26	0.72	0.25	0.87	0.20	0.73	0.27	0.40	0.17	0.19	0.33	0.84	0.18	0.66	0.30	0.83	0.18	0.66

Table D.7: Improvements using batch norm adaptation on the Horse-C Noise and Blur corruption subsets for a pre-trained ResNet50 model.

Noise Corruption	Severity	Base	Adapt	Δ_{abs}	Δ_{rel}
Gaussian Noise	1	0.427	0.138	0.289	67.7%
	2	2.187	0.201	1.986	90.8%
	3	5.556	0.314	5.242	94.3%
	4	7.843	0.649	7.194	91.7%
	5	8.894	1.410	7.484	84.1%
Impulse Noise	1	1.079	0.201	0.878	81.4%
	2	3.432	0.276	3.156	92.0%
	3	5.393	0.360	5.033	93.3%
	4	7.839	0.663	7.176	91.5%
	5	8.923	1.339	7.584	85.0%
Shot Noise	1	0.191	0.114	0.077	40.3%
	2	0.986	0.152	0.834	84.6%
	3	3.618	0.244	3.374	93.3%
	4	7.225	0.516	6.709	92.9%
	5	8.365	0.894	7.471	89.3%

Blur Corruption	Severity	Base	Adapt	Δ_{abs}	Δ_{rel}
Defocus Blur	1	0.137	0.100	0.037	27.0%
	2	0.169	0.127	0.042	24.9%
	3	0.369	0.233	0.136	36.9%
	4	1.569	0.446	1.123	71.6%
	5	3.480	0.763	2.717	78.1%
Motion Blur	1	0.213	0.160	0.053	24.9%
	2	0.290	0.224	0.066	22.8%
	3	0.340	0.335	0.005	1.5%
	4	0.864	0.501	0.363	42.0%
	5	1.596	0.645	0.951	59.6%
Zoom Blur	1	0.331	0.288	0.043	13.0%
	2	0.536	0.436	0.100	18.7%
	3	0.654	0.467	0.187	28.6%
	4	0.974	0.620	0.354	36.3%
	5	1.217	0.640	0.577	47.4%

Table D.8: Improvements using batch norm adaptation on the Horse-C Weather and Digital corruptions subsets for a pre-trained ResNet50 model.

Weather Corruption	Severity	Base	Adapt	Δ_{abs}	Δ_{rel}
Brightness	1	0.120	0.084	0.036	30.0%
	2	0.127	0.083	0.044	34.6%
	3	0.141	0.084	0.057	40.4%
	4	0.165	0.089	0.076	46.1%
	5	0.205	0.097	0.108	52.7%
Fog	1	0.156	0.097	0.059	37.8%
	2	0.191	0.107	0.084	44.0%
	3	0.289	0.126	0.163	56.4%
	4	0.330	0.137	0.193	58.5%
	5	0.764	0.176	0.588	77.0%
Frost	1	0.193	0.125	0.068	35.2%
	2	0.672	0.249	0.423	62.9%
	3	1.447	0.393	1.054	72.8%
	4	1.680	0.449	1.231	73.3%
	5	2.375	0.573	1.802	75.9%
Snow	1	0.229	0.155	0.074	32.3%
	2	0.737	0.252	0.485	65.8%
	3	0.720	0.270	0.450	62.5%
	4	1.873	0.386	1.487	79.4%
	5	2.146	0.348	1.798	83.8%

Digital Corruption	Severity	Base	Adapt	Δ_{abs}	Δ_{rel}
Contrast	1	0.151	0.085	0.066	43.7%
	2	0.211	0.084	0.127	60.2%
	3	0.840	0.083	0.757	90.1%
	4	3.700	0.085	3.615	97.7%
	5	6.406	0.103	6.303	98.4%
Elastic Transform	1	0.121	0.092	0.029	24.0%
	2	0.127	0.101	0.026	20.5%
	3	0.139	0.116	0.023	16.5%
	4	0.154	0.133	0.021	13.6%
	5	0.175	0.157	0.018	10.3%
Jpeg Compression	1	0.136	0.108	0.028	20.6%
	2	0.152	0.128	0.024	15.8%
	3	0.170	0.138	0.032	18.8%
	4	0.216	0.189	0.027	12.5%
	5	0.305	0.276	0.029	9.5%
Pixelate	1	0.117	0.087	0.030	25.6%
	2	0.117	0.089	0.028	23.9%
	3	0.125	0.100	0.025	20.0%
	4	0.142	0.112	0.030	21.1%
	5	0.156	0.132	0.024	15.4%

Table D.9: Small improvements by using batch adaptation on the identity shift task for a pre-trained ResNet50 model. Note that the o.o.d. performance is still substantially worse (higher normalized error) than the within-domain performance.

	Base	Adapt	Δ_{abs}	Δ_{rel}
Identity (wd)	0.115	0.086	0.029	25.2%
Identity (ood)	0.271	0.247	0.024	8.9%

Inference Speed Benchmarking

We introduced new DeepLabCut variants that can achieve high accuracy but with higher speed than the original ResNet backbone (Mathis et al., 2018). Here we provide a simple benchmark to document how fast the EfficientNet and MobileNetv2 backbones are (Figure D.7). We evaluated the inference speed for one video with 11,178 frames at resolutions 512×512, 256×256 and 128×128. We used batch sizes: $[1, 2, 4, 16, 32, 128, 256, 512]$, and ran all models for all 3 (training set shuffles) trained with 50% of the data in a pseudo random order on a NVIDIA Titan RTX. We also updated the inference code from its numpy implementation (Mathis & Warren, 2018) to TensorFlow, which brings a $2-10\%$ gain in speed.

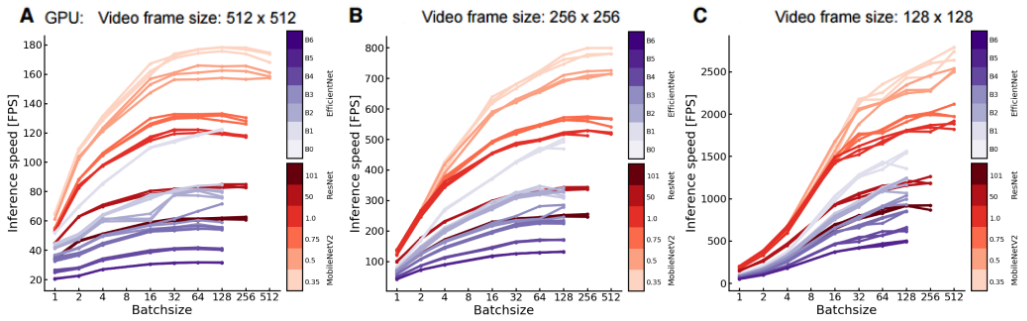


Figure D.7: Speed Benchmarking for MobileNetV2s, ResNets and EfficientNets: Inference speed for videos of different dimensions for all the architectures. **A-C:** FPS vs. batchsize, with video frame sizes as stated in the title. Three splits are shown for each network. MobileNetV2 gives a more than 2X speed improvement (over ResNet-50) for offline processing and about 40% for batchsize=1 on a Titan RTX GPU.

E

Contrastive Learning Inverts the Data Generating Process

Extended Theory for Hyperspheres

Assumptions

Generative Process Let the generator $g : \mathbb{R}^N \to \mathcal{X}$ with $\mathcal{X} \subseteq \mathbb{R}^K$ and $K \geq N$. Further, let the restriction of g to the space $\mathcal{Z} = \mathbb{S}^{N-1} \subset \mathbb{R}^N$ be injective and g be differentiable in the vicinity of $\mathcal{Z}$. We assume that the marginal distribution $p(\mathbf{z})$ over latent variables $\mathbf{z} \in \mathcal{Z}$ is uniform:

$$p(\mathbf{z}) = \frac{1}{|\mathcal{Z}|}. \tag{E.1}$$

Further, we assume that the conditional distribution over positive pairs $p(\tilde{\mathbf{z}}|\mathbf{z})$ is a von Mises-Fisher (vMF) distribution

$$p(\tilde{\mathbf{z}}|\mathbf{z}) = C_p^{-1} e^{\kappa \mathbf{z}^\top \tilde{\mathbf{z}}} \tag{E.2}$$

$$\text{with} \quad C_p := \int e^{\kappa \boldsymbol{\eta}^\top \tilde{\mathbf{z}}} \, \mathrm{d}\tilde{\mathbf{z}}, \tag{E.3}$$

where κ is a parameter controlling the width of the distribution and $\boldsymbol{\eta}$ is any vector on the hypersphere. Finally, we assume that during training one has access to observations $\mathbf{x}$, which are samples from these distributions transformed by the generator function g.

Model Let $f : \mathcal{X} \to \mathbb{S}_r^{N-1}$, where $\mathbb{S}_r^{N-1}$ denotes a hypersphere with radius r. The parameters of this model are optimized using contrastive learning. We associate a

conditional distribution $q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ with our model f through $h = f \circ g$ and

$$\begin{aligned} q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_q^{-1}(\mathbf{z}) e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \\ \text{with} \quad C_q(\mathbf{z}) &:= \int e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \, \mathrm{d}\tilde{\mathbf{z}}, \end{aligned} \tag{E.4}$$

where $C_q(\mathbf{z})$ is the partition function and $\tau > 0$ is a scale parameter.

Proofs for Theory section

We begin by recalling a result of Wang and Isola (2020), where the authors show an asymptotic relation between the contrastive loss $\mathcal{L}_{\mathsf{contr}}$ and two loss functions, the *alignment* loss $\mathcal{L}_{\mathsf{align}}$ and the *uniformity* loss $\mathcal{L}_{\mathsf{uni}}$:

Proposition A (Asymptotics of $\mathcal{L}_{\mathsf{contr}}$, Wang and Isola, 2020)**.** *For fixed $\tau > 0$, as the number of negative samples $M \to \infty$, the (normalized) contrastive loss converges to*

$$\lim_{M \to \infty} \mathcal{L}_{\mathsf{contr}}(f; \tau, M) - \log M = \mathcal{L}_{\mathsf{align}}(f; \tau) + \mathcal{L}_{\mathsf{uni}}(f; \tau), \tag{E.5}$$

where

$$\begin{aligned} \mathcal{L}_{\mathsf{align}}(f; \tau) &:= -\frac{1}{\tau} \mathop{\mathbb{E}}_{(\tilde{\mathbf{z}}, \mathbf{z}) \sim p(\tilde{\mathbf{z}}, \mathbf{z})} \left[(f \circ g)(\mathbf{z})^\mathsf{T} (f \circ g)(\mathbf{z}) \right] \\ \mathcal{L}_{\mathsf{uni}}(f; \tau) &:= \mathop{\mathbb{E}}_{\mathbf{z} \sim p(\mathbf{z})} \left[\log \mathop{\mathbb{E}}_{\tilde{\mathbf{z}} \sim p(\tilde{\mathbf{z}})} \left[e^{(f \circ g)(\tilde{\mathbf{z}})^\mathsf{T} (f \circ g)(\mathbf{z})/\tau} \right] \right]. \end{aligned} \tag{E.6}$$

Proof. See Theorem 1 of Wang and Isola (2020). Note that they originally formulated the losses in terms of observations $\mathbf{x}$ and not in terms of the latent variables $\mathbf{z}$. However, this modified version simplifies notation in the following. □

Based on this result, we show that the contrastive loss $\mathcal{L}_{\mathsf{contr}}$ asymptotically converges to the cross-entropy between the ground-truth conditional p and our assumed model conditional distribution q_h, up to a constant. This is notable, because given the correct model specification for q_h, it is well-known that the cross-entropy is minimized iff $q_h = p$, i.e., the ground-truth conditional distribution and the model distribution will match.

Theorem 3 ($\mathcal{L}_{\mathsf{contr}}$ converges to the cross-entropy between latent distributions)**.** *If the ground-truth marginal distribution p is uniform, then for fixed $\tau > 0$, as the number of negative samples $M \to \infty$, the (normalized) contrastive loss converges to*

$$\begin{aligned} \lim_{M \to \infty} \mathcal{L}_{\mathsf{contr}}(f; \tau, M) - \log M + \log |\mathcal{Z}| = \\ \mathop{\mathbb{E}}_{\mathbf{z} \sim p(\mathbf{z})} \left[H(p(\cdot|\mathbf{z}), q_h(\cdot|\mathbf{z})) \right] \end{aligned} \tag{E.7}$$

where H is the cross-entropy between the ground-truth conditional distribution p over positive

pairs and a conditional distribution q_{h} parameterized by the model f, and $C_h(\mathbf{z}) \in \mathbb{R}^+$ is the partition function of q_{h} (see Appendix E):

$$\begin{aligned} q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_h(\mathbf{z})^{-1} e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \\ \textit{with} \quad C_h(\mathbf{z}) &:= \int e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \, \mathrm{d}\tilde{\mathbf{z}}. \end{aligned} \tag{E.8}$$

Proof. The cross-entropy between the conditional distributions p and q_h is given by

$$\mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[H(p(\cdot|\mathbf{z}), q_h(\cdot|\mathbf{z})) \right] \tag{E.9}$$

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})} \left[-\log q_h(\tilde{\mathbf{z}}|\mathbf{z}) \right] \right] \tag{E.10}$$

$$= \mathop{\mathbb{E}}_{\tilde{\mathbf{z}},\mathbf{z}\sim p(\tilde{\mathbf{z}},\mathbf{z})} \left[-\frac{1}{\tau} h(\tilde{\mathbf{z}})^\top h(\mathbf{z}) + \log C_h(\mathbf{z}) \right] \tag{E.11}$$

$$= -\frac{1}{\tau} \mathop{\mathbb{E}}_{\tilde{\mathbf{z}},\mathbf{z}\sim p(\tilde{\mathbf{z}},\mathbf{z})} \left[h(\tilde{\mathbf{z}})^\top h(\mathbf{z}) \right] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log C_h(\mathbf{z}) \right]. \tag{E.12}$$

Using the definition of C_h in Eq. (E.8) we obtain

$$= -\frac{1}{\tau} \mathop{\mathbb{E}}_{\tilde{\mathbf{z}},\mathbf{z}\sim p(\tilde{\mathbf{z}},\mathbf{z})} \left[h(\tilde{\mathbf{z}})^\top h(\mathbf{z}) \right] \tag{E.13}$$

$$+ \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log \int_{\mathcal{Z}} e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \, \mathrm{d}\tilde{\mathbf{z}} \right]. \tag{E.14}$$

By assumption the marginal distribution is uniform, i.e., $p(\mathbf{z}) = |\mathcal{Z}|^{-1}$. We expand by $|\mathcal{Z}||\mathcal{Z}|^{-1}$ and estimate the integral by sampling from $p(\mathbf{z}) = |\mathcal{Z}|^{-1}$, yielding

$$= -\frac{1}{\tau} \mathop{\mathbb{E}}_{\tilde{\mathbf{z}},\mathbf{z}\sim p(\tilde{\mathbf{z}},\mathbf{z})} \left[h(\tilde{\mathbf{z}})^\top h(\mathbf{z}) \right] \tag{E.15}$$

$$+ \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log |\mathcal{Z}| \mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})} \left[e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \right] \right] \tag{E.16}$$

$$= -\frac{1}{\tau} \mathop{\mathbb{E}}_{\tilde{\mathbf{z}},\mathbf{z}\sim p(\tilde{\mathbf{z}},\mathbf{z})} \left[h(\tilde{\mathbf{z}})^\top h(\mathbf{z}) \right] \tag{E.17}$$

$$+ \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log \mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})} \left[e^{h(\tilde{\mathbf{z}})^\top h(\mathbf{z})/\tau} \right] \right] + \log |\mathcal{Z}|. \tag{E.18}$$

By inserting the definition $h = f \circ g$,

$$= -\frac{1}{\tau} \mathop{\mathbb{E}}_{\tilde{\mathbf{z}},\mathbf{z}\sim p(\tilde{\mathbf{z}},\mathbf{z})} \left[(f \circ g)(\tilde{\mathbf{z}})^\top (f \circ g)(\mathbf{z}) \right] \tag{E.19}$$

$$+ \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log \mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})} \left[e^{(f\circ g)(\tilde{\mathbf{z}})^\top (f\circ g)(\mathbf{z})/\tau} \right] \right] \tag{E.20}$$

$$+ \log |\mathcal{Z}|, \tag{E.21}$$

we can identify the losses introduced in Proposition A,

$$= \mathcal{L}_{\text{align}}(f;\tau) + \mathcal{L}_{\text{uni}}(f;\tau) + \log|\mathcal{Z}|, \tag{E.22}$$

which recovers the original alignment term and the uniformity term for maximimizing entropy by means of a von Mises-Fisher KDE up to the constant $\log|\mathcal{Z}|$. According to Proposition A this equals

$$= \lim_{M\to\infty} \mathcal{L}_{\text{contr}}(f;\tau,M) - \log M + \log|Z|, \tag{E.23}$$

which concludes the proof. □

Proposition 4 (Minimizers of the cross-entropy maintain the dot product)**.** *Let* $\mathcal{Z} = \mathbb{S}^{N-1}$, $\tau > 0$ *and consider the ground-truth conditional distribution of the form* $p(\tilde{\mathbf{z}}|\mathbf{z}) = C_p^{-1}\exp(\kappa\tilde{\mathbf{z}}^\top\mathbf{z})$*. Let* h *map onto a hypersphere with radius* $\sqrt{\tau\kappa}$*.*[1] *Consider the conditional distribution* q_h *parameterized by the model, as defined above in Theorem 3, where the hypothesis class for* h *is assumed to be sufficiently flexible such that* $p(\tilde{\mathbf{z}}|\mathbf{z})$ *and* $q_{\text{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ *can match. If* h *is a minimizer of the cross-entropy* $\mathbb{E}_{p(\tilde{\mathbf{z}}|\mathbf{z})}[-\log q_{\text{h}}(\tilde{\mathbf{z}}|\mathbf{z})]$*, then* $p(\tilde{\mathbf{z}}|\mathbf{z}) = q_{\text{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ *and* $\forall\mathbf{z},\tilde{\mathbf{z}} : \kappa\mathbf{z}^\top\tilde{\mathbf{z}} = h(\mathbf{z})^\top h(\tilde{\mathbf{z}})$.

Proof. By assumption, $q_{\text{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ is powerful enough to match $p(\tilde{\mathbf{z}}|\mathbf{z})$ for the correct choice of h — in particular, for $h(\mathbf{z}) = \sqrt{\tau\kappa}\mathbf{z}$. The global minimum of the cross-entropy between two distributions is reached if they match by value and have the same support. Thus, this means

$$p(\tilde{\mathbf{z}}|\mathbf{z}) = q_{\text{h}}(\tilde{\mathbf{z}}|\mathbf{z}). \tag{E.24}$$

This expression also holds true for $\tilde{\mathbf{z}} = \mathbf{z}$; additionally using that h maps from a unit hypersphere to one with radius $\sqrt{\tau\kappa}$ yields

$$p(\mathbf{z}|\mathbf{z}) = q_{\text{h}}(\mathbf{z}|\mathbf{z}) \tag{E.25}$$

$$\Leftrightarrow \quad C_p^{-1}e^{\kappa\mathbf{z}^\top\mathbf{z}} = C_h(\mathbf{z})^{-1}e^{h(\mathbf{z})^\top h(\mathbf{z})/\tau} \tag{E.26}$$

$$\Leftrightarrow \quad C_p^{-1}e^{\kappa} = C_h(\mathbf{z})^{-1}e^{\kappa} \tag{E.27}$$

$$\Leftrightarrow \quad C_p = C_h. \tag{E.28}$$

As the normalization constants are identical we get for all $\mathbf{z},\tilde{\mathbf{z}} \in \mathcal{Z}$

$$e^{\kappa\mathbf{z}^\top\tilde{\mathbf{z}}} = e^{h(\mathbf{z})^\top h(\tilde{\mathbf{z}})} \Leftrightarrow \kappa\mathbf{z}^\top\tilde{\mathbf{z}} = h(\mathbf{z})^\top h(\tilde{\mathbf{z}}). \tag{E.29}$$

□

Proposition 5 (Extension of the Mazur-Ulam theorem to hyperspheres and the dot product)**.** *Let* $\mathcal{Z} = \mathbb{S}^{N-1}$ *and* $\mathcal{Z}' = \mathbb{S}_r^{N-1}$ *be the hyperspheres with radius* 1 *and* $r > 0$*, respectively. If* $h : \mathbb{R}^N \to \mathcal{Z}'$ *is differentiable in the vicinity of* $\mathcal{Z}$ *and its restriction to* $\mathcal{Z}$

[1] Note that in practice this can be implemented as a learnable rescaling operation of the network f.

maintains the dot product up to a constant factor, i.e., $\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z} : r^2 \mathbf{z}^\top \tilde{\mathbf{z}} = h(\mathbf{z})^\top h(\tilde{\mathbf{z}})$*, then* h *is an orthogonal linear transformation scaled by* r *for all* $\mathbf{z} \in \mathcal{Z}$*.*

Proof. First, we begin with the case $r = 1$. As h maintains the dot product we have:

$$\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z} : \mathbf{z}^\top \tilde{\mathbf{z}} = h(\mathbf{z})^\top h(\tilde{\mathbf{z}}). \tag{E.30}$$

We consider the partial derivative w.r.t. $\mathbf{z}$ and obtain:

$$\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z} : \tilde{\mathbf{z}} = \mathbf{J}_h^\top(\mathbf{z}) h(\tilde{\mathbf{z}}). \tag{E.31}$$

Taking the partial derivative w.r.t. $\tilde{\mathbf{z}}$ yields

$$\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z} : \mathbf{I} = \mathbf{J}_h^\top(\mathbf{z}) \mathbf{J}_h(\tilde{\mathbf{z}}). \tag{E.32}$$

We can now conclude

$$\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z} : \mathbf{J}_h(\tilde{\mathbf{z}})^{-1} = \mathbf{J}_h^\top(\mathbf{z}). \tag{E.33}$$

which implies a constant Jacobian matrix $\mathbf{J}_h(\mathbf{z}) = \mathbf{J}_h$ as the identity holds on all points in $\mathcal{Z}$, and further that the Jacobian $\mathbf{J}_h$ is orthogonal. Hence, $\forall \mathbf{z} \in \mathcal{Z} : h(\mathbf{z}) = \mathbf{J}_h \mathbf{z}$ is an orthogonal linear transformation.

Finally, for $r \neq 1$ we can leverage the previous result by introducing $h'(\mathbf{z}) := h(\mathbf{z})/r$. For h' the previous argument holds, implying that h' is an orthogonal transformation. Therefore, the restriction of h to $\mathcal{Z}$ is an orthogonal linear transformation scaled by r^2. □

Taking all of this together, we can now prove Theorem 4:

Theorem 4. *Let* $\mathcal{Z} = \mathbb{S}^{N-1}$*, the ground-truth marginal be uniform, and the conditional a vMF distribution (cf. Eq. 6.2). Let the restriction of the mixing function* g *to* $\mathcal{Z}$ *be injective and* h *be differentiable in a vicinity of* $\mathcal{Z}$*. If the assumed form of* q_{h}*, as defined above, matches that of* p*, and if* f *is differentiable and minimizes the CL loss as defined in Eq. (6.1), then for fixed* $\tau > 0$ *and* $M \to \infty$*,* $h = f \circ g$ *is linear, i.e.,* f *recovers the latent sources up to an orthogonal linear transformation and a constant scaling factor.*

Proof. As f minimzes the contrastive loss $\mathcal{L}_{\mathsf{contr}}$ we can apply Theorem 3 to see that f also minimizes the cross-entropy between $p(\tilde{\mathbf{z}}|\mathbf{z})$ and $q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ for any point $\mathbf{z}$ on $\mathcal{Z}$. This means, we can apply Proposition 4 to show that the concatenation $h = f \circ g$ is an isometry with respect to the dot product. Finally, according to Proposition 5, h must then be a composition of an orthogonal linear transformation and a constant scaling factor. Thus, f recovers the latent sources up to orthogonal linear transformations, concluding the proof. □

Extension of theory to subspaces of $\mathbb{R}^N$

Here, we show how one can generalize the theory above from $\mathcal{Z} = \mathbb{S}^{N-1}$ to $\mathcal{Z} \subseteq \mathbb{R}^N$. Under mild assumptions regarding the ground-truth conditional distribution p and the model distribution q_h, we prove that all minimizers of the cross-entropy between p and q_h are linear functions, if $\mathcal{Z}$ is a convex body. Note that the hyperrectangle $[a_1, b_1] \times \ldots \times [a_N, b_N]$ is an example of such a convex body.

Assumptions

First, we restate the core assumptions for this proof. The main difference to the assumptions for the hyperspherical case above is that we assume different conditional distributions: instead of rotation-invariant von Mises-Fisher distributions, we use translation-invariant distributions (up to restrictions determined by the finite size of the space) of the exponential family.

Generative process Let $g : \mathcal{Z} \to \mathcal{X}$ be an injective function between the two spaces $\mathcal{Z} \subseteq \mathbb{R}^N$ and $\mathcal{X} \subseteq \mathbb{R}^K$ with $K \geq N$ and where $\mathcal{Z}$ is a convex body (e.g., a hyperrectangle). Further, let the marginal distribution be uniform, i.e., $p(\mathbf{z}) = |\mathcal{Z}|^{-1}$. We assume that the conditional distribution over positive pairs $p(\tilde{\mathbf{z}}|\mathbf{z})$ is an exponential distribution

$$\begin{aligned} p(\tilde{\mathbf{z}}|\mathbf{z}) &= C_p^{-1}(\mathbf{z}) e^{-\lambda \delta(\tilde{\mathbf{z}}, \mathbf{z})} \\ \text{with} \quad C_p(\mathbf{z}) &:= \int e^{-\lambda \delta(\mathbf{z}, \tilde{\mathbf{z}})} \, \mathrm{d}\tilde{\mathbf{z}}, \end{aligned} \tag{E.34}$$

where $\lambda > 0$ a parameter controlling the width of the distribution and δ is a (semi-)metric. If δ is a semi-metric, i.e., it does not fulfill the triangle inequality, there must exist a metric δ' such that δ can be written as the composition of a continuously invertible map $j : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $j(0) = 0$ and the metric, i.e., $\delta = j \circ \delta'$. Finally, we assume that during training one has access to samples from both of these distributions.

Note that unlike for the hypersphere, when sampling positive pairs $\mathbf{z}, \tilde{\mathbf{z}} \sim p(\mathbf{z})p(\tilde{\mathbf{z}}|\mathbf{z})$, it is no longer guaranteed that the marginal distributions of $\mathbf{z}$ and $\tilde{\mathbf{z}}$ are the same. When referencing the density functions – or using them in expectation values – $p(\cdot)$ will always denote the same marginal density, no matter if the argument is $\mathbf{z}$ or $\tilde{\mathbf{z}}$. Specifically, $p(\tilde{\mathbf{z}})$ does not refer to $\int p(\mathbf{z})p(\tilde{\mathbf{z}}|\mathbf{z})d\mathbf{z}$.

Model Let $\mathcal{Z}'$ be a subset of $\mathbb{R}^N$ that is a convex body and let $f : \mathcal{X} \to \mathcal{Z}'$ be the model whose parameters are optimized. We associate a conditional distribution $q_\mathrm{h}(\tilde{\mathbf{z}}|\mathbf{z})$ with our model f through

$$\begin{aligned} q_\mathrm{h}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_q^{-1}(\mathbf{z}) e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \\ \text{with} \quad C_q(\mathbf{z}) &:= \int e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \, \mathrm{d}\tilde{\mathbf{z}}, \end{aligned} \tag{E.35}$$

where $C_q(\mathbf{z})$ is the partition function and δ is defined above.

Minimizing the cross-entropy

In a first step, we show the analogue of Proposition A for $\mathcal{Z}$ being a convex body:

Proposition 6. *For fixed $\tau > 0$, as the number of negative samples $M \to \infty$, the $\mathcal{L}_{\delta\text{-contr}}$ loss converges to*

$$\begin{aligned} \lim_{M\to\infty} \mathcal{L}_{\delta\text{-contr}}(f;\tau,M) - \log M = \\ \mathcal{L}_{\delta\text{-align}}(f;\tau) + \mathcal{L}_{\delta\text{-uni}}(f;\tau), \end{aligned} \tag{E.36}$$

where

$$\begin{aligned} \mathcal{L}_{\delta\text{-align}}(f;\tau) &:= \frac{1}{\tau} \mathop{\mathbb{E}}_{\substack{\mathbf{z}\sim p(\mathbf{z}) \\ \tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}} [\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))] \\ \mathcal{L}_{\delta\text{-uni}}(f;\tau) &:= \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log \left(\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})} \left[e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \right] \right) \right], \end{aligned} \tag{E.37}$$

and $\mathcal{L}_{\delta\text{-contr}}(f;\tau,M)$ is as defined in Eq. (6.6).

Proof. This proof is adapted from Wang and Isola (2020). By the Continuous Mapping Theorem and the law of large numbers, for any $\mathbf{x}, \tilde{\mathbf{x}}$ and $\{\mathbf{x}_i^-\}_{i=1}^M$ it follows almost surely

$$\begin{aligned} &\lim_{M\to\infty} \log \left(\frac{1}{M} e^{-\delta(f(\mathbf{x}), f(\tilde{\mathbf{x}}))/\tau} + \frac{1}{M} \sum_{i=1}^{M} e^{-\delta(f(\mathbf{x}), f(\mathbf{x}_i^-))/\tau} \right) \\ &\quad = \log \left(\mathop{\mathbb{E}}_{\mathbf{x}^-\sim p_{\text{data}}} \left[e^{-\delta(f(\mathbf{x}), f(\mathbf{x}^-))/\tau} \right] \right) \\ &\quad = \log \left(\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})} \left[e^{-\delta(h(\mathbf{z}), h(\tilde{\mathbf{z}}))/\tau} \right] \right), \end{aligned} \tag{E.38}$$

where in the last step we expressed the sample $\mathbf{x}$ and negative examples $\mathbf{x}^-$ in terms of their latent factors.

We can now express the limit of the entire loss function as

$$\begin{aligned}
&\lim_{M\to\infty} \mathcal{L}_{\delta\text{-contr}}(f;\tau,M) - \log M \\
&= \frac{1}{\tau} \mathop{\mathbb{E}}_{(\mathbf{x},\tilde{\mathbf{x}})\sim p_{\text{pos}}} [\delta(f(\mathbf{x}), f(\tilde{\mathbf{x}}))] \\
&\quad + \lim_{M\to\infty} \mathop{\mathbb{E}}_{\substack{(\mathbf{x},\tilde{\mathbf{x}})\sim p_{\text{pos}} \\ \{\mathbf{x}_i^-\}_{i=1}^M \overset{\text{i.i.d.}}{\sim} p_{\text{data}}}} \left[\log \left(\frac{1}{M} e^{-\delta(f(\mathbf{x}),f(\tilde{\mathbf{x}}))/\tau} + \frac{1}{M} \sum_{i=1}^M e^{-\delta(f(\mathbf{x}),f(\mathbf{x}^-))/\tau} \right) \right] \\
&= \frac{1}{\tau} \mathop{\mathbb{E}}_{(\mathbf{x},\tilde{\mathbf{x}})\sim p_{\text{pos}}} [\delta(f(\mathbf{x}), f(\tilde{\mathbf{x}}))] \\
&\quad + \mathop{\mathbb{E}}_{\substack{(\mathbf{x},\tilde{\mathbf{x}})\sim p_{\text{pos}} \\ \{\mathbf{x}_i^-\}_{i=1}^M \overset{\text{i.i.d.}}{\sim} p_{\text{data}}}} \left[\lim_{M\to\infty} \log \left(\frac{1}{M} e^{-\delta(f(\mathbf{x}),f(\tilde{\mathbf{x}}))/\tau} + \frac{1}{M} \sum_{i=1}^M e^{-\delta(f(\mathbf{x}),f(\mathbf{x}_i^-))/\tau} \right) \right].
\end{aligned} \tag{E.39}$$

Note that as δ is a (semi-)metric, the expression $e^{-\delta(f(\mathbf{x}),f(\tilde{\mathbf{x}}))}$ is upper-bounded by 1. Hence, according to the Dominated Convergence Theorem one can switch the limit with the expectation value in the second step. Inserting the previous results yields

$$\begin{aligned}
&= \frac{1}{\tau} \mathop{\mathbb{E}}_{(\mathbf{x},\tilde{\mathbf{x}})\sim p_{\text{pos}}} [\delta(f(\mathbf{x}), f(\tilde{\mathbf{x}}))] + \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\text{data}}} \left[\log \left(\mathop{\mathbb{E}}_{\mathbf{x}^-\sim p_{\text{data}}} \left[e^{-\delta(f(\mathbf{x}),f(\mathbf{x}^-))/\tau} \right] \right) \right] \\
&= \frac{1}{\tau} \mathop{\mathbb{E}}_{\substack{\mathbf{z}\sim p(\mathbf{z}) \\ \tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}} [\delta(h(\mathbf{z}), h(\tilde{\mathbf{z}}))] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log \left(\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})} \left[e^{-\delta(h(\mathbf{z}),h(\tilde{\mathbf{z}}))/\tau} \right] \right) \right] \\
&= \mathcal{L}_{\delta\text{-align}}(f;\tau) + \mathcal{L}_{\delta\text{-uni}}(f;\tau).
\end{aligned} \tag{E.40}$$

□

Next, we derive a property similar to Theorem 3, which suggests a practical method to find minimizers of the cross-entropy between the ground-truth p and model conditional q_{h}. This property is based on our previously introduced objective function in Eq. (6.6), which is a modified version of the InfoNCE objective in Eq. (6.1).

Theorem 5. *Let δ be a semi-metric and $\tau, \lambda > 0$ and let the ground-truth marginal distribution p be uniform. Consider a ground-truth conditional distribution $p(\tilde{\mathbf{z}}|\mathbf{z}) = C_p^{-1}(\mathbf{z}) \exp(-\lambda\delta(\tilde{\mathbf{z}},\mathbf{z}))$ and the model conditional distribution*

$$\begin{aligned}
q_{\text{h}}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_h^{-1}(\mathbf{z}) e^{-\delta(h(\tilde{\mathbf{z}}),h(\mathbf{z}))/\tau} \\
\textit{with} \quad C_h(\mathbf{z}) &:= \int_{\mathcal{Z}} e^{-\delta(h(\tilde{\mathbf{z}}),h(\mathbf{z}))/\tau} \mathrm{d}\tilde{\mathbf{z}}.
\end{aligned} \tag{E.41}$$

Then the cross-entropy between p and q_h is given by

$$\lim_{M\to\infty} \mathcal{L}_{\delta\text{-contr}}(f;\tau,M) - \log M + \log|\mathcal{Z}| = \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[H(p(\cdot|\mathbf{z}), q_\mathrm{h}(\cdot|\mathbf{z})\right], \tag{E.42}$$

which can be implemented by sampling data from the accessible distributions.
Proof. We use the definition of the cross-entropy to write

$$\mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[H(p(\cdot|\mathbf{z}), q_\mathrm{h}(\cdot|\mathbf{z})\right] \tag{E.43}$$

$$= -\mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}\left[\log(q_\mathrm{h}(\tilde{\mathbf{z}}|\mathbf{z}))\right]\right]. \tag{E.44}$$

We insert the definition of q_h and get

$$= -\mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}\left[\log(C_h^{-1}(\mathbf{z})) - \frac{1}{\tau}\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))\right]\right] \tag{E.45}$$

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}\left[\log(C_h(\mathbf{z})) + \frac{1}{\tau}\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))\right]\right]. \tag{E.46}$$

As $C_h(\mathbf{z})$ does not depend on $\tilde{\mathbf{z}}$ it can be moved out of the inner expectation value, yielding

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[\frac{1}{\tau}\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}\left[\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))\right] + \log(C_h(\mathbf{z}))\right], \tag{E.47}$$

which can be written as

$$= \frac{1}{\tau}\mathop{\mathbb{E}}_{\substack{\mathbf{z}\sim p(\mathbf{z})\\ \tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}}\left[\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))\right] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log(C_h(\mathbf{z}))\right]. \tag{E.48}$$

Inserting the definition of C_h gives

$$= \frac{1}{\tau}\mathop{\mathbb{E}}_{\substack{\mathbf{z}\sim p(\mathbf{z})\\ \tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}}\left[\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))\right] \tag{E.49}$$

$$+ \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(\int e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} d\tilde{\mathbf{z}}\right)\right]. \tag{E.50}$$

Next, the second term can be expanded by $1 = |\mathcal{Z}||\mathcal{Z}|^{-1}$, yielding

$$= \frac{1}{\tau}\mathop{\mathbb{E}}_{\substack{\mathbf{z}\sim p(\mathbf{z})\\ \tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}}\left[\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))\right] \tag{E.51}$$

$$+ \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(\int \frac{|\mathcal{Z}|}{|\mathcal{Z}|} e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} d\tilde{\mathbf{z}}\right)\right]. \tag{E.52}$$

Finally, by using that the marginal is uniform, i.e., $p(\mathbf{z}) = |\mathcal{Z}|^{-1}$, this can be simplified as

$$= \frac{1}{\tau} \mathop{\mathbb{E}}_{\substack{\mathbf{z}\sim p(\mathbf{z}) \\ \tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}}|\mathbf{z})}} [\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))] \tag{E.53}$$

$$+ \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log \left(\mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})} \left[e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \right] \right) \right] \tag{E.54}$$

$$+ \log |\mathcal{Z}| \tag{E.55}$$

$$= \lim_{M\to\infty} \mathcal{L}_{\delta\text{-contr}}(f; \tau, M) - \log M + \log p|\mathcal{Z}|. \tag{E.56}$$

□

Cross-entropy minimizers are isometries

Now we show a version of Proposition 4, that is generalized from hyperspherical spaces to (subsets of) $\mathbb{R}^N$.

Proposition 7 (Minimizers of the cross-entropy are isometries)**.** *Let δ be a semi-metric. Consider the conditional distributions of the form $p(\tilde{\mathbf{z}}|\mathbf{z}) = C_p^{-1}(\mathbf{z}) \exp(-\delta(\tilde{\mathbf{z}}, \mathbf{z})/\lambda)$ and*

$$\begin{aligned} q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_h^{-1}(\mathbf{z}) e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \\ \textit{with} \quad C_h(\mathbf{z}) &:= \int_{\mathcal{Z}} e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \mathrm{d}\tilde{\mathbf{z}}, \end{aligned} \tag{E.57}$$

where the hypothesis class for h is assumed to be sufficiently flexible such that $p(\tilde{\mathbf{z}}|\mathbf{z})$ and $q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ can match for any point $\mathbf{z}$. If h is a minimizer of the cross-entropy $\mathcal{L}_{\mathsf{CE}} = \mathbb{E}_{p(\tilde{\mathbf{z}}|\mathbf{z})}[-\log q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})]$, then h is an isometry, i.e., $\forall \mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z} : \lambda\tau\delta(\mathbf{z}, \tilde{\mathbf{z}}) = \delta(h(\mathbf{z}), h(\tilde{\mathbf{z}}))$.

Proof. Note that $q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z})$ is powerful enough to match $p(\tilde{\mathbf{z}}|\mathbf{z})$ for the correct choice of h, e.g. the identity. The global minimum of cross-entropy between two distributions is reached if they match by value and have the same support. Hence, if p is a regular density, q_{h} will be a regular density, i.e., q_{h} is continuous and has only finite values $0 \leq q_{\mathrm{h}} < \infty$. As the two distributions match, this means

$$p(\tilde{\mathbf{z}}|\mathbf{z}) = q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z}). \tag{E.58}$$

This expression also holds true for $\tilde{\mathbf{z}} = \mathbf{z}$; additionally using the property $\delta(\mathbf{z}, \mathbf{z}) = 0$ yields

$$p(\mathbf{z}|\mathbf{z}) = q_{\mathrm{h}}(\mathbf{z}|\mathbf{z}) \tag{E.59}$$

$$\Leftrightarrow \quad C_p^{-1}(\mathbf{z}) e^{-\delta(\mathbf{z},\mathbf{z})/\lambda} = C_h^{-1}(\mathbf{z}) e^{-\delta(h(\mathbf{z}), h(\mathbf{z}))/\tau} \tag{E.60}$$

$$\Leftrightarrow \quad C_p(\mathbf{z}) = C_h(\mathbf{z}). \tag{E.61}$$

As the normalization constants are identical, we obtain for all $\mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z}$

$$e^{-\delta(\tilde{\mathbf{z}},\mathbf{z})/\lambda} = e^{-\delta(h^*(\tilde{\mathbf{z}}),h^*(\mathbf{z}))/\tau} \tag{E.62}$$

$$\Leftrightarrow \qquad \delta(\tilde{\mathbf{z}},\mathbf{z}) = \frac{\lambda}{\tau}\delta(h^*(\tilde{\mathbf{z}}),h^*(\mathbf{z})). \tag{E.63}$$

By introducing a new semi-metric $\delta' := \lambda\tau^{-1}\delta$, we can write this as $\delta(\tilde{\mathbf{z}},\mathbf{z}) = \delta'(h(\tilde{\mathbf{z}}),h(\mathbf{z}))$, which shows that h is an isometry. If there is no model mismatch, i.e., $\lambda = \tau$, this means $\delta(\mathbf{z},\tilde{\mathbf{z}}) = \delta(h(\mathbf{z}),h(\tilde{\mathbf{z}}))$. □

Note, that this result does not depend on the choice of $\mathcal{Z}$ but just on the class of conditional distributions allowed.

Cross-entropy minimization identifies the ground-truth factors

Before we continue, let us recall a Theorem by Mankiewicz (1972):

Theorem C (Mankiewicz, 1972)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be normed linear spaces and let $\mathcal{V}$ be a convex body in $\mathcal{X}$ and $\mathcal{W}$ a convex body in $\mathcal{Y}$. Then every surjective isometry between $\mathcal{V}$ and $\mathcal{W}$ can be uniquely extended to an affine isometry between $\mathcal{X}$ and $\mathcal{Y}$.*
Proof. See Mankiewicz (1972). □

In addition, it is known that isometries on closed spaces are bijective:

Lemma A. *Assume h is an isometry of the closed space $\mathcal{Z}$ into itself, i.e., $\forall \mathbf{z}, \tilde{\mathbf{z}} : \delta(\mathbf{z},\tilde{\mathbf{z}}) = \delta(h(\mathbf{z}),h(\tilde{\mathbf{z}}))$. Then h is bijective.*

Proof. See Lemma (2.6) in Całka (1982) for surjectivity. We show the injectivity by contradiction. Assume h is not injective. Then we can find a point $\tilde{\mathbf{z}} \neq \mathbf{z}$ where $h(\mathbf{z}) = h(\tilde{\mathbf{z}})$. But then $\delta(\mathbf{z},\tilde{\mathbf{z}}) > \delta(\mathbf{z},\mathbf{z})$ and $\delta(h(\mathbf{z}),h(\tilde{\mathbf{z}})) = \delta(h(\mathbf{z}),h(\mathbf{z})) = 0$ by the properties of δ. Hence, h is injective. □

Before continuing, we need to generalize the class of functions we consider as distance measures:

Lemma 4. *Let δ' be a the composition of a continuously invertible function $j : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $j(0) = 0$ and a metric δ, i.e., $\delta' := j \circ \delta$. Then, (i) δ' is a semi-metric and (ii) if a function $h : \mathbb{R}^n \to \mathbb{R}^n$ is an isometry of a space with the semi-metric δ', it is also an isometry of the space with the metric δ.*

Proof. (i) Let $\mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z}$. Per assumption j must be strictly monotonically increasing on $\mathbb{R}_{\geq 0}$. Since δ is a metric it follows $\delta(\mathbf{z},\tilde{\mathbf{z}}) \geq 0 \Rightarrow \delta'(\mathbf{z},\tilde{\mathbf{z}}) = j(\delta(\mathbf{z},\tilde{\mathbf{z}})) \geq 0$, with equality iff $\mathbf{z} = \tilde{\mathbf{z}}$. Furthermore, since δ is a metric it is symmetric in its arguments and, hence, δ' is symmetric in its arguments. Thus, δ' is a semi-metric.

(ii) h is an isometry of a space with the semi-metric δ', allowing to derive that for all $\mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{Z}$,

$$\delta'(h(\mathbf{z}), h(\tilde{\mathbf{z}})) = \delta'(\mathbf{z}, \tilde{\mathbf{z}}) \tag{E.64}$$

$$j(\delta(h(\mathbf{z}), h(\tilde{\mathbf{z}}))) = j(\delta(\mathbf{z}, \tilde{\mathbf{z}})) \tag{E.65}$$

and, applying the inverse j^{-1} which exists by assumption, yields

$$\delta(h(\mathbf{z}), h(\tilde{\mathbf{z}})) = \delta(\mathbf{z}, \tilde{\mathbf{z}}), \tag{E.66}$$

concluding the proof. □

By combining the properties derived before we can show that h is an affine function:

Theorem 6. *Let $\mathcal{Z} = \mathcal{Z}'$ be a convex body in $\mathbb{R}^N$. Let the mixing function g be differentiable and invertible. If the assumed form of q_{h} as defined in Eq. (E.35) matches that of p, and if f is differentiable and minimizes the cross-entropy between p and q_{h}, then we find that $h = f \circ g$ is affine, i.e., we recover the latent sources up to affine transformations.*

Proof. According to Proposition 7 h is an isometry and q_{h} is a regular probability density function. If the distance δ used in the conditional distributions p and q_{h} is a semi-metric as in Lemma 4, it follows that h is also an isometry for a proper metric. This also means that h is bijective according to Lemma A. Finally, Theorem C says that h is an affine transformation. □

We use the assumption that the marginal $p(\mathbf{z})$ is uniform, to show

Theorem 7. *Let $\mathcal{Z}$ be a convex body in $\mathbb{R}^N$, $h = f \circ g : \mathcal{Z} \to \mathcal{Z}$, and δ be a metric or a semi-metric as defined in Lemma 4. Further, let the ground-truth marginal distribution be uniform and the conditional distribution be as (6.5). Let the mixing function g be differentiable and injective. If the assumed form of q_{h} matches that of p, i.e.,*

$$\begin{aligned} q_{\mathrm{h}}(\tilde{\mathbf{z}}|\mathbf{z}) &= C_q^{-1}(\mathbf{z}) e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \\ \textit{with} \quad C_q(\mathbf{z}) &:= \int e^{-\delta(h(\tilde{\mathbf{z}}), h(\mathbf{z}))/\tau} \, \mathrm{d}\tilde{\mathbf{z}}, \end{aligned} \tag{E.67}$$

and if f is differentiable and minimizes the $\mathcal{L}_{\delta\text{-contr}}$ objective in (6.6) for $M \to \infty$, we find that $h = f \circ g$ is invertible and affine, i.e., we recover the latent sources up to affine transformations.

Proof. According to Theorem 5 h minimizes the cross-entropy between p and q_{h} as defined in Eq. (6.4). Then according to Theorem 6, h is an affine transformation. □

This result can be seen as a generalized version of Theorem 4, as it is valid for any convex body $\mathcal{Z} \subseteq \mathbb{R}^N$ and allows a larger variety of conditional distributions. A missing step is to extend this theory beyond uniform marginal distributions. This will be addressed in future work.

Under some assumptions we can further narrow down possible forms of h, thus, showing that h in fact solves the nonlinear ICA problem only up to permutations and elementwise transformations.

For this, let us first repeat a result from Li and So (1994), that shows an important property of isometric matrices:

Theorem D. *Suppose $1 \leq \alpha \leq \infty$ and $\alpha \neq 2$. An $n \times n$ matrix $\mathbf{A}$ is an isometry of L^α-norm if and only if $\mathbf{A}$ is a generalized permutation matrix, i.e., $\forall \mathbf{z} : (\mathbf{A}\mathbf{z})_\mathbf{i} = \mathbf{f}\mathbf{f}_\mathbf{i}\mathbf{z}_{\boldsymbol{\sigma}(\mathbf{i})}$, with $\alpha_i = \pm 2$ and σ being a permutation.*

Proof. See Li and So (1994). Note that this can also be concluded from the Banach-Lamperti Theorem (Lamperti et al., 1958). □

Leveraging this insight, we can finally show:

Theorem 8. *Let $\mathcal{Z}$ be a convex body in $\mathbb{R}^N$, $h : \mathcal{Z} \to \mathcal{Z}$, and δ be an L^α metric for $\alpha \geq 1, \alpha \neq 2$ or the α-th power of such an L^α metric. Further, let the ground-truth marginal distribution be uniform and the conditional distribution be as in Eq. (6.5), and let the mixing function g be differentiable and invertible. If the assumed form of $q_\mathrm{h}(\cdot|\mathbf{z})$ matches that of $p(\cdot|\mathbf{z})$, i.e., both use the same metric δ up to a constant scaling factor, and if f is differentiable and minimizes the $\mathcal{L}_{\delta\text{-contr}}$ objective in Eq. (6.6) for $M \to \infty$ we find that $h = f \circ g$ is a composition of input independent permutations, sign flips and rescalings.*

Proof. First, we prove the case where both conditional distributions use exactly the same metric. By Theorem 7 h is an affine transformation. Moreover, according to Proposition 7 is an isometry. Thus, by Theorem D, h is a generalized permutation matrix, i.e., a composition of permutations and sign flips.

Finally, for the case that δ matches the similarity measure in the ground-truth conditional distribution defined in Eq. (6.5) (denoted as δ^*) only up to a constant rescaling factor r, we know

$$\begin{aligned} &\forall \mathbf{z}, \tilde{\mathbf{z}} : \delta^*(\mathbf{z}, \tilde{\mathbf{z}}) = \delta(h(\mathbf{z}), h(\tilde{\mathbf{z}})) \\ \Leftrightarrow\ &\delta^*(\mathbf{z}, \tilde{\mathbf{z}}) = \delta^*\left(\frac{1}{r}h(\mathbf{z}), \frac{1}{r}h(\tilde{\mathbf{z}})\right). \end{aligned} \tag{E.68}$$

Thus, $\frac{1}{r}h$ is a δ^* isometry and the same argument as above holds, concluding the proof. □

Experimental details

For the experiments presented in Sec. 6 we train our feature encoder for 300 000 iterations with a batch size of 6144 utilizing Adam (Kingma & Ba, 2014) with a learning rate of 10^{-4}. Like Hyvärinen and Morioka (2016, 2017), for the mixing network, we i)

Table E.1: Identifiability up to affine transformations on the training set of 3DIdent. Mean ± standard deviation over 3 random seeds. As earlier, only the first row corresponds to a setting that matches the theoretical assumptions for linear identifiability; the others show distinct violations. Supervised training with unbounded space achieves scores of $R^2 = (99.98 \pm 0.01)\%$ and MCC $= (99.99 \pm 0.01)\%$. The last row refers to using the SimCLR (Chen et al., 2020a) augmentations to generate positive pairs. The last row refers to using the image augmentations suggested by Chen et al. (2020a) to generate positive image pairs; for details see Sec. E. In contrast to Table 6.4, the scores here are reported on the same data the models were trained on.

Dataset	Model f			Identity [%]	Unsupervised [%]	
$p(\cdot\|\cdot)$	Space	$q_{\mathrm{h}}(\cdot\|\cdot)$	M.	R^2	R^2	MCC
Normal	Box	Normal	✓	5.35 ± 0.72	97.83 ± 0.13	98.85 ± 0.07
Normal	Unbounded	Normal	✗	—"—	97.72 ± 0.02	55.90 ± 2.22
Laplace	Box	Normal	✗	—"—	97.95 ± 0.05	98.94 ± 0.03
Normal	Sphere	vMF	✗	—"—	66.73 ± 0.03	42.72 ± 3.20
Augm.	Sphere	vMF	✗	—"—	45.94 ± 1.80	47.6 ± 1.45

use 0.2 for the angle of the negative slope[2], ii) use L^2 normalized weight matrices with minimum condition number of 25 000 uniformly distributed samples. For the encoder, we i) use the default (0.01) negative slope ii) use 6 hidden layers with dimensionality $[N \cdot 10, N \cdot 50, N \cdot 50, N \cdot 50, N \cdot 50, N \cdot 10]$ and iii) initialize the normalization magnitude as 1. We sample 4096 latents from the marginal for evaluation. For MCC (Hyvärinen & Morioka, 2016, 2017) we use the Pearson correlation coefficient[3]; we found there to be no difference with Spearman[4].

For the experiments presented in Sec. 6, we use the same architecture as the encoder in (Klindt et al., 2021a). As in (Klindt et al., 2021a), we train for 300 000 iterations with a batch size of 64 utilizing Adam (Kingma & Ba, 2014) with a learning rate of 10^{-4}. For evaluation, as in (Klindt et al., 2021a), we use 10 000 samples and the Spearman correlation coefficient.

For the experiments presented in Sec. 6, we train the feature encoder for 200 000 iterations using Adam with a learning rate of 10^{-4}. For the encoder we use a ResNet18 (He et al., 2016b) architecture followed by a single hidden layer with dimensionality $N \cdot 10$ and LeakyReLU activation function using the default (0.01) negative slope. The scores on the training set are evaluated on 10% of the whole training set, 25 000 random samples. The test set consists of 25 000 samples not included in the training set. For the last row of Tab. 6.4 and Tab. E.1 we used the best-working combination of image augmentations found by Chen et al. (2020a) to sample positive pairs. To be precise, we used a random crop and resize operation followed by a color distortion augmentation. The random crops had a uniformly distributed size (between 8% and 100% of the original image area) and a random aspect ration (between 3/4 and 4/3); subsequently, they were resized to the original image dimension (224×224) again. The

[2]See e.g. https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html
[3]See e.g. https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html
[4]See e.g. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html

color distortion operation itself combined color jittering (i.e., random changes of the brightness, contrast, saturation and hue) with color dropping (i.e., random grayscale conversations). We used the same parameters for these augmentations as recommended by Chen et al. (2020a).

The experiments in Sec. 6 took on the order of 5-10 hours on a GeForce RTX 2080 Ti GPU, the experiments on KITTI Masks took 1.5 hours on a GeForce RTX 2080 Ti GPU and those on 3DIdent took 28 hours on four GeForce RTX 2080 Ti GPUs. The creation of the 3DIdent dataset additionally required approximately 150 hours of compute time on a GeForce RTX 2080 Ti.

Details on 3DIdent

We build on the rendering pipeline of Johnson et al. (2017b) and use the Blender engine (Blender Online Community, 2021), as of version 2.91.0, for image rendering. The scenes depicted in the dataset show a rotated and translated object onto which a spotlight is directed. The spotlight is located on a half-circle above the scene and shines down. The scenes can be described by 10 parameters: the position of the object along the X-, Y- and Z-axis, the rotation of the object described by Euler angles (3), the position of the spotlight described by a polar angle, and the hue of the object, the ground and the spotlight. The value range is $[-3, 3]$ for all position parameters, and is $[-\pi/2, \pi/2]$ for the remaining parameters. The parameters are sampled from a 10-dimensional unit hyperrectangle, then rescaled to their corresponding value range. This ensures that the variance of the latent factors is the same for all latent dimensions.

To ensure that the generative process is injective, we take two measures: First, we use a non-rotationally symmetric object (Utah tea pot, Newell, 1975), thus the rotation information is unambiguous. Second, we use different levels of color saturation for the object, the spotlight and the ground (1.0, 0.8 and 0.6, respectively), thus the object is always distinguishable from the ground.

Comparison to existing datasets

The proposed dataset contains high-resolution renderings of an object in a 3D scene. It features some aspects of natural scenes, e.g. complex 3D objects, different lighting conditions and continuous variables. Existing benchmarks (Burgess & Kim, 2018; Dittadi et al., 2021; Gondal et al., 2019; Klindt et al., 2021a) for disentanglement in 3D scenes differ in important aspects to 3DIdent.

KITTI Masks (Klindt et al., 2021a) only enables evaluating identification of the two-dimensional position and scale of the object instance. In addition, the observed segmentation masks are significantly lower resolution than examples in our dataset. 3D Shapes (Burgess & Kim, 2018) and MPI3D (Gondal et al., 2019) are rendered at the same resolution (64×64) as KITTI Masks. Whereas the dataset contributed by (Dittadi et al., 2021) is rendered at $2\times$ that resolution (128×128), our dataset is rendered at $3.5\times$ that

resolution (224×224), the resolution at which natural image classification is typically evaluated (Deng et al., 2009b). With that being said, we do note that KITTI Masks is unique in containing frames of natural video, and we thus consider it complementary to 3DIdent.

Burgess and Kim (2018), Dittadi et al. (2021), and Gondal et al. (2019) contribute datasets which contain variable object rotations around one, one, and two rotation axes, respectively, while 3DIdent contains variable object rotation around all three rotation axes as well as variable lighting conditions. Furthermore, each of these datasets were generated by sampling latent factors from an equidistant grid, thus only covering a limited number values along each axis of variation, effectively resulting in a highly coarse discretization of naturally continuous variables. As 3DIdent instead samples the latent factors uniformly in the latent space, this better reflects the continuous nature of the latent dimensions.

Effects of the Uniformity Loss

In previous work, Wang and Isola (2020) showed that a part of the contrastive (InfoNCE) loss — the uniformity loss — effectively ensures that the encoded features are uniformly distributed over a hypersphere. We now show that this part is crucial to ensure that the mapping is bijective. More precisely, we demonstrate that if the distribution of the encoded/reconstructed latents $h(\mathbf{z})$ has the same support as the distribution of $\mathbf{z}$, and both distributions are regular, i.e., their densities are non-zero and finite, then the transformation h is bijective.

First, we focus on the more general case of a map between manifolds:

Proposition 8. *Let $\mathcal{M}, \mathcal{N}$ be simply connected and oriented $\mathcal{C}^1$ manifolds without boundaries and $h : \mathcal{M} \to \mathcal{N}$ be a differentiable map. Further, let the random variable $\mathbf{z} \in \mathcal{M}$ be distributed according to $\mathbf{z} \sim p(\mathbf{z})$ for a regular density function p, i.e., $0 < p < \infty$. If the pushforward $p_{\#h}(\mathbf{z})$ of p through h is also a regular density, i.e., $0 < p_{\#h} < \infty$, then h is a bijection.*

Proof. We begin by showing by contradiction that the Jacobian determinant of h does not vanish, i.e., $|\det J_h| > 0$:

Suppose that the Jacobian determinant $|\det J_h|$ vanishes for some $\mathbf{z} \in \mathcal{M}$. Then the inverse of the Jacobian determinant goes to infinity at this point and so does the density of $h(\mathbf{z})$ according to the well-known transformation of probability densities. By assumption, both p and $p_{\#h}$ must be regular density functions and, thus, be finite. This contradicts the initial assumption and so the Jacobian determinant $|\det J_h|$ cannot vanish.

Next, we show that the mapping h is proper. Note that a map is called proper if pre-images of compact sets are compact (Ruzhansky & Sugimoto, 2015). Firstly, a continuous mapping between $\mathcal{M}$ and $\mathcal{N}$ is also closed, i.e., pre-images of closed subsets are also closed (Lee, 2013b). In addition, it is well-known that continuous functions

on compact sets are bounded. Lastly, according to the Heine–Borel theorem, compact subsets of $\mathbb{R}^D$ are closed and bounded. Taken together, this shows that h is proper.

Finally, according to Theorem 2.1 in (Ruzhansky & Sugimoto, 2015) a proper h with non-vanishing Jacobian determinant is bijective, concluding the proof. □

This theorem directly applies to the case of hyperspheres, which are simply connected and oriented manifolds without boundary. This yields:

Corollary 1. *Let $\mathcal{Z}$ be a hypersphere and $h : \mathcal{Z} \to \mathcal{Z}$ be a differentiable map. Further, let the marginal distribution $p(\mathbf{z})$ of the variable $\mathbf{z} \in \mathcal{Z}$ be a regular density function, i.e., $0 < p < \infty$. If the pushforward $p_{\#h}$ of p through h is also a regular density, i.e., $0 < p_{\#h} < \infty$, then h is a bijection.*

Therefore, we can conclude that a loss term ensuring that the encoded features are distributed according to a regular density function, such as the uniformity term, makes the map h bijective and prevents an information loss. Note that this does not assume that the marginal distribution of the ground-truth latents $p(\mathbf{z})$ is uniform but only that it is regular and non-vanishing.

Note that while the proposition shows that the uniformity loss is sufficient to ensure bijectivity, we can construct counterexamples if its assumptions (like differentiability) are violated even in just a single point. For instance, the requirement of h being fully differentiable is most likely violated in large unregularized neural networks with ReLU nonlinearities. Here, one might need the full contrastive loss to ensure bijectivity of h.

F

Learnable latent embeddings for joint behavioral and neural analysis

Supplementary Tables

Table F.1: **Consistency statistics related to Fig. 7.1.** Data includes all rats (n=4, with 10 seeds per model run averaged first); one-way ANOVA F(24.5) $p = 1.92x10^{-16}$.

group 1	group 2	P-value	Reject
CEBRA-Behavior	CEBRA-Time	0.0108	**True**
CEBRA-Behavior	conv-pi-VAE w/labels	0.0021	**True**
CEBRA-Behavior	conv-pi-VAE without	0.0	**True**
CEBRA-Behavior	tSNE	0.0	**True**
CEBRA-Behavior	UMAP	0.0	**True**
CEBRA-Behavior	autoLFADS	0.0	**True**
CEBRA-Time	conv-pi-VAE w/labels	0.9988	False
CEBRA-Time	conv-pi-VAE without	0.0001	**True**
CEBRA-Time	tSNE	.4808	False
CEBRA-Time	UMAP	0.0	**True**
CEBRA-Time	autoLFADS	0.1129	False

Table F.2: **Decoding statistics related to Fig. 7.6.** Data includes all rats (n=4); supervised grouping one way ANOVA F(55) p=4.7e-31; self- and unsupervised, one way ANOVA F(14.7) $p = 1.53x10^{-10}$. Posthoc Tukey HSD Tests:

group 1	group 2	P-value	Reject
CEBRA-Behavior	conv-pi-VAE (MC decoding)	0.9	False
CEBRA-Behavior	conv-pi-VAE (kNN)	0.001	**True**
CEBRA-Behavior	pi-VAE (MC decoding)	0.001	**True**
CEBRA-Behavior	pi-VAE (kNN)	0.001	**True**
CEBRA-Time	autoLFADS	0.0	**True**
CEBRA-Time	PCA	0.0	**True**
CEBRA-Time	tSNE	0.0174	**True**
CEBRA-Time	UMAP	0.0368	**True**

Table F.3: **Related to Fig. 7.15.** One-way ANOVA F(3, 197)=5.88, $p = 0.0007$ and posthoc Tukey HSD tests. Allen Neuropixels dataset, 1 Frame window (below 50 neurons all False):

neuron no.	group 1	group 2	P-value	Reject
50	baseline-bayes	baseline-knn	0.0233	**True**
50	baseline-bayes	CEBRA	0.8918	False
50	baseline-knn	CEBRA	0.0055	True
50	baseline-bayes	CEBRA-joint	0.8238	False
50	baseline-knn	CEBRA-joint	0.004	**True**
50	CEBRA	CEBRA-joint	0.9987	False
100	baseline-bayes	baseline-knn	0.0	**True**
100	baseline-bayes	CEBRA	0.0275	**True**
100	baseline-knn	CEBRA	0.0132	**True**
100	baseline-bayes	CEBRA-joint	0.9977	False
100	baseline-knn	CEBRA-joint	0.0	**True**
100	CEBRA	CEBRA-joint	0.0395	**True**
200	baseline-bayes	baseline-knn	0.0005	**True**
200	baseline-bayes	CEBRA	0.3703	False
200	baseline-knn	CEBRA	0.0	**True**
200	baseline-bayes	CEBRA-joint	0.0058	**True**
200	baseline-knn	CEBRA-joint	0.0	**True**
200	CEBRA	CEBRA-joint	0.1478	False
400	baseline-bayes	baseline-knn	0.0044	**True**
400	baseline-bayes	CEBRA	0.0336	**True**
400	baseline-knn	CEBRA	0.0	**True**
400	baseline-bayes	CEBRA-joint	0.0	**True**
400	baseline-knn	CEBRA-joint	0.0	**True**
400	CEBRA	CEBRA-joint	0.0	**True**
600	baseline-bayes	baseline-knn	0.0125	**True**
600	baseline-bayes	CEBRA	0.6063	False
600	baseline-knn	CEBRA	0.001	**True**
600	baseline-bayes	CEBRA-joint	0.0	**True**
600	baseline-knn	CEBRA-joint	0.0	**True**
600	CEBRA	CEBRA-joint	0.0	**True**
800	baseline-bayes	baseline-knn	0.0006	**True**
800	baseline-bayes	CEBRA	0.0008	**True**
800	baseline-knn	CEBRA	0.0	**True**
800	baseline-bayes	CEBRA-joint	0.0	**True**
800	baseline-knn	CEBRA-joint	0.0	**True**
800	CEBRA	CEBRA-joint	0	**True**
900	baseline-bayes	baseline-knn	0.0004	**True**
900	baseline-bayes	CEBRA	0.0048	**True**
900	baseline-knn	CEBRA	0.0	**True**
900	baseline-bayes	CEBRA-joint	0.0	**True**
900	baseline-knn	CEBRA-joint	0.0	**True**
900	CEBRA	CEBRA-joint	0.0	**True**
1000	baseline-bayes	baseline-knn	0.0	**True**
1000	baseline-bayes	CEBRA	0.0	**True**
1000	baseline-knn	CEBRA	0.0	**True**
1000	baseline-bayes	CEBRA-joint	0.0	**True**
1000	baseline-knn	CEBRA-joint	0.0	**True**
1000	CEBRA	CEBRA-joint	0.0019	**True**

Table F.4: **Related to Fig. 7.15.** One-way ANOVA F(3, 197)=1.29, p=0.279 and posthoc Tukey HSD tests. Allen Neuropixels dataset, 10 Frame window (< 30 neurons= all were False). Note, here CEBRA vs. CEBRA-joint is not significant.

neuron no.	group 1	group 2	P-value	Reject
30	baseline-bayes	baseline-knn	0.016	**True**
30	baseline-bayes	CEBRA	0.1255	False
30	baseline-knn	CEBRA	0.7083	False
30	baseline-bayes	CEBRA-joint	0.0072	**True**
30	baseline-knn	CEBRA-joint	0.9784	False
30	CEBRA	CEBRA-joint	0.4762	False
50	baseline-bayes	baseline-knn	0.0358	**True**
50	baseline-bayes	CEBRA	0.324	False
50	baseline-knn	CEBRA	0.5956	False
50	baseline-bayes	CEBRA-joint	0.1296	False
50	baseline-knn	CEBRA-joint	0.8989	False
50	CEBRA	CEBRA-joint	0.9379	False
100	baseline-bayes	baseline-knn	0.0	**True**
100	baseline-bayes	CEBRA	0.2589	False
100	baseline-knn	CEBRA	0.0002	**True**
100	baseline-bayes	CEBRA-joint	0.372	False
100	baseline-knn	CEBRA-joint	0.0001	**True**
100	CEBRA	CEBRA-joint	0.9941	False
200	baseline-bayes	baseline-knn	0.0	**True**
200	baseline-bayes	CEBRA	0.9976	False
200	baseline-knn	CEBRA	0.0	**True**
200	baseline-bayes	CEBRA-joint	0.7999	False
200	baseline-knn	CEBRA-joint	0.0	**True**
200	CEBRA	CEBRA-joint	0.6964	False
400	baseline-bayes	baseline-knn	0.0004	**True**
400	baseline-bayes	CEBRA	0.2531	False
400	baseline-knn	CEBRA	0.0	**True**
400	baseline-bayes	CEBRA-joint	0.2166	False
400	baseline-knn	CEBRA-joint	0.0	**True**
400	CEBRA	CEBRA-joint	0.9996	False
600	baseline-bayes	baseline-knn	0.0002	**True**
600	baseline-bayes	CEBRA	0.2095	False
600	baseline-knn	CEBRA	0.0	**True**
600	baseline-bayes	CEBRA-joint	0.2884	False
600	baseline-knn	CEBRA-joint	0.0	**True**
600	CEBRA	CEBRA-joint	0.9967	False
800	baseline-bayes	baseline-knn	0.0001	**True**
800	baseline-bayes	CEBRA	0.3691	False
800	baseline-knn	CEBRA	0.0	**True**
800	baseline-bayes	CEBRA-joint	0.1668	False
800	baseline-knn	CEBRA-joint	0.0	**True**
800	CEBRA	CEBRA-joint	0.9524	False
900	baseline-bayes	baseline-knn	0.0003	**True**
900	baseline-bayes	CEBRA	0.3867	False
900	baseline-knn	CEBRA	0.0	**True**
900	baseline-bayes	CEBRA-joint	0.3522	False
900	baseline-knn	CEBRA-joint	0.0	**True**
900	CEBRA	CEBRA-joint	0.9999	False
1000	baseline-bayes	baseline-knn	0.0018	**True**
1000	baseline-bayes	CEBRA	0.4707	False
1000	baseline-knn	CEBRA	0.0001	**True**
1000	baseline-bayes	CEBRA-joint	0.3785	False
1000	baseline-knn	CEBRA-joint	0.0001	**True**
1000	CEBRA	CEBRA-joint	0.9981	False

Table F.5: **Related to Fig. 7.15** One-way ANOVA F(3, 197) = 15.73, $p = 3.31x10^{-9}$ and posthoc Tukey HSD test. Allen Neuropixels dataset, scene classification with 1 Frame window:

neuron no.	group 1	group 2	P-value	Reject
50	baseline-bayes	baseline-knn	0.4575	False
50	baseline-bayes	CEBRA	0.1986	False
50	baseline-knn	CEBRA	0.9355	False
50	baseline-bayes	CEBRA-joint	0.0	**True**
50	baseline-knn	CEBRA-joint	0.0	**True**
50	CEBRA	CEBRA-joint	0.0	**True**
100	baseline-bayes	baseline-knn	0.0207	**True**
100	baseline-bayes	CEBRA	0.0084	**True**
100	baseline-knn	CEBRA	0.9694	False
100	baseline-bayes	CEBRA-joint	0.0	**True**
100	baseline-knn	CEBRA-joint	0.0	**True**
100	CEBRA	CEBRA-joint	0.0	**True**
200	baseline-bayes	baseline-knn	0.0106	**True**
200	baseline-bayes	CEBRA	0.0001	**True**
200	baseline-knn	CEBRA	0.1269	False
200	baseline-bayes	CEBRA-joint	0.0	**True**
200	baseline-knn	CEBRA-joint	0.0	**True**
200	CEBRA	CEBRA-joint	0.0	**True**
400	baseline-bayes	baseline-knn	0.0047	**True**
400	baseline-bayes	CEBRA	0.0001	**True**
400	baseline-knn	CEBRA	0.239	False
400	baseline-bayes	CEBRA-joint	0.0	**True**
400	baseline-knn	CEBRA-joint	0.0	**True**
400	CEBRA	CEBRA-joint	0.0	**True**
600	baseline-bayes	baseline-knn	0.0013	**True**
600	baseline-bayes	CEBRA	0.0	**True**
600	baseline-knn	CEBRA	0.0032	**True**
600	baseline-bayes	CEBRA-joint	0.0	**True**
600	baseline-knn	CEBRA-joint	0.0	**True**
600	CEBRA	CEBRA-joint	0.0	**True**
800	baseline-bayes	baseline-knn	0.0	**True**
800	baseline-bayes	CEBRA	0.0	**True**
800	baseline-knn	CEBRA	0.0	**True**
800	baseline-bayes	CEBRA-joint	0.0	**True**
800	baseline-knn	CEBRA-joint	0.0	**True**
800	CEBRA	CEBRA-joint	0.0	**True**
900	baseline-bayes	baseline-knn	0.0062	**True**
900	baseline-bayes	CEBRA	0.0	**True**
900	baseline-knn	CEBRA	0.0168	**True**
900	baseline-bayes	CEBRA-joint	0.0	**True**
900	baseline-knn	CEBRA-joint	0.0	**True**
900	CEBRA	CEBRA-joint	0.0	**True**
1000	baseline-bayes	baseline-knn	0.0002	**True**
1000	baseline-bayes	CEBRA	0.0	**True**
1000	baseline-knn	CEBRA	0.0	**True**
1000	baseline-bayes	CEBRA-joint	0.0	**True**
1000	baseline-knn	CEBRA-joint	0.0	**True**
1000	CEBRA	CEBRA-joint	0.0	**True**

Table F.6: **Related to Fig. 7.15** One-way ANOVA (10 frame window, 1000 neurons) F(3, 16) = 20.22, p = $1.09x10^{-5}$ and posthoc Tukey HSD tests. Allen Neuropixels dataset, Mean frame error, 10 frames

neuron no.	group 1	group 2	P-value	Reject
1000	baseline-bayes	baseline-knn	0.5277	False
1000	baseline-bayes	CEBRA	0.0013	**True**
1000	baseline-knn	CEBRA	0.0001	**True**
1000	baseline-bayes	CEBRA-joint	0.0011	**True**
1000	baseline-knn	CEBRA-joint	0.0001	**True**
1000	CEBRA	CEBRA-joint	0.9996	False

G

Identifiable attribution maps using regularized contrastive learning

Implementation notes

Obtaining the attribution map

Since $\mathbf{J}_{\mathbf{f}}^{+}$ identifies $\mathbf{J}_{\mathbf{g}}$ as derived in Theorem 2, we can obtain the final attribution map according to Def. 3 using

$$\hat{\mathbf{A}} = \mathbf{1}\{\max_{\mathbf{x}} |\mathbf{J}_{\mathbf{f}}^{+}(\mathbf{x})| > \epsilon\} \tag{G.1}$$

where $\epsilon > 0$ is a threshold that weights false-positive and false-negative predictions. In practice, we found that the operation

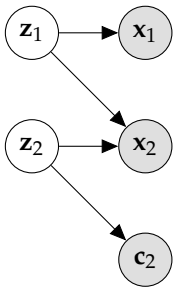
$$\hat{\mathbf{A}} = \mathbf{1}\{\sum_{\mathbf{x}} |\mathbf{J}_{\mathbf{f}}^{+}(\mathbf{x})|\} > \epsilon \tag{G.2}$$

yields even better performance, and we will use this estimation method for all experiments. In general, working on improved estimation methods taking into account sources of estimation noise could be an interesting avenue for future work.

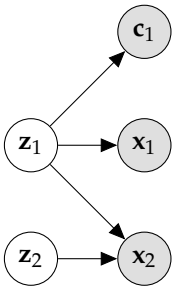
Synthetic data design

An essential aspect of our synthetic design lies in the definition of the mixing function $\mathbf{g}$ which, consequently, defines the ground truth attribution map. We split the factors $\mathbf{z}$ into two parts, $\mathbf{z}_1$ and $\mathbf{z}_2$. Figure G illustrates the two experimental configurations employed in this work. In both settings $\mathbf{z}_1$ is connected both to $\mathbf{x}_1$ and $\mathbf{x}_2$ whereas $\mathbf{z}_2$ is only be connected to $\mathbf{x}_2$. The main difference is that in the first setting $\mathbf{z}_2 = \gamma_2(\mathbf{c}_2)$

whereas in the second setting $\mathbf{z}_1 = \gamma_1(\mathbf{c}_1)$.



(a) Graphical model for the data generating process where $\mathbf{z}_2$ is observed through $\mathbf{c}_2$. The attribution map needs to be computed with respect to $\mathbf{z}_2$, which is inferred with supervised (contrastive) learning. This is the experiment setting for Table 8.1.



(b) Graphical model for the data generating process where $\mathbf{z}_1$ is observed through $\mathbf{c}_1$. Since $\mathbf{z}_2$ is not observed, the attribution map can only be estimated through the time-contrastive component in RegCL. This is the experiment setting for Table 8.2.

Detailed experimental setup

In our experiments, we consider variations of three factors. Our theory predicts that the combination of estimating the inverse of the feature encoder Jacobian with regularized training allows to identify the ground truth attribution map. We test the following factors and underline our proposed method:

Factor	Possible values
Training mode	Supervised, Supervised contrastive, Hybrid contrastive
Regularization	Off, On ($\lambda = 0.1$)
Attribution map estimation	Neuron gradient, integrated gradients, Shapley values, inverted Jacobian

Combinations of these factors can have positive effects on the output performance. We therefore run all combinations of these factors with 10 seeds (i.e., different latents and mixing functions) across different numbers of latent dimensions.

Statistical analysis

We fit an ANOVA on an ordinary least squares model using combinations of all latent factors, see Table G.1. As a post-hoc test, we use a Tukey HSD test on the statistically significant factors. See Table G.2 we show that hybrid contrastive learning computing followed by computing the pseudo-inverse significantly outperforms all other methods, and in Table G.3 we show that combining the pseudo-inverse on

regularized trained models also significantly outperforms all other methods. Statistical analysis is implemented using `statsmodels`[1].

Table G.1: Results for fitting an ANOVA on all combination of factors.

	sum sq	df	F	PR(>F)
C(attribution method name)	807.50	5	6.14	0.00
C(dim Z1)	3286.82	5	24.98	0.00
C(method name)	1505.46	2	28.60	0.00
C(extension)	15722.40	1	597.37	0.00
C(attribution method name):C(dim Z1)	456.86	25	0.69	0.85
C(attribution method name):C(method name)	8747.26	10	33.24	0.00
C(dim Z1):C(method name)	270.05	10	1.03	0.41
C(attribution method name):C(extension)	6661.36	5	50.62	0.00
C(dim Z1):C(extension)	2647.68	5	20.12	0.00
C(method name):C(extension)	2813.94	2	53.46	0.00
C(attribution method name):C(dim Z1):C(method name)	463.75	50	0.35	1.00
C(attribution method name):C(dim Z1):C(extension)	672.62	25	1.02	0.43
C(attribution method name):C(method name):C(extension)	177.68	10	0.68	0.71
C(dim Z1):C(method name):C(extension)	237.40	10	0.90	0.51
C(attribution method name):C(dim Z1):C(method name):C(extension)	932.86	50	0.71	0.93
Residual	50059.50	1902	NaN	NaN

[1]https://github.com/statsmodels/statsmodels/

Table G.2: Post-hoc test for the combination of attribution method and training method.

group1	group2	meandiff	p-adj	lower	upper	reject
J_f^+:hybrid contrastive	J_f:behavior contrastive	9.41	0.00	5.83	12.98	True
J_f^+:hybrid contrastive	J_f:hybrid contrastive	9.51	0.00	5.93	13.08	True
J_f^+:hybrid contrastive	J_f:supervised	6.97	0.00	3.40	10.55	True
J_f^+:hybrid contrastive	J_f^+:behavior contrastive	11.29	0.00	7.71	14.87	True
J_f^+:hybrid contrastive	integrated-gradients:hybrid contrastive	-7.67	0.00	-11.75	-3.59	True
J_f^+:hybrid contrastive	feature-ablation:supervised	-7.23	0.00	-10.81	-3.66	True
J_f^+:hybrid contrastive	feature-ablation:hybrid contrastive	-9.00	0.00	-12.58	-5.42	True
J_f^+:hybrid contrastive	feature-ablation:behavior contrastive	-8.74	0.00	-12.32	-5.16	True
J_f^+:hybrid contrastive	J_f^+:supervised	-8.14	0.00	-11.72	-4.57	True
J_f^+:hybrid contrastive	shapely-zeros:hybrid contrastive	-10.68	0.00	-14.26	-7.10	True
J_f^+:hybrid contrastive	shapely-shuffle:hybrid contrastive	-9.71	0.00	-13.29	-6.13	True
J_f^+:hybrid contrastive	shapely-shuffle:supervised	-7.48	0.00	-11.06	-3.90	True
J_f^+:hybrid contrastive	shapely-zeros:behavior contrastive	-10.90	0.00	-14.47	-7.32	True
J_f^+:hybrid contrastive	shapely-zeros:supervised	-10.09	0.00	-13.66	-6.51	True
J_f^+:hybrid contrastive	integrated-gradients:behavior contrastive	-10.96	0.00	-14.54	-7.38	True
J_f^+:hybrid contrastive	integrated-gradients:supervised	-10.14	0.00	-13.72	-6.57	True
J_f^+:hybrid contrastive	shapely-shuffle:behavior contrastive	-9.13	0.00	-12.71	-5.55	True
J_f:supervised	J_f^+:behavior contrastive	-4.31	0.00	-7.89	-0.74	True
J_f:supervised	shapely-zeros:hybrid contrastive	-3.70	0.03	-7.28	-0.13	True
J_f:supervised	shapely-zeros:behavior contrastive	-3.92	0.02	-7.50	-0.35	True
J_f:supervised	integrated-gradients:behavior contrastive	-3.99	0.01	-7.56	-0.41	True
feature-ablation:supervised	J_f^+:behavior contrastive	4.05	0.01	0.48	7.63	True
feature-ablation:supervised	integrated-gradients:behavior contrastive	-3.73	0.03	-7.30	-0.15	True
feature-ablation:supervised	shapely-zeros:behavior contrastive	-3.66	0.04	-7.24	-0.09	True
shapely-shuffle:supervised	J_f^+:behavior contrastive	3.81	0.02	0.23	7.39	True

Table G.3: Posthoc test for the combination of attribution method and regularization scheme.

group1	group2	meandiff	p-adj	lower	upper	reject
J_f^+:REG	J_f:REG	3.16	0.00	0.58	5.75	True
J_f^+:REG	shapely-zeros:REG	-8.91	0.00	-11.50	-6.32	True
J_f^+:REG	J_f^+:none	-11.61	0.00	-14.20	-9.03	True
J_f^+:REG	feature-ablation:REG	-6.25	0.00	-8.84	-3.67	True
J_f^+:REG	feature-ablation:none	-9.06	0.00	-11.64	-6.47	True
J_f^+:REG	integrated-gradients:REG	-8.02	0.00	-10.70	-5.35	True
J_f^+:REG	integrated-gradients:none	-10.38	0.00	-13.06	-7.70	True
J_f^+:REG	shapely-shuffle:REG	-6.11	0.00	-8.69	-3.52	True
J_f^+:REG	shapely-shuffle:none	-10.10	0.00	-12.69	-7.51	True
J_f^+:REG	J_f:none	12.75	0.00	10.17	15.34	True
J_f^+:REG	shapely-zeros:none	-10.86	0.00	-13.44	-8.27	True
J_f:REG	shapely-zeros:REG	-5.75	0.00	-8.33	-3.16	True
J_f:REG	shapely-shuffle:REG	-2.94	0.01	-5.53	-0.35	True
J_f:REG	integrated-gradients:none	-7.21	0.00	-9.89	-4.53	True
J_f:REG	integrated-gradients:REG	-4.86	0.00	-7.53	-2.18	True
J_f:REG	feature-ablation:none	-5.89	0.00	-8.48	-3.30	True
J_f:REG	feature-ablation:REG	-3.09	0.01	-5.68	-0.50	True
J_f:REG	J_f^+:none	-8.45	0.00	-11.04	-5.86	True
J_f:REG	J_f:none	-9.59	0.00	-12.18	-7.00	True
J_f:REG	shapely-shuffle:none	-6.93	0.00	-9.52	-4.35	True
J_f:REG	shapely-zeros:none	-7.69	0.00	-10.28	-5.10	True
shapely-shuffle:REG	J_f:none	6.65	0.00	4.06	9.24	True
shapely-shuffle:REG	shapely-zeros:none	-4.75	0.00	-7.34	-2.16	True
shapely-shuffle:REG	shapely-zeros:REG	-2.81	0.02	-5.39	-0.22	True
shapely-shuffle:REG	J_f^+:none	5.51	0.00	2.92	8.10	True
shapely-shuffle:REG	integrated-gradients:none	4.27	0.00	1.59	6.95	True
shapely-shuffle:REG	feature-ablation:none	2.95	0.01	0.36	5.54	True
shapely-shuffle:REG	shapely-shuffle:none	-3.99	0.00	-6.58	-1.41	True
feature-ablation:REG	feature-ablation:none	-2.80	0.02	-5.39	-0.21	True
feature-ablation:REG	shapely-shuffle:none	-3.84	0.00	-6.43	-1.26	True
feature-ablation:REG	J_f:none	6.50	0.00	3.91	9.09	True
feature-ablation:REG	shapely-zeros:REG	-2.66	0.04	-5.24	-0.07	True
feature-ablation:REG	integrated-gradients:none	-4.12	0.00	-6.80	-1.44	True
feature-ablation:REG	J_f^+:none	5.36	0.00	2.77	7.95	True
feature-ablation:REG	shapely-zeros:none	-4.60	0.00	-7.19	-2.01	True
integrated-gradients:REG	J_f^+:none	3.59	0.00	0.92	6.27	True
integrated-gradients:REG	shapely-zeros:none	-2.83	0.03	-5.51	-0.16	True
integrated-gradients:REG	J_f:none	4.73	0.00	2.06	7.41	True
shapely-zeros:REG	J_f:none	3.84	0.00	1.26	6.43	True
shapely-zeros:REG	J_f^+:none	2.70	0.03	0.11	5.29	True
feature-ablation:none	J_f:none	3.70	0.00	1.11	6.29	True
shapely-shuffle:none	J_f:none	2.66	0.04	0.07	5.24	True

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283 (pp. 87, 212, 262).

Abi-Rached, J. M., & Rose, N. (2010). The birth of the neuromolecular gaze. *History of the human sciences*, *23*(1), 11–36 (p. 6).

Abnar, S., Berg, R. v. d., Ghiasi, G., Dehghani, M., Kalchbrenner, N., & Sedghi, H. (2021). Gradual domain adaptation in the wild: When intermediate distributions are absent. *arXiv preprint arXiv:2106.06080* (p. 78).

Ahrens, M. B., Orger, M. B., Robson, D. N., Li, J. M., & Keller, P. J. (2013). Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature methods*, *10*(5), 413–420 (p. 5).

Aimon, S., Katsuki, T., Jia, T., Grosenick, L., Broxton, M., Deisseroth, K., Sejnowski, T. J., & Greenspan, R. J. (2019). Fast near-whole–brain imaging in adult drosophila during responses to stimuli and behavior. *PLoS biology*, *17*(2), e2006732 (p. 5).

Akiyama, K., Alberdi, A., Alef, W., Asada, K., Azulay, R., Baczko, A.-K., Ball, D., Baloković, M., Barrett, J., Bintley, D., et al. (2019). First m87 event horizon telescope results. iv. imaging the central supermassive black hole. *The Astrophysical Journal Letters*, *875*(1), L4 (p. 3).

Ancona, M., Ceolini, E., Öztireli, C., & Gross, M. H. (2017). Towards better understanding of gradient-based attribution methods for deep neural networks. *International Conference on Learning Representations*. https://api.semanticscholar.org/CorpusID:3728967 (p. 176).

Anderson, T., & McIlwraith, C. (2004). Longitudinal development of equine conformation from weanling to age 3 years in the thoroughbred. *Equine veterinary journal*, *36*(7), 563–570 (p. 86).

Andriluka, M., Iqbal, U., Insafutdinov, E., Pishchulin, L., Milan, A., Gall, J., & Schiele, B. (2018). Posetrack: A benchmark for human pose estimation and tracking. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5167–5176 (pp. 83, 84).

Andriluka, M., Pishchulin, L., Gehler, P., & Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. *Proceedings of the IEEE Conference*

on computer Vision and Pattern Recognition, 3686–3693. http://ieeexplore.ieee.org/document/6909866/ (pp. 83, 84, 86).

Araki, T., Yoshida, F., Uemura, T., Noda, Y., Yoshimoto, S., Kaiju, T., Suzuki, T., Hamanaka, H., Baba, K., Hayakawa, H., et al. (2019). Long-term implantable, flexible, and transparent neural interface based on ag/au core–shell nanowires. *Advanced Healthcare Materials*, *8*(10), 1900130 (p. 9).

Azimi, F., Palacio, S., Raue, F., Hees, J., Bertinetto, L., & Dengel, A. (2022). Self-supervised test-time adaptation on video data. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3439–3448 (p. 41).

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7), e0130140 (p. 182).

Bachman, P., Hjelm, R. D., & Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, neurips 2019, december 8-14, 2019, vancouver, bc, canada* (pp. 15509–15519). (Pp. 98, 100, 101, 111).

Bachmann, D., Weichert, F., & Rinkenauer, G. (2015). Evaluation of the leap motion controller as a new contact-free pointing device. *Sensors*, *15*(1), 214–233 (p. 82).

Bachmann, R., Mizrahi, D., Atanov, A., & Zamir, A. (2022). Multimae: Multi-modal multi-task masked autoencoders. *European Conference on Computer Vision*, 348–367 (p. 200).

Baevski, A., Schneider, S., & Auli, M. (2020a). Vq-wav2vec: Self-supervised learning of discrete speech representations. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* (p. 98).

Baevski, A., Schneider, S., & Auli, M. (2020b). Vq-wav2vec: Self-supervised learning of discrete speech representations. *International Conference on Learning Representations* (p. 196).

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020c). Wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*. (Pp. 17, 98, 101).

Bai, S., Koltun, V., & Kolter, J. Z. (2020). Multiscale deep equilibrium models. *Advances in Neural Information Processing Systems*, *33*, 5238–5250 (p. 48).

Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., & Katz, B. (2019). Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in Neural Information Processing Systems 32* (p. 26).

Barlow, H. B., et al. (1961). Possible principles underlying the transformation of sensory messages. *Sensory communication*, *1*(01), 217–233 (p. 6).

Bartler, A., Bühler, A., Wiewel, F., Döbler, M., & Yang, B. (2022). Mt3: Meta test-time training for self-supervised test-time adaption. *International Conference on Artificial Intelligence and Statistics*, 3080–3090 (pp. xxiii, 40, 41, 48, 50, 51, 53, 54, 246, 247, 250).

Bassett, D. S., & Sporns, O. (2017). Network neuroscience. *Nature neuroscience*, *20*(3), 353. https://www.nature.com/articles/nn.4502 (p. 8).

Bastani, O., Kim, C., & Bastani, H. (2017). Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504* (p. 182).

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6541–6549 (p. 182).

Becker, R. A. (2012). The variance drain and jensen's inequality. *2012-004* (p. 223).

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828 (p. 16).

Berens, P., Freeman, J., Deneux, T., Chenkov, N., McColgan, T., Speiser, A., Macke, J. H., Turaga, S. C., Mineault, P. J., Rupprecht, P., Gerhard, S., Friedrich, R. W., Friedrich, J., Paninski, L., Pachitariu, M., Harris, K. D., Bolte, B., Machado, T. A., Ringach, D. L., … Bethge, M. (2018). Community-based benchmarking improves spike rate inference from two-photon calcium imaging data. *PLoS Computational Biology*, *14* (p. 131).

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *ArXiv preprint*, *abs/1912.06680*. https://arxiv.org/abs/1912.06680 (p. 38).

Berthelot, D., Roelofs, R., Sohn, K., Carlini, N., & Kurakin, A. (2021). Adamatch: A unified approach to semi-supervised learning and domain adaptation. (P. 42).

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag. (P. 25).

Blender Online Community. (2021). *Blender - a 3d modelling and rendering package*. Blender Foundation. Blender Institute, Amsterdam. (Pp. 109, 303).

Bobu, A., Tzeng, E., Hoffman, J., & Darrell, T. (2018). Adapting to continuously shifting domains. *Workshop Track - ICLR 2018* (p. 78).

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (pp. 5, 202).

Breen, R., Karlson, K. B., & Holm, A. (2018). Interpreting and understanding logits, probits, and other nonlinear probability models. *annual review of sociology*, *44*, 39–54 (p. 176).

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., …

Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (p. 38).

Bug, D., Schneider, S., Grote, A., Oswald, E., Feuerhake, F., Schüler, J., & Merhof, D. (2017). Context-based normalization of histological stains using deep convolutional features. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer. (P. 33).

Burgess, C., & Kim, H. (2018). 3d shapes dataset. (Pp. 303, 304).

Cai, T., Gao, R., Lee, J. D., & Lei, Q. (2021). A theory of label propagation for subpopulation shift. *arXiv preprint arXiv:2102.11203* (p. 42).

Całka, A. (1982). Local isometries of compact metric spaces. *Proceedings of the American Mathematical Society*, *85*(4), 643–647 (p. 299).

Cao, J., Tang, H., Fang, H.-S., Shen, X., Lu, C., & Tai, Y.-W. (2019). Cross-domain adaptation for animal pose estimation. *Proceedings of the IEEE International Conference on Computer Vision*, 9498–9507 (pp. 84, 86, 91).

Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. *CVPR* (p. 82).

Cariucci, F. M., Porzi, L., Caputo, B., Ricci, E., & Bulo, S. R. (2017). Autodial: Automatic domain alignment layers. *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 23, 24, 33).

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021a). Emerging properties in self-supervised vision transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9650–9660 (pp. 131, 152).

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021b). Emerging properties in self-supervised vision transformers. *ArXiv preprint*, *abs/2104.14294*. https://arxiv.org/abs/2104.14294 (pp. xxii, 17, 40, 43, 46, 51, 58, 242).

Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A., & Fiete, I. R. (2019). The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nature Neuroscience*, *22*, 1512–1520 (p. 124).

Chen, D., Wang, D., Darrell, T., & Ebrahimi, S. (2022). Contrastive test-time adaptation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 295–305 (pp. 42, 45).

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. *International conference on machine learning*, 1597–1607 (pp. 17, 41, 98–101, 108, 111, 196, 302, 303).

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. E. (2020b). A simple framework for contrastive learning of visual representations. *ArXiv*, *abs/2002.05709* (pp. 117, 140).

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., & Hinton, G. (2020c). Big self-supervised models are strong semi-supervised learners. *CoRR*, *abs/2006.10029* (pp. 26, 56, 108, 232).

Cheng, B., Xiao, B., Wang, J., Shi, H., Huang, T. S., & Zhang, L. (2020). Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5386–5395 (p. 82).

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., & Jitsev, J. (2023). Reproducible scaling laws for contrastive language-image learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2818–2829 (p. 195).

Chiang, C.-H., Won, S. M., Orsborn, A. L., Yu, K. J., Trumpis, M., Bent, B., Wang, C., Xue, Y., Min, S., Woods, V., et al. (2020). Development of a neural interface for high-definition, long-term recording in rodents and nonhuman primates. *Science translational medicine*, *12*(538), eaay4682 (p. 9).

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195 (pp. 46, 240).

Chowdhury, R. H., Glaser, J. I., & Miller, L. E. (2020). Area 2 of primary somatosensory cortex encodes kinematics of the whole arm. *ELife*, *9*, e48198 (pp. 119, 128, 129, 138).

Chuang, C., Robinson, J., Lin, Y., Torralba, A., & Jegelka, S. (2020). Debiased contrastive learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*. (P. 100).

Chung, J. E., Joo, H. R., Fan, J. L., Liu, D. F., Barnett, A. H., Chen, S., Geaghan-Breiner, C., Karlsson, M. P., Karlsson, M., Lee, K. Y., et al. (2019). High-density, long-lasting, and multi-region electrophysiological recordings using polymer electrode arrays. *Neuron*, *101*(1), 21–31 (p. 9).

Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Foster, J. D., Nuyujukian, P., Ryu, S. I., & Shenoy, K. V. (2012a). Neural population dynamics during reaching. *Nature*, *487*(7405), 51–56 (pp. 7, 13).

Churchland, M., Cunningham, J., Kaufman, M., Foster, J., Nuyujukian, P., Ryu, S., & Shenoy, K. V. (2012b). Neural population dynamics during reaching. *Nature*, *487*, 51–56 (p. 114).

Coates, A., Ng, A., & Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (p. 45).

Cohen, J. Y., Haesler, S., Vong, L., Lowell, B. B., & Uchida, N. (2012). Neuron-type specific signals for reward and punishment in the ventral tegmental area. *Nature*, *482*, 85–88 (p. 131).

Cong, L., Wang, Z., Chai, Y., Hang, W., Shang, C., Yang, W., Bai, L., Du, J., Wang, K., & Wen, Q. (2017). Rapid whole brain imaging of neural activity in freely behaving larval zebrafish (danio rerio). *Elife*, *6*, e28158 (pp. 5, 10).

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., & Hein, M. (2020). Robustbench: A standardized adversarial robustness benchmark. *ArXiv preprint*, *abs/2010.09670*. https://arxiv.org/abs/2010.09670 (pp. xxii, 242).

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., & Le, Q. V. (2019). Autoaugment: Learning augmentation policies from data. *Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 26).

Curto, C. (2016). What can topology tell us about the neural code. *arXiv: Neurons and Cognition* (p. 124).

Deitch, D., Rubin, A., & Ziv, Y. (2021). Representational drift in the mouse visual cortex. *Current Biology*, *31*(19), 4327–4339 (p. 138).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009a). Imagenet: A large-scale hierarchical image database. *Conference on computer vision and pattern recognition (CVPR)* (pp. 5, 23, 24, 26, 38).

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Li, F. (2009b). Imagenet: A large-scale hierarchical image database. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848 (p. 304).

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142 (p. 45).

Denk, W., & Svoboda, K. (1997). Photon upmanship: Why multiphoton imaging is more than a gimmick. *Neuron*, *18*(3), 351–357 (pp. 7, 8).

Denk, W., Strickler, J. H., & Webb, W. W. (1990). Two-photon laser scanning fluorescence microscopy. *Science*, *248*(4951), 73–76 (pp. 2, 7, 8).

de Silva, V., Morozov, D., & Vejdemo-Johansson, M. (2009). Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, *45*, 737–759 (p. 124).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423 (p. 195).

de Vries, S. E., Lecoq, J. A., Buice, M. A., Groblewski, P. A., Ocker, G. K., Oliver, M., Feng, D., Cain, N., Ledochowitsch, P., Millman, D., et al. (2020). A large-scale standardized physiological survey reveals functional organization of the mouse visual cortex. *Nature Neuroscience*, *23*(1), 138–151 (pp. 5, 14, 119, 131, 138, 151).

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (p. 137).

Dittadi, A., Träuble, F., Locatello, F., Wüthrich, M., Agrawal, V., Winther, O., Bauer, S., & Schölkopf, B. (2021). On the transfer of disentangled representations in realistic settings. *International Conference on Learning Representations (ICLR)* (pp. 303, 304).

Dodge, S. F., & Karam, L. J. (2017). A study and comparison of human and deep learning recognition performance under visual distortions. *International Conference on Computer Communications and Networks, ICCCN 2017* (p. 38).

Dombeck, D. A., Harvey, C. D., Tian, L., Looger, L. L., & Tank, D. W. (2010). Functional imaging of hippocampal place cells at cellular resolution during virtual navigation. *Nature neuroscience*, *13*, 1433–1440 (p. 124).

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). DeCaf: A deep convolutional activation feature for generic visual recognition. *International conference on machine learning*, 647–655. https://arxiv.org/abs/1310.1531 (p. 84).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of the 36th International Conference on Machine Learning* (pp. 40, 46, 74–76).

Durstewitz, D. (2017). A state space approach for piecewise-linear recurrent neural networks for identifying computational dynamics from neural measurements. *PLoS computational biology*, *13*(6), e1005542 (p. 13).

Eastwood, C., Mason, I., Williams, C. K., & Schölkopf, B. (2022). Source-free adaptation to measurement shift via bottom-up feature restoration. *International Conference on Learning Representations (ICLR)* (pp. 41, 50, 51, 55).

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Steinhardt, J., & Madry, A. (2020). Identifying statistical bias in dataset replication. *CoRR*, *abs/2005.09619* (p. 212).

Esfahany, K., Siergiej, I., Zhao, Y., & Park, I. M. (2018). Organization of neural population code in mouse visual system. *eNeuro*, *5* (p. 132).

Farahani, A., Voghoei, S., Rasheed, K., & Arabnia, H. R. (2021). A brief review of domain adaptation. *Advances in data science and information engineering*, 877–894 (p. 38).

Feng, F., Chan, R. H., Shi, X., Zhang, Y., & She, Q. (2019). Challenges in task incremental learning for assistive robotics. *IEEE Access*, *8*, 3434–3441 (p. 79).

Ferlauto, L., Vagni, P., Fanelli, A., Zollinger, E. G., Monsorno, K., Paolicelli, R. C., & Ghezzi, D. (2021). All-polymeric transient neural probe for prolonged in-vivo electrophysiological recordings. *Biomaterials*, *274*, 120889 (p. 9).

Ford, N., Gilmer, J., Carlini, N., & Cubuk, D. (2019). Adversarial examples are a natural consequence of test error in noise. *International Conference on Machine Learning (ICML)* (pp. 33, 211).

Frankle, J., Schwab, D. J., & Morcos, A. S. (2020). Training batchnorm and only batchnorm: On the expressive power of random features in cnns. *CoRR*, *abs/2003.00152* (p. 34).

French, G., Mackiewicz, M., & Fisher, M. H. (2017). Self-ensembling for domain adaptation. *CoRR*, *abs/1706.05208* (pp. 33, 43, 260, 262).

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, *36*(4), 193–202 (p. 5).

Gallego, J. A., Perich, M. G., Naufel, S., Ethier, C., Solla, S. A., & Miller, L. E. (2018). Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. *Nature Communications*, *9* (p. 114).

Galloway, A., Golubeva, A., Tanay, T., Moussa, M., & Taylor, G. W. (2019). Batch normalization is a cause of adversarial vulnerability. *CoRR*, *abs/1905.02161* (p. 30).

Galstyan, A., & Cohen, P. R. (2007). Empirical comparison of hard and soft label propagation for relational classification. *17th international conference on Inductive logic programming* (pp. 43, 44).

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, *17*(1), 2096–2030 (pp. 38, 45, 47–49, 78).

Gao, Y., Archer, E., Paninski, L., & Cunningham, J. P. (2016a). Linear dynamical neural population models through nonlinear embeddings. *NIPS* (p. 157).

Gao, Y., Archer, E. W., Paninski, L., & Cunningham, J. P. (2016b). Linear dynamical neural population models through nonlinear embeddings. *Advances in neural information processing systems*, *29* (p. 13).

Gardner, R. J., Hermansen, E., Pachitariu, M., Burak, Y., Baas, N. A., Dunn, B. A., Moser, M.-B., & Moser, E. I. (2022). Toroidal topology of population activity in grid cells. *Nature*, *602*(7895), 123–128. https://doi.org/10.1038/s41586-021-04268-7 (pp. 124, 131, 151).

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074 (p. 109).

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *CoRR*, *abs/2004.07780* (pp. 3, 34).

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2018a). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231* (p. 78).

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2019). Imagenet-trained CNNs are biased towards texture; increasing shape

bias improves accuracy and robustness. *International Conference on Learning Representations (ICLR)* (pp. 26, 33, 38, 232, 252).

Geirhos, R., Temme, C. R. M., Rauber, J., Schütt, H. H., Bethge, M., & Wichmann, F. A. (2018b). Generalisation in humans and deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31* (pp. 7538–7550). Curran Associates, Inc. http://papers.nips.cc/paper/7982-generalisation-in-humans-and-deep-neural-networks.pdf (pp. 22, 38).

Ghosh, A., Kumar, H., & Sastry, P. S. (2017). Robust loss functions under label noise for deep neural networks. In S. P. Singh & S. Markovitch (Eds.), *Proceedings of the thirty-first AAAI conference on artificial intelligence, february 4-9, 2017, san francisco, california, USA* (pp. 1919–1925). AAAI Press. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14759 (p. 44).

Ghosh, K. K., Burns, L. D., Cocker, E. D., Nimmerjahn, A., Ziv, Y., El Gamal, A., & Schnitzer, M. J. (2011). Miniaturized integration of a fluorescence microscope. *Nature methods*, *8*(10), 871 (pp. 2, 8).

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728* (p. 41).

Goh, W. W. B., Wang, W., & Wong, L. (2017). Why batch effects matter in omics data, and how to avoid them. *Trends in biotechnology*, *35*(6), 498–507 (pp. 16, 193).

Golub, M. D., Sadtler, P. T., Oby, E. R., Quick, K. M., Ryu, S. I., Tyler-Kabara, E. C., Batista, A. P., Chase, S. M., & Yu, B. M. (2018). Learning by neural reassociation. *Nature neuroscience*, *21*(4), 607–616 (p. 197).

Gondal, M. W., Wuthrich, M., Miladinovic, D., Locatello, F., Breidt, M., Volchkov, V., Akpo, J., Bachem, O., Schölkopf, B., & Bauer, S. (2019). On the transfer of inductive bias from simulation to the real world: A new disentanglement dataset. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, neurips 2019, december 8-14, 2019, vancouver, bc, canada* (pp. 15714–15725). (Pp. 303, 304).

Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., & Lee, S.-J. (2022). Note: Robust continual test-time adaptation against temporal correlation. *Advances in Neural Information Processing Systems* (pp. 67–69).

Goyal, S., Sun, M., Raghunathan, A., & Kolter, Z. (2022). Test-time adaptation via conjugate pseudo-labels. *arXiv preprint arXiv:2207.09640* (pp. 66, 67, 72, 74, 271).

Grandvalet, Y., & Bengio, Y. (2004). Semi-supervised learning by entropy minimization. *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, 529–536. https://proceedings.neurips.cc/paper/2004/hash/96f2b50b5d3613adf9c27049b2a888c7-Abstract.html (p. 44).

Grewe, B. F., Langer, D., Kasper, H., Kampa, B. M., & Helmchen, F. (2010). High-speed in vivo calcium imaging reveals neuronal network activity with near-millisecond precision. *Nature methods*, *7*(5), 399–405 (p. 2).

Grosmark, A. D., & Buzsáki, G. (2016). Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences. *Science*, *351*(6280), 1440–1443 (pp. 115, 119, 120, 138).

Guan, S., Wang, J., Gu, X., Zhao, Y., Hou, R., Fan, H., Zou, L., Gao, L., Du, M., Li, C., et al. (2019). Elastocapillary self-assembled neurotassels for stable neural activity recordings. *Science advances*, *5*(3), eaav2842 (p. 9).

Gulrajani, I., & Lopez-Paz, D. (2021). In search of lost domain generalization. *Proceedings of the 36th International Conference on Machine Learning* (pp. 43, 61).

Gutmann, M., & Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 297–304 (p. 16).

Gutmann, M. U., & Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, *13*, 307–361 (pp. 98–101, 117).

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., & Moser, E. I. (2005a). Microstructure of a spatial map in the entorhinal cortex. *Nature*, *436*(7052), 801–806 (p. 6).

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., & Moser, E. I. (2005b). Microstructure of a spatial map in the entorhinal cortex. *Nature*, *436*, 801–806 (p. 131).

Hälvä, H., Le Corff, S., Lehéricy, L., So, J., Zhu, Y., Gassiat, E., & Hyvärinen, A. (2021). Disentangling identifiable features from noisy data with structured nonlinear ica. *Advances in Neural Information Processing Systems*, *34*, 1624–1633 (p. 118).

Han, J., Liang, X., Xu, H., Chen, K., Lanqing, H., Mao, J., Ye, C., Zhang, W., Li, Z., Liang, X., et al. (2021). Soda10m: A large-scale 2d self/semi-supervised object detection dataset for autonomous driving. *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)* (p. 79).

Harmeling, S., Ziehe, A., Kawanabe, M., & Müller, K.-R. (2003). Kernel-based nonlinear blind source separation. *Neural Computation*, *15*(5), 1089–1124 (p. 101).

Harris, K. D., Quiroga, R. Q., Freeman, J., & Smith, S. L. (2016). Improving data quality in neuronal population recordings. *Nature neuroscience*, *19*(9), 1165–1174 (p. 8).

Hausmann, S. B., Vargas, A. M., Mathis, A., & Mathis, M. W. (2021). Measuring and modeling the motor system with machine learning. *Current opinion in neurobiology*, *70*, 11–23 (p. 11).

He, K., Chen, X., Xie, S., Li, Y., Doll'ar, P., & Girshick, R. B. (2021). Masked autoencoders are scalable vision learners. 2022 ieee. *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15979–15988 (p. 17).

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. B. (2020a). Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 9726–9735. https://doi.org/10.1109/CVPR42600.2020.00975 (pp. 98, 100, 101).

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. B. (2020b). Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 9726–9735. https://doi.org/10.1109/CVPR42600.2020.00975 (pp. 101, 111).

He, K., Girshick, R., & Dollár, P. (2018). Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883* (pp. 84, 88, 92, 194).

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, 2961–2969 (p. 82).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, *abs/1502.01852*. http://arxiv.org/abs/1502.01852 (p. 88).

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778. https://doi.org/10.1109/CVPR.2016.90 (pp. 38, 71, 74–76).

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778. https://doi.org/10.1109/CVPR.2016.90 (pp. 110, 302).

He, K., Zhang, X., Ren, S., & Sun, J. (2016c). Deep residual learning for image recognition. *Conference on computer vision and pattern recognition (CVPR)* (pp. xxii, 26, 45, 46, 231, 242).

He, K., Zhang, X., Ren, S., & Sun, J. (2016d). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778. https://arxiv.org/abs/1512.03385 (p. 87).

Hénaff, O. J. (2020). Data-efficient image recognition with contrastive predictive coding. *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, *119*, 4182–4192 (pp. 98, 108).

Hénaff, O. J., Srinivas, A., Fauw, J. D., Razavi, A., Doersch, C., Eslami, S. M. A., & van den Oord, A. (2020). Data-efficient image recognition with contrastive predictive coding. *International conference on machine learning*, 4182–4192 (p. 17).

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. (2020a). The many faces of robustness: A critical analysis of out-of-distribution generalization. *CoRR*, *abs/2006.16241* (pp. xxii, 16, 26, 38, 40, 45, 46, 48, 66, 75, 76, 232, 240–242, 252).

Hendrycks, D., & Dietterich, T. (2019a). Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)* (pp. 16, 22, 24, 26, 33, 41, 45, 67, 69, 212, 217, 218, 251).

Hendrycks, D., & Dietterich, T. (2019b). Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261* (pp. 71, 84–86, 270, 279).

Hendrycks, D., & Gimpel, K. (2016a). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (p. 145).

Hendrycks, D., & Gimpel, K. (2016b). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (p. 179).

Hendrycks, D., Lee, K., & Mazeika, M. (2019a). Using pre-training can improve model robustness and uncertainty. *ICML* (p. 85).

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2020b). Augmix: A simple data processing method to improve robustness and uncertainty. *International Conference on Learning Representations (ICLR)* (pp. 26, 33, 47–49, 66, 75, 76, 78, 232, 241, 252).

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., & Song, D. (2019b). Natural adversarial examples. *CoRR, abs/1907.07174* (pp. 26, 31, 45).

Hewitt, C., & Mahmoud, M. (2019). Pose-informed face alignment for extreme head pose variations in animals. *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, 1–6 (p. 84).

Hinton, G., Vinyals, O., & Dean, J. (2014). Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop* (pp. 46, 240).

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., & Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (pp. 98, 100, 101).

Hodgkin, A. L., & Huxley, A. F. (1952). Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *The Journal of physiology*, *116*(4), 449 (p. 6).

Hoffman, J., Darrell, T., & Saenko, K. (2014). Continuous manifold based adaptation for evolving visual domains. *Computer Vision and Pattern Recognition (CVPR)* (p. 78).

Hoffman, J., Roberts, D. A., & Yaida, S. (2019). Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729* (p. 178).

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., & Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. *International conference on machine learning*, 1989–1998 (p. 38).

Hong, G., & Lieber, C. M. (2019). Novel electrode technologies for neural recordings. *Nature Reviews Neuroscience*, *20*(6), 330–345 (p. 7).

Hong, J.-W., Yoon, C., Jo, K., Won, J. H., & Park, S. (2021). Recent advances in recording and modulation technologies for next-generation neural interfaces. *IScience*, *24*(12) (pp. 9, 361).

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. *Proceedings of the 36th International Conference on Machine Learning* (p. 52).

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2021). Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations* (p. 195).

Huang, G., Liu, Z., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. xxii, 26, 28, 46, 48, 231, 242, 258).

Hubel, D. H. (1957). Tungsten microelectrode for recording from single units. *Science*, *125*(3247), 549–550 (p. 7).

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, *148*(3), 574 (p. 6).

Hubel, D. H., & Wiesel, T. N. (1963). Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, *165*(3), 559 (p. 6).

Hubel, D. H., & Wiesel, T. N. (1977). Ferrier lecture - functional architecture of macaque monkey visual cortex. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *198*, 1–59 (p. 131).

Humphries, M. D. (2021). Strong and weak principles of neural dimension reduction. (P. 114).

Hurwitz, C., Kudryashova, N., Onken, A., & Hennig, M. H. (2021). Building population models for large-scale neural recordings: Opportunities and pitfalls. *Current opinion in neurobiology*, *70*, 64–73 (pp. 13, 197).

Huxter, J. R., Burgess, N., & O'Keefe, J. (2003). Independent rate and temporal coding in hippocampal pyramidal cells. *Nature*, *425*, 828–832 (p. 120).

Hyvarinen, A., & Morioka, H. (2016). Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in neural information processing systems*, *29* (pp. 18, 196).

Hyvarinen, A., & Morioka, H. (2017). Nonlinear ica of temporally dependent stationary sources. *Artificial Intelligence and Statistics*, 460–469 (pp. 18, 196).

Hyvarinen, A., Sasaki, H., & Turner, R. (2019). Nonlinear ica using auxiliary variables and generalized contrastive learning. *The 22nd International Conference on Artificial Intelligence and Statistics*, 859–868 (pp. 18, 176, 178, 184, 196).

Hyvärinen, A., Karhunen, J., & Oja, E. (2001). *Independent component analysis*. Wiley Interscience. (P. 100).

Hyvärinen, A., & Morioka, H. (2016). Unsupervised feature extraction by time-contrastive learning and nonlinear ICA. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29: Annual conference on neural information processing systems 2016, december 5-10, 2016, barcelona, spain* (pp. 3765–3773). (Pp. 100, 101, 106, 176, 179, 301, 302).

Hyvärinen, A., & Morioka, H. (2017). Nonlinear ICA of temporally dependent stationary sources. In A. Singh & X. (Zhu (Eds.), *Proceedings of the 20th international conference on artificial intelligence and statistics, AISTATS 2017, 20-22 april 2017, fort lauderdale, fl, USA* (pp. 460–469, Vol. 54). PMLR. (Pp. 100, 101, 106, 301, 302).

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural networks*, *13*(4-5), 411–430 (p. 199).

Hyvärinen, A., & Pajunen, P. (1999). Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, *12*(3), 429–439 (p. 101).

Hyvärinen, A., Sasaki, H., & Turner, R. E. (2019a). Nonlinear ICA using auxiliary variables and generalized contrastive learning. In K. Chaudhuri & M. Sugiyama (Eds.), *The 22nd international conference on artificial intelligence and statistics, AISTATS 2019, 16-18 april 2019, naha, okinawa, japan* (pp. 859–868, Vol. 89). PMLR. (Pp. 100, 101).

Hyvärinen, A., Sasaki, H., & Turner, R. E. (2019b). Nonlinear ICA using auxiliary variables and generalized contrastive learning. *The 22nd International Conference on Artificial Intelligence and Statistics*, *89*, 859–868. http://proceedings.mlr.press/v89/hyvarinen19a.html (pp. 117, 118, 140, 143, 153, 158, 163, 165, 169, 173, 174).

Ij, H. (2018). Statistics versus machine learning. *Nat Methods*, *15*(4), 233 (p. 14).

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., & Schmidt, L. (2021, July). *Openclip* (Version 0.1) [If you use this software, please cite it as below.]. Zenodo. https://doi.org/10.5281/zenodo.5143773 (p. 195).

Insafutdinov, E., Andriluka, M., Pishchulin, L., Tang, S., Levinkov, E., Andres, B., & Schiele, B. (2017). Arttrack: Articulated multi-person tracking in the wild. *CVPR'17*. http://arxiv.org/abs/1612.01465 (p. 82).

Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., & Schiele, B. (2016). DeeperCut: A deeper, stronger, and faster multi-person pose estimation model. *European Conference on Computer Vision*, 34–50. https://arxiv.org/abs/1605.03170 (p. 87).

International Brain Lab, Benson, B., Benson, J., Birman, D., Bonacchi, N., Carandini, M., Catarino, J. A., Chapuis, G. A., Churchland, A. K., Dan, Y., et al. (2023). A brain-wide map of neural activity during complex behaviour. *bioRxiv*, 2023–07 (pp. 5, 14).

International Brain Laboratory. (2023). Data release - Brainwide map - Q4 2022. https://doi.org/10.6084/m9.figshare.21400815.v6 (p. 5).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning (ICLR)* (pp. 23, 25, 41, 44).

Iwasawa, Y., & Matsuo, Y. (2021). Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, *34*, 2427–2440 (p. 42).

Jazayeri, M., & Ostojic, S. (2021). Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Current Opinion in Neurobiology*, *70*, 113–120 (pp. 114, 176).

Jin, M., & Glickfeld, L. L. (2020). Mouse higher visual areas provide both distributed and specialized contributions to visually guided behaviors. *Current Biology*, *30*, 4682–4692.e7 (p. 132).

Johnson, J., Douze, M., & Jégou, H. (2017a). Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734* (p. 110).

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., & Girshick, R. B. (2017b). CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 1988–1997. https://doi.org/10.1109/CVPR.2017.215 (pp. 99, 109, 303).

Jonas, E., & Kording, K. P. (2017). Could a neuroscientist understand a microprocessor? *PLoS computational biology*, *13*(1), e1005268 (p. 13).

Jones, E. G. (2000). Microcolumns in the cerebral cortex. *Proceedings of the National Academy of Sciences*, *97*(10), 5019–5021 (p. 202).

Jun, J. J., Steinmetz, N. A., Siegle, J. H., Denman, D. J., Bauza, M., Barbarits, B., Lee, A. K., Anastassiou, C. A., Andrei, A., Aydın, Ç., et al. (2017). Fully integrated silicon probes for high-density recording of neural activity. *Nature*, *551*(7679), 232–236 (pp. 2, 7, 9).

Jung, A. B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F.-M., Weng, C.-H., ... Laporte, M., et al. (2020). imgaug [Online; accessed 01-Feb-2020]. (P. 87).

Jutten, C., Babaie-Zadeh, M., & Karhunen, J. (2010). Nonlinear mixtures. *Handbook of Blind Source Separation, Independent Component Analysis and Applications*, 549–592 (p. 100).

Kamann, C., & Rother, C. (2019). Benchmarking the robustness of semantic segmentation models. *CoRR*, *abs/1908.05005* (p. 33).

Kandel, E. (1982). The origins of modern neuroscience. *Annual review of Neuroscience*, *5*(1), 299–303 (p. 6).

Kane, G., Lopes, G., Sanders, J., Mathis, A., & Mathis, M. (2020). Real-time, low-latency closed-loop feedback using markerless posture tracking. *eLife* (p. 14).

Kar, O. F., Yeo, T., Atanov, A., & Zamir, A. (2022). 3d common corruptions and data augmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18963–18974 (pp. 74, 270).

Keshtkaran, M. R., Sedler, A. R., Chowdhury, R. H., Tandon, R., Basrai, D., Nguyen, S. L., Sohn, H., Jazayeri, M., Miller, L. E., & Pandarinath, C. (2022). A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *Nature Methods* (pp. 119, 149, 150, 157).

Khan, M. H., McDonagh, J., Khan, S., Shahabuddin, M., Arora, A., Khan, F. S., Shao, L., & Tzimiropoulos, G. (2020). Animalweb: A large-scale hierarchical dataset of annotated animal faces. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6939–6948 (p. 84).

Khemakhem, I., Kingma, D., Monti, R., & Hyvarinen, A. (2020a, 26–28 Aug). Variational autoencoders and nonlinear ica: A unifying framework. In S. Chiappa & R. Calandra (Eds.), *Proceedings of the twenty third international conference on artificial intelligence and statistics* (pp. 2207–2217, Vol. 108). PMLR. https://proceedings.mlr.press/v108/khemakhem20a.html (p. 196).

Khemakhem, I., Kingma, D. P., Monti, R. P., & Hyvärinen, A. (2020b). Variational autoencoders and nonlinear ICA: A unifying framework. In S. Chiappa & R. Calandra (Eds.), *The 23rd international conference on artificial intelligence and statistics, AISTATS 2020, 26-28 august 2020, online [palermo, sicily, italy]* (pp. 2207–2217, Vol. 108). PMLR. (Pp. 100, 101).

Khemakhem, I., Monti, R. P., Kingma, D. P., & Hyvärinen, A. (2020c). Ice-beem: Identifiable conditional energy-based deep models based on nonlinear ICA. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*. (P. 101).

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., & Krishnan, D. (2020). Supervised contrastive learning. *arXiv preprint arXiv:2004.11362* (pp. 117, 118, 196).

Killcoyne, S., & Boyle, J. (2009). Managing chaos: Lessons learned developing software in the life sciences. *Computing in science & engineering*, *11*(6), 20–29 (p. 2).

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *International conference on machine learning*, 2668–2677 (p. 182).

Kim, D. H., Kim, J., Marques, J. C., Grama, A., Hildebrand, D. G., Gu, W., Li, J. M., & Robson, D. N. (2017). Pan-neuronal calcium imaging with cellular resolution in freely swimming zebrafish. *Nature methods*, *14*(11), 1107–1114 (p. 5).

Kim, Y., Cho, D., Han, K., Panda, P., & Hong, S. (2021). Domain adaptation without source data. *IEEE Transactions on Artificial Intelligence* (p. 39).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (pp. 72, 87, 273, 301, 302).

Klindt, D., Schott, L., Sharma, Y., Ustyuzhaninov, I., Brendel, W., Bethge, M., & Paiton, D. (2021a). Towards nonlinear disentanglement in natural data with temporal sparse coding. *International Conference on Learning Representations (ICLR)* (pp. 100, 108, 109, 302, 303).

Klindt, D. A., Schott, L., Sharma, Y., Ustyuzhaninov, I., Brendel, W., Bethge, M., & Paiton, D. (2021b). Towards nonlinear disentanglement in natural data with temporal sparse coding. *International Conference on Learning Representations*. https://openreview.net/forum?id=EbIDjBynYJ8 (p. 117).

Klioutchnikov, A., Wallace, D. J., Frosz, M. H., Zeltner, R., Sawinski, J., Pawlak, V., Voit, K.-M., Russell, P. S. J., & Kerr, J. N. (2020). Three-photon head-mounted microscope for imaging deep cortical layers in freely moving rats. *Nature methods*, *17*(5), 509–513 (p. 8).

Kobak, D., Brendel, W., Constantinidis, C., Feierstein, C. E., Kepecs, A., Mainen, Z. F., Qi, X.-L., Romo, R., Uchida, N., & Machens, C. K. (2016a). Demixed principal component analysis of neural population data. *elife*, *5*, e10989 (p. 13).

Kobak, D., Brendel, W., Constantinidis, C., Feierstein, C. E., Kepecs, A., Mainen, Z. F., Qi, X.-L., Romo, R., Uchida, N., & Machens, C. K. (2016b). Demixed principal component analysis of neural population data. *eLife*, *5* (p. 157).

Kobak, D., & Linderman, G. C. (2021). Initialization is critical for preserving global data structure in both t-sne and umap. *Nature Biotechnology*, *39*(2), 156–157. https://doi.org/10.1038/s41587-020-00809-z (p. 150).

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Beery, S., et al. (2020). Wilds: A benchmark of in-the-wild distribution shifts 2021. *arXiv preprint arXiv:2012.07421* (p. 193).

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., … Liang, P. (2021). WILDS: A benchmark of in-the-wild distribution shifts. *International Conference on Machine Learning (ICML)* (pp. 56, 258, 262).

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., et al. (2020). Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896* (p. 180).

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020). Big transfer (bit): General visual representation learning. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, 491–507 (pp. 53, 54, 259).

Kornblith, S., Norouzi, M., Lee, H., & Hinton, G. (2019a). Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414* (pp. 88, 92, 277).

Kornblith, S., Shlens, J., & Le, Q. V. (2019b). Do better imagenet models transfer better? *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2661–2671 (pp. 84, 87, 273).

Kozai, T. D. Y., Langhals, N. B., Patel, P. R., Deng, X., Zhang, H., Smith, K. L., Lahann, J., Kotov, N. A., & Kipke, D. R. (2012). Ultrasmall implantable composite microelectrodes with bioactive surfaces for chronic neural interfaces. *Nature materials*, *11*(12), 1065–1073 (p. 9).

Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A., & Poeppel, D. (2017a). Neuroscience Needs Behavior: Correcting a Reductionist Bias. *Neuron*, *93*(3), 480–490. http://www.cell.com/neuron/references/S0896-6273(16)31040-6 (pp. 9–11).

Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A., & Poeppel, D. (2017b). Neuroscience needs behavior: Correcting a reductionist bias. *Neuron*, *93*, 480–490 (p. 114).

Kreiss, S., Bertoni, L., & Alahi, A. (2019). Pifpaf: Composite fields for human pose estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11977–11986 (p. 82).

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images (pp. 41, 45, 67).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105). (Pp. 24, 231).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25: 26th annual conference on neural information processing systems 2012. proceedings of a meeting held december 3-6, 2012, lake tahoe, nevada, united states* (pp. 1106–1114). https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (p. 71).

Kumar, A., Ma, T., & Liang, P. (2020). Understanding self-training for gradual domain adaptation. *International Conference on Machine Learning*, 5468–5479 (p. 42).

Kümmerer, M., Wallis, T. S., & Bethge, M. (2016). Deepgaze ii: Reading fixations from deep features trained on object recognition. *arXiv preprint arXiv:1610.01563*. https://arxiv.org/abs/1610.01563 (p. 84).

Kundu, J. N., Venkat, N., Babu, R. V., et al. (2020). Universal source-free domain adaptation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4544–4553 (p. 39).

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., et al. (2018). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982* (p. 85).

Lamperti, J., et al. (1958). On the isometries of certain function-spaces. *Pacific J. Math*, *8*(3), 459–466 (p. 301).

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., & Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, *10*(1), 1–8 (p. 182).

Lauer, J., Zhou, M., Ye, S., Menegas, W., Schneider, S., Nath, T., Rahman, M. M., Di Santo, V., Soberanes, D., Feng, G., et al. (2022). Multi-animal pose estimation, identification and tracking with deeplabcut. *Nature Methods*, *19*(4), 496–504 (p. 20).

Lecoq, J. A., Boehringer, R., & Grewe, B. F. (2023). Deep brain imaging on the move. *Nature Methods*, *20*(4), 495–496 (pp. 2, 8).

LeCun, Y., Cortes, C., & Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, *2* (p. 67).

Lee, D.-H. (2013a). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *ICML Workshop : Challenges in Representation Learning (WREPL)* (pp. 43, 44, 78).

Lee, J. M. (2013b). Smooth manifolds. In *Introduction to smooth manifolds* (pp. 606–607). Springer. (P. 304).

Lee, J., Won, T., & Hong, K. (2020). Compounding the performance improvements of assembled techniques in a convolutional neural network. *CoRR*, *abs/2001.06268* (pp. 26, 28, 33).

Li, C.-K., & So, W. (1994). Isometries of ℓ_p-norm. *The American Mathematical Monthly*, *101*(5), 452–453 (p. 301).

Li, H., Singh, B., Najibi, M., Wu, Z., & Davis, L. S. (2019). An analysis of pre-training on object detection. *arXiv preprint arXiv:1904.05871* (pp. 84, 85).

Li, R., Jiao, Q., Cao, W., Wong, H.-S., & Wu, S. (2020a). Model adaptation: Unsupervised domain adaptation without source data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 39).

Li, S., Gunel, S., Ostrek, M., Ramdya, P., Fua, P., & Rhodin, H. (2020b). Deformation-aware unpaired image translation for pose estimation on laboratory animals. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13158–13168 (p. 84).

Li, Y., Wang, N., Shi, J., Liu, J., & Hou, X. (2017). Revisiting batch normalization for practical domain adaptation. *International Conference on Machine Learning (ICLR)* (pp. 23, 24, 33).

Liang, J., Hu, D., & Feng, J. (2020). Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. *International Conference on Machine Learning* (pp. 39, 45).

Liang, J., Hu, D., Wang, Y., He, R., & Feng, J. (2021). Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (p. 78).

Lin, J. (2020 (accessed October 21, 2020)). *A pytorch converter for simclr checkpoints* [Commit ID: 139d3cb0bd0c64b5ad32aab810e0bd0a0ddd aae0]. https://github.com/tonylins/simclr-converter (p. 215).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *European conference on computer vision*, 740–755 (pp. 83, 84).

Linderman, S., Johnson, M., Miller, A., Adams, R., Blei, D., & Paninski, L. (2017). Bayesian learning and inference in recurrent switching linear dynamical systems. *Artificial Intelligence and Statistics*, 914–922 (p. 13).

Linsker, R. (1988). Self-organization in a perceptual network. *Computer*, *21*(3), 105–117 (pp. 100, 101).

Liu, Y., Kothari, P., van Delft, B. G., Bellot-Gurlet, B., Mordan, T., & Alahi, A. (2021). Ttt++: When does self-supervised test-time training fail or thrive? *Thirty-Fifth Conference on Neural Information Processing Systems* (pp. 41, 50, 51).

Liu, Y., Li, J., Song, S., Kang, J., Tsao, Y., Chen, S., Mottini, V., McConnell, K., Xu, W., Zheng, Y.-Q., et al. (2020a). Morphing electronics enable neuromodulation in growing tissue. *Nature biotechnology*, *38*(9), 1031–1036 (p. 9).

Liu, Y., Liu, J., Chen, S., Lei, T., Kim, Y., Niu, S., Wang, H., Wang, X., Foudeh, A. M., Tok, J. B.-H., et al. (2019). Soft and elastic hydrogel-based microelectronics for

localized low-voltage neuromodulation. *Nature biomedical engineering*, *3*(1), 58–68 (p. 9).

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., & Guo, B. (2022). Swin transformer v2: Scaling up capacity and resolution. *International Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 75, 76).

Liu, Z., Miao, Z., Pan, X., Zhan, X., Lin, D., Yu, S. X., & Gong, B. (2020b). Open compound domain adaptation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12406–12415 (p. 78).

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., & Bachem, O. (2019a, September). Challenging common assumptions in the unsupervised learning of disentangled representations. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 4114–4124, Vol. 97). PMLR. https://proceedings.mlr.press/v97/locatello19a.html (p. 197).

Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., & Bachem, O. (2019b). Challenging common assumptions in the unsupervised learning of disentangled representations. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA* (pp. 4114–4124, Vol. 97). PMLR. (P. 110).

Locatello, F., Poole, B., Rätsch, G., Schölkopf, B., Bachem, O., & Tschannen, M. (2020). Weakly-supervised disentanglement without compromises. *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, *119*, 6348–6359 (p. 101).

Logeswaran, L., & Lee, H. (2018). An efficient framework for learning sentence representations. *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (pp. 98, 101).

Lomonaco, V., & Maltoni, D. (2017, 13–15 Nov). Core50: A new dataset and benchmark for continuous object recognition. In S. Levine, V. Vanhoucke, & K. Goldberg (Eds.), *Proceedings of the 1st annual conference on robot learning* (pp. 17–26, Vol. 78). PMLR. https://proceedings.mlr.press/v78/lomonaco17a.html (p. 79).

London, B. M., & Miller, L. E. (2013). Responses of somatosensory area 2 neurons to actively and passively generated limb movements. *Journal of neurophysiology*, *109 6*, 1505–13 (p. 128).

Luan, L., Wei, X., Zhao, Z., Siegel, J. J., Potnis, O., Tuppen, C. A., Lin, S., Kazmi, S., Fowler, R. A., Holloway, S., et al. (2017). Ultraflexible nanoelectronic probes form reliable, glial scar–free neural integration. *Science advances*, *3*(2), e1601966 (p. 9).

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30* (pp. 176, 180, 182).

Ma, N., Zhang, X., Zheng, H.-T., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 26, 231).

Maceira-Elvira, P., Popa, T., Schmid, A.-C., & Hummel, F. C. (2019). Wearable technology in stroke rehabilitation: Towards improved diagnosis and treatment of upper-limb motor impairment. *Journal of neuroengineering and rehabilitation*, *16*(1), 142 (pp. 82, 83).

Macke, J. H., Buesing, L., Cunningham, J. P., Yu, B. M., Shenoy, K. V., & Sahani, M. (2011). Empirical models of spiking in neural populations. *Advances in neural information processing systems*, *24* (p. 13).

Magland, J., Jun, J. J., Lovero, E., Morley, A. J., Hurwitz, C. L., Buccino, A. P., Garcia, S., & Barnett, A. H. (2020). Spikeforest, reproducible web-facing ground-truth validation of automated neural spike sorters. *Elife*, *9*, e55167 (p. 15).

Magnusson, L.-E., & Thafvellin, B. (1990). Studies on the conformation and related traits of standardbred trotters in sweden. *Journal of Animal Physiology and Animal Nutrition (Germany, FR)* (p. 86).

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., & van der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. xxii, 26, 29, 34, 38, 46, 50, 66, 85, 194, 232, 240, 242).

Mankiewicz, P. (1972). Extension of isometries in normed linear spaces. *Bulletin de l'Academie polonaise des sciences: Serie des sciences mathematiques, astronomiques et physiques*, *20*(5), 367–+ (p. 299).

Mann, K., Gallen, C. L., & Clandinin, T. R. (2017). Whole-brain calcium imaging reveals an intrinsic functional network in drosophila. *Current Biology*, *27*(15), 2389–2396 (p. 5).

Marcel, S., & Rodriguez, Y. (2010). Torchvision the machine-vision package of torch. *ACM International Conference on Multimedia* (pp. 26, 212, 241, 262).

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., & Bethge, M. (2018). Deeplabcut: Markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, *21*(9), 1281–1289 (pp. 2, 10, 84, 87, 287).

Mathis, A., Schneider, S., Lauer, J., & Mathis, M. W. (2020). A primer on motion capture with deep learning: Principles, pitfalls, and perspectives. *Neuron*, *108*(1), 44–65 (pp. 2, 10, 20).

Mathis, A., Stemmler, M. B., & Herz, A. V. (2015). Probable nature of higher-dimensional symmetries underlying mammalian grid-cell activity patterns. *Elife*, *4*, e05979 (p. 7).

Mathis, A., & Warren, R. A. (2018). On the inference speed and video-compression robustness of deeplabcut. *BioRxiv*. https://doi.org/10.1101/457242 (p. 287).

Mathis, M. W., & Schneider, S. (2021). Motor control: Neural correlates of optimal feedback control theory. *Current Biology*, *31*(7), R356–R358 (p. 7).

Mathis, M. W. (2023). The neocortical column as a universal template for perception and world-model learning. *Nature Reviews Neuroscience*, *24*(1), 3–3 (p. 202).

Mathis, M. W., & Mathis, A. (2020). Deep learning tools for the measurement of animal behavior in neuroscience. *Current Opinion in Neurobiology*, *60*, 1–11 (pp. 11, 12, 82, 83).

Maynard, E. M., Nordhausen, C. T., & Normann, R. A. (1997). The utah intracortical electrode array: A recording structure for potential brain-computer interfaces. *Electroencephalography and clinical neurophysiology*, *102*(3), 228–239 (pp. 2, 7).

Mazzocchi, F. (2015). Could big data be the end of theory in science? a few remarks on the epistemology of data-driven science. *EMBO reports*, *16*(10), 1250–1255 (p. 192).

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, *27*(4), 12–12 (p. 4).

McCullagh, P., & Nelder, J. A. (1972). Generalized linear models. *Predictive Analytics*. https://api.semanticscholar.org/CorpusID:14154576 (p. 176).

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (pp. 117, 149, 157, 199).

Menegas, W., Bergan, J. F., Ogawa, S. K., Isogai, Y., Venkataraju, K. U., Osten, P., Uchida, N., & Watabe-Uchida, M. (2015). Dopamine neurons projecting to the posterior striatum form an anatomically distinct subclass. *eLife*, *4* (p. 131).

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, *2014*(239) (pp. 212, 262).

Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M., & Brendel, W. (2019a). Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484* (pp. 84, 85, 87, 279).

Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M., & Brendel, W. (2019b). Benchmarking robustness in object detection: Autonomous driving when winter is coming. *CoRR*, *abs/1907.07484* (p. 33).

Mikołajczyk, A., & Grochowski, M. (2018). Data augmentation for improving deep learning in image classification problem. *International Interdisciplinary PhD Workshop (IIPhDW)* (p. 33).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26* (pp. 17, 196).

Miller, J. P., Taori, R., Raghunathan, A., Sagawa, S., Koh, P. W., Shankar, V., Liang, P., Carmon, Y., & Schmidt, L. (2021). Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization. *International Conference on Machine Learning*, 7721–7735 (p. 49).

Minev, I. R., Musienko, P., Hirsch, A., Barraud, Q., Wenger, N., Moraud, E. M., Gandar, J., Capogrosso, M., Milekovic, T., Asboth, L., et al. (2015). Electronic dura mater for long-term multimodal neural interfaces. *Science*, *347*(6218), 159–163 (p. 9).

Misra, I., & van der Maaten, L. (2019). Self-supervised learning of pretext-invariant representations. *CoRR*, *abs/1912.01991* (p. 17).

Molnar, C. (2022). *Interpretable machine learning: A guide for making black box models explainable* (2nd ed.). https://christophm.github.io/interpretable-ml-book (p. 180).

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., & Müller, K.-R. (2015). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognit.*, *65*, 211–222. https://api.semanticscholar.org/CorpusID:5731985 (p. 176).

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., & Müller, K.-R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern recognition*, *65*, 211–222 (p. 182).

Moore, G. P., Perkel, D. H., & Segundo, J. P. (1966). Statistical analysis and functional interpretation of neuronal spike data. *Annual review of physiology*, *28*(1), 493–522 (p. 6).

Morioka, H., & Hyvarinen, A. (2023). Connectivity-contrastive learning: Combining causal discovery and representation learning for multimodal data. *International Conference on Artificial Intelligence and Statistics*, 3399–3426 (p. 183).

Moser, E. I., Kropff, E., & Moser, M.-B. (2008). Place cells, grid cells, and the brain's spatial representation system. *Annual review of neuroscience*, *31*, 69–89 (p. 120).

Motiian, S., Jones, Q., Iranmanesh, S., & Doretto, G. (2017). Few-shot adversarial domain adaptation. *Advances in neural information processing systems*, *30* (p. 78).

Mountcastle, V. B. (1957). Modality and topographic properties of single neurons of cat's somatic sensory cortex. *Journal of neurophysiology*, *20*(4), 408–434 (p. 202).

Mu, J., Qiu, W., Hager, G. D., & Yuille, A. L. (2020). Learning from synthetic animals. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12386–12395 (p. 84).

Mu, N., & Gilmer, J. (2019). MNIST-C: A robustness benchmark for computer vision. *CoRR*, *abs/1906.02337* (p. 33).

Mudrakarta, P. K., Taly, A., Sundararajan, M., & Dhamdhere, K. (2018). Did the model understand the question? (P. 179).

Mummadi, C. K., Hutmacher, R., Rambach, K., Levinkov, E., Brox, T., & Metzen, J. H. (2021). Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999* (pp. 40, 41, 45, 48, 50, 51, 55, 66, 67, 72, 74, 78, 79).

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., & Snoek, J. (2020). Evaluating prediction-time batch normalization for robustness under covariate shift. *CoRR*, *abs/2006.10963* (pp. 34, 40, 66, 67, 72, 74, 78, 271).

Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. *Twenty-Ninth AAAI Conference on Artificial Intelligence* (p. 55).

Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. (2019). Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature Protocols*, *14*, 2152–2176 (pp. 2, 86, 87).

Neher, E., & Sakmann, B. (1976). Single-channel currents recorded from membrane of denervated frog muscle fibres. *Nature*, *260*(5554), 799–802 (p. 8).

Newell, M. E. (1975). *The utilization of procedure models in digital image synthesis.* [Doctoral dissertation, The University of Utah] [AAI7529894]. The University of Utah. (P. 303).

Nguyen, J. P., Shipley, F. B., Linder, A. N., Plummer, G. S., Liu, M., Setru, S. U., Shaevitz, J. W., & Leifer, A. M. (2016). Whole-brain calcium imaging with cellular resolution in freely behaving caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, *113*(8), E1074–E1081 (pp. 5, 10).

Niell, C. M., Stryker, M. P., & Keck, W. M. (2008). Highly selective receptive fields in mouse visual cortex. *The Journal of Neuroscience*, *28*, 7520–7536 (p. 131).

Ning, G., Pei, J., & Huang, H. (2020). Lighttrack: A generic framework for online top-down human pose tracking. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 1034–1035 (p. 83).

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., & Tan, M. (2022a). Efficient test-time model adaptation without forgetting. *Proceedings of the 39th International Conference on Machine Learning* (pp. 41, 45, 50, 55, 56, 62).

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., & Tan, M. (2022b). Efficient test-time model adaptation without forgetting. *arXiv preprint arXiv:2204.02610* (pp. 66–74, 76, 78, 193, 269, 271).

Nordhausen, C. T., Maynard, E. M., & Normann, R. A. (1996). Single unit recording capabilities of a 100 microelectrode array. *Brain research*, *726*(1-2), 129–140 (p. 7).

Oby, E. R., Golub, M. D., Hennig, J. A., Degenhart, A. D., Tyler-Kabara, E. C., Yu, B. M., Chase, S. M., & Batista, A. P. (2019). New neural activity patterns emerge with long-term learning. *Proceedings of the National Academy of Sciences*, *116*(30), 15210–15215 (p. 197).

O'Keefe, J., & Dostrovsky, J. (1971). The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain research* (p. 6).

Okorokova, E. V., Goodman, J. M., Hatsopoulos, N. G., & Bensmaia, S. J. (2020). Decoding hand kinematics from population responses in sensorimotor cortex during grasping. *Journal of neural engineering* (p. 114).

Olshausen, B. A., & Field, D. J. (1996a). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*(6583), 607–609 (p. 6).

Olshausen, B. A., & Field, D. J. (1996b). Natural image statistics and efficient coding. *Network: computation in neural systems*, *7*(2), 333 (p. 6).

Oord, A. v. d., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (pp. 17, 98–101, 117, 159).

Orhan, A. E. (2019). Robustness properties of facebook's resnext wsl models. *CoRR*, *abs/1907.07640* (p. 29).

Ostrek, M., Rhodin, H., Fua, P., Müller, E., & Spörri, J. (2019). Are existing monocular computer vision-based 3d motion capture approaches ready for deployment? a methodological study on the example of alpine skiing. *Sensors*, *19*(19), 4323 (pp. 82, 83).

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., Trautmann, E. M., Kaufman, M. T., Ryu, S. I., Hochberg, L. R., et al. (2018a). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, *15*(10), 805–815 (p. 13).

Pandarinath, C., O'Shea, D. J., Collins, J., Józefowicz, R., Stavisky, S. D., Kao, J. C., Trautmann, E. M., Kaufman, M. T., Ryu, S. I., Hochberg, L. R., Henderson, J. M., Shenoy, K. V., Abbott, L. F., & Sussillo, D. (2018b). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, *15*, 805–815 (pp. 117, 118, 157).

Park, S., Guo, Y., Jia, X., Choe, H. K., Grena, B., Kang, J., Park, J., Lu, C., Canales, A., Chen, R., et al. (2017). One-step optogenetics with multifunctional flexible polymer fibers. *Nature neuroscience*, *20*(4), 612–619 (p. 9).

Park, S., Yuk, H., Zhao, R., Yim, Y. S., Woldeghebriel, E. W., Kang, J., Canales, A., Fink, Y., Choi, G. B., Zhao, X., et al. (2021). Adaptive and multifunctional hydrogel hybrid probes for long-term sensing and modulation of neural activity. *Nature communications*, *12*(1), 3435 (p. 9).

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. *NIPS Autodiff Workshop* (pp. 25, 212, 262).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019a). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32* (p. 271).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019b). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (p. 145).

Patel, P. R., Welle, E. J., Letner, J. G., Shen, H., Bullard, A. J., Caldwell, C. M., Vega-Medina, A., Richie, J. M., Thayer, H. E., Patil, P. G., et al. (2023). Utah array characterization and histological analysis of a multi-year implant in non-human

primate motor and sensory cortices. *Journal of Neural Engineering*, *20*(1), 014001 (p. 7).

Pearl, J. (2009). *Causality*. Cambridge university press. (P. 183).

Pearson, K. (1901). Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *6*(2), 559 (pp. 13, 198).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830 (pp. 145, 150, 151).

Pei, F., Ye, J., Zoltowski, D., Wu, A., Chowdhury, R. H., Sohn, H., O'Doherty, J. E., Shenoy, K. V., Kaufman, M. T., Churchland, M., et al. (2021a). Neural latents benchmark'21: Evaluating latent variable models of neural population activity. *arXiv preprint arXiv:2109.04463* (pp. 119, 138, 157).

Pei, F., Ye, J., Zoltowski, D., Wu, A., Chowdhury, R. H., Sohn, H., O'Doherty, J. E., Shenoy, K. V., Kaufman, M. T., Churchland, M., et al. (2021b). Neural latents benchmark'21: Evaluating latent variable models of neural population activity. *arXiv preprint arXiv:2109.04463* (p. 197).

Pei, F., Ye, J., Zoltowski, D., Wu, A., Chowdhury, R. H., Sohn, H., O'Doherty, J. E., Shenoy, K. V., Kaufman, M. T., Churchland, M., Jazayeri, M., Miller, L. E., Pillow, J., Park, I. M., Dyer, E. L., & Pandarinath, C. (2021c). Neural latents benchmark '21: Evaluating latent variable models of neural population activity. https://doi.org/10.48550/ARXIV.2109.04463 (pp. 130, 151).

Peters, J., Janzing, D., & Schölkopf, B. (2017). *Elements of causal inference: Foundations and learning algorithms*. The MIT Press. (P. 183).

Petreska, B., Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S., Shenoy, K. V., & Sahani, M. (2011). Dynamical segmentation of single trials from population neural data. *Advances in neural information processing systems*, *24* (p. 13).

Poličar, P. G., Stražar, M., & Zupan, B. (2019). Opentsne: A modular python library for t-sne dimensionality reduction and embedding. *bioRxiv*. https://doi.org/10.1101/731877 (p. 150).

Prabhu, A., Torr, P. H., & Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, 524–540 (p. 68).

Prabhu, V., Khare, S., Kartik, D., & Hoffman, J. (2021). Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8558–8567 (p. 42).

Prince, L. Y., Bakhtiari, S., Gillon, C. J., & Richards, B. A. (2021). Parallel inference of hierarchical latent dynamics in two-photon calcium imaging of neuronal populations. *bioRxiv* (p. 117).

Prud'homme, M. J., & Kalaska, J. F. (1994). Proprioceptive activity in primate primary somatosensory cortex during active arm reaching movements. *Journal of neurophysiology*, *72 5*, 2280–301 (p. 128).

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *International Conference on Machine Learning*, 8748–8763 (pp. 18, 66, 196, 251).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9 (pp. 17, 194, 195, 200).

Raghu, M., Zhang, C., Kleinberg, J., & Bengio, S. (2019). Transfusion: Understanding transfer learning for medical imaging. *Advances in Neural Information Processing Systems*, 3342–3352 (pp. 84, 92).

Rashid, M., Gu, X., & Jae Lee, Y. (2017). Interspecies knowledge transfer for facial keypoint detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6894–6903 (p. 84).

Ratcliff, R. (1990). Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological review*, *97*(2), 285 (p. 78).

Ravanelli, M., Zhong, J., Pascual, S., Swietojanski, P., Monteiro, J., Trmal, J., & Bengio, Y. (2020). Multi-task self-supervised learning for robust speech recognition. *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, 6989–6993. https://doi.org/10.1109/ICASSP40776.2020.9053569 (p. 98).

Rebuffi, S.-A., Bilen, H., & Vedaldi, A. (2017). Learning multiple visual domains with residual adapters. *Advances in Neural Information Processing Systems (NIPS)* (p. 33).

Recce, M. (1989). The tetrode: A new technique for multi-unit extracellular recording. *Soc. Neurosci. Abstr.*, *15*, 1250 (p. 7).

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2020). Do imagenet classifiers generalize to imagenet? *Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 26, 251).

Reizinger, P., Sharma, Y., Bethge, M., Schölkopf, B., Huszár, F., & Brendel, W. (2022). Jacobian-based causal discovery with nonlinear ica. *Transactions on Machine Learning Research* (p. 183).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144 (p. 182).

Ringach, D. L., Mineault, P. J., Tring, E., Olivas, N. D., García-Junco-Clemente, P., & Trachtenberg, J. T. (2016). Spatial clustering of tuning in mouse primary visual cortex. *Nature Communications*, *7* (p. 131).

Robinson, J., Chuang, C.-Y., Sra, S., & Jegelka, S. (2020). Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592* (p. 100).

Roeder, G., Metz, L., & Kingma, D. P. (2020a). On linear identifiability of learned representations. *arXiv*. https://doi.org/10.48550/ARXIV.2007.00810 (pp. 117, 143, 153, 154, 158, 163, 165–168).

Roeder, G., Metz, L., & Kingma, D. P. (2020b). On linear identifiability of learned representations. *arXiv preprint arXiv:2007.00810* (p. 101).

Roeder, G., Metz, L., & Kingma, D. (2021). On linear identifiability of learned representations. *International Conference on Machine Learning*, 9030–9039 (p. 195).

Roweis, S., & Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural computation*, *11*(2), 305–345 (pp. 13, 198).

Rupprecht, P., Carta, S., Hoffmann, A., Echizen, M., Blot, A., Kwan, A. C., Dan, Y., Hofer, S. B., Kitamura, K., Helmchen, F., et al. (2021). A database and deep learning toolbox for noise-optimized, generalized spike inference from calcium imaging. *Nature neuroscience*, *24*(9), 1324–1337 (p. 8).

Rusak, E., Schneider, S., Pachitariu, G., Eck, L., Gehler, P. V., Bringmann, O., Brendel, W., & Bethge, M. (2021). If your data distribution shifts, use self-learning. *Transactions of Machine Learning Research* (pp. 66, 67, 72, 74, 78, 79, 271).

Rusak, E., Schott, L., Zimmermann, R., Bitterwolf, J., Bringmann, O., Bethge, M., & Brendel, W. (2020). Increasing the robustness of dnns against image corruptions by playing the game of noise. *CoRR*, *abs/2001.06057* (pp. 26, 33, 38, 61, 66, 78, 217, 232, 252).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision (IJCV)* (pp. 23, 26, 66).

Rutten, V., Bernacchia, A., Sahani, M., & Hennequin, G. (2020). Non-reversible gaussian processes for identifying latent dynamical structure in neural data. *Advances in neural information processing systems*, *33*, 9622–9632 (p. 13).

Ruzhansky, M., & Sugimoto, M. (2015). On global inversion of homogeneous maps. *Bulletin of Mathematical Sciences*, *5*(1), 13–18 (pp. 304, 305).

Sadtler, P. T., Quick, K. M., Golub, M. D., Chase, S. M., Ryu, S. I., Tyler-Kabara, E. C., Yu, B. M., & Batista, A. P. (2014). Neural constraints on learning. *Nature*, *512*(7515), 423–426 (p. 197).

Saenko, K., Peng, X., Usman, B., Saito, K., & Hu, P. (2019). *Visual domain adaptation challenge (visda-2019)*. http://ai.bu.edu/visda-2019/ (p. 59).

Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., & Müller, K.-R. (2019). *Explainable ai: Interpreting, explaining and visualizing deep learning* (Vol. 11700). Springer Nature. (Pp. 176, 180, 181).

Sanakoyeu, A., Khalidov, V., McCarthy, M. S., Vedaldi, A., & Neverova, N. (2020). Transferring dense pose to proximal animal classes. *arXiv preprint arXiv:2003.00080* (pp. 83, 84).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018a). Mobilenetv2: Inverted residuals and linear bottlenecks. *Conference on computer vision and pattern recognition (CVPR)* (pp. 26, 48, 231).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018b). Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520 (p. 87).

Sani, O. G., Abbaspourazad, H., Wong, Y., Pesaran, B., & Shanechi, M. (2020). Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nature Neuroscience*, *24*, 140–149 (p. 117).

Sani, O. G., Abbaspourazad, H., Wong, Y. T., Pesaran, B., & Shanechi, M. M. (2021). Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nature Neuroscience*, *24*(1), 140–149 (p. 13).

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? *Advances in Neural Information Processing Systems (NIPS)* (p. 25).

Saunshi, N., Plevrakis, O., Arora, S., Khodak, M., & Khandeparkar, H. (2019). A theoretical analysis of contrastive unsupervised representation learning. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA* (pp. 5628–5637, Vol. 97). PMLR. (Pp. 98, 100).

Sawinski, J., Wallace, D. J., Greenberg, D. S., Grossmann, S., Denk, W., & Kerr, J. N. (2009). Visually evoked activity in cortical cells imaged in freely moving animals. *Proceedings of the National Academy of Sciences*, *106*(46), 19557–19562 (p. 8).

Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). Wav2vec: Unsupervised pre-training for speech recognition. *CoRR*, *abs/1904.05862* (pp. 17, 98, 196).

Schneider, S., Ecker, A. S., Macke, J. H., & Bethge, M. (2018). Multi-task generalization and adaptation between noisy digit datasets: An empirical study. *Neural Information Processing Systems (NeurIPS), Workshop on Continual Learning* (pp. 23, 33).

Schneider, S., Lee, J. H., & Mathis, M. W. (2023). Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, *617*, 360–368 (pp. 178, 179, 184, 185, 187, 361).

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., & Bethge, M. (2020a). Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems* (pp. 40, 41, 46, 49–51, 53, 55, 66, 67, 72, 74, 78, 240, 247, 271).

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., & Bethge, M. (2020b). Removing covariate shift improves robustness against common corruptions. *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)* (pp. 85, 93, 279, 280).

Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., & Mooij, J. (2012). On causal and anticausal learning. *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 459–466 (pp. 25, 196).

Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, *31*, 1–10 (p. 151).

Schrimpf, M., Kubilius, J., Hong, H., Majaj, N. J., Rajalingham, R., Issa, E. B., Kar, K., Bashivan, P., Prescott-Roy, J., Geiger, F., et al. (2018). Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv*, 407007 (p. 197).

Schrödel, T., Prevedel, R., Aumayr, K., Zimmer, M., & Vaziri, A. (2013). Brain-wide 3d imaging of neuronal activity in caenorhabditis elegans with sculpted light. *Nature methods*, *10*(10), 1013–1020 (p. 5).

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C. W., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S. R., Crowson, K., Schmidt, L., Kaczmarczyk, R., & Jitsev, J. (2022). LAION-5b: An open large-scale dataset for training next generation image-text models. *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=M3Y74vmsMcY (p. 195).

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*, 1593–1599 (p. 131).

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE international conference on computer vision*, 618–626 (p. 182).

Shah, S., Sharma, A., Jain, A., et al. (2019). On the robustness of human pose estimation. *arXiv preprint arXiv:1908.06401* (p. 86).

Shapley, L. S., et al. (1953). A value for n-person games (p. 180).

She, Q., & Wu, A. (2020). Neural dynamics discovery via gaussian process recurrent neural networks. *Uncertainty in Artificial Intelligence*, 454–464 (p. 13).

Shi, X., Li, D., Zhao, P., Tian, Q., Tian, Y., Long, Q., Zhu, C., Song, J., Qiao, F., Song, L., Guo, Y., Wang, Z., Zhang, Y., Qin, B., Yang, W., Wang, F., Chan, R. H. M., & She, Q. (2020). Are we ready for service robots? the OpenLORIS-Scene datasets for lifelong SLAM. *2020 International Conference on Robotics and Automation (ICRA)*, 3139–3145 (p. 79).

Shrikumar, A., Greenside, P., Shcherbina, A., & Kundaje, A. (2016). Not just a black box: Learning important features through propagating activation differences. *ArXiv*, *abs/1605.01713*. https://api.semanticscholar.org/CorpusID:8564234 (p. 176).

Shrikumar, A., Su, J., & Kundaje, A. (2018). Computationally efficient measures of internal neuron importance. (P. 180).

Shu, J., Zhao, Q., Chen, K., Xu, Z., & Meng, D. (2020). Learning adaptive loss for robust learning with noisy labels. *ArXiv preprint*, *abs/2002.06482*. https://arxiv.org/abs/2002.06482 (p. 44).

Shu, R., Bui, H. H., Narui, H., & Ermon, S. (2018). A DIRT-T approach to unsupervised domain adaptation. *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. https://openreview.net/forum?id=H1q-TM-AW (pp. 42, 43, 260).

Siegle, J. H., Jia, X., Durand, S., Gale, S., Bennett, C., Graddis, N., Heller, G., Ramirez, T. K., Choi, H., Luviano, J. A., et al. (2021a). Survey of spiking in the mouse visual system reveals functional hierarchy. *Nature*, *592*(7852), 86–92 (p. 5).

Siegle, J. H., Jia, X., Durand, S., Gale, S. D., Bennett, C., Graddis, N., Heller, G., Ramirez, T., Choi, H., Luviano, J. A., Groblewski, P. A., Ahmed, R., Arkhipov, A., Bernard, A., Billeh, Y. N., Brown, D., Buice, M. A., Cain, N., Caldejon, S., … Koch, C. (2021b). Survey of spiking in the mouse visual system reveals functional hierarchy. *Nature* (pp. 119, 131, 138).

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR, abs/1312.6034* (pp. 176, 179).

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)* (pp. 26, 231).

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: Removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (p. 182).

Smith, A. C., & Brown, E. N. (2003). Estimating a state-space model from point process observations. *Neural computation*, *15*(5), 965–991 (p. 13).

Sofroniew, N. J., Flickinger, D., King, J., & Svoboda, K. (2016). A large field of view two-photon mesoscope with subcellular resolution for in vivo imaging. *elife*, *5*, e14472 (pp. 2, 7).

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., & Raffel, C. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS* (p. 42).

Song, H., Kim, M., Park, D., & Lee, J.-G. (2020a). Learning from noisy labels with deep neural networks: A survey. *ArXiv preprint*, *abs/2007.08199*. https://arxiv.org/abs/2007.08199 (p. 44).

Song, K.-I., Seo, H., Seong, D., Kim, S., Yu, K. J., Kim, Y.-C., Kim, J., Kwon, S. J., Han, H.-S., Youn, I., et al. (2020b). Adaptive self-healing electronic epineurium for chronic bidirectional neural interfaces. *Nature communications*, *11*(1), 4195 (p. 9).

Sprekeler, H., Zito, T., & Wiskott, L. (2014). An extension of slow feature analysis for nonlinear blind source separation. *The Journal of Machine Learning Research*, *15*(1), 921–947 (p. 101).

Steinmetz, N. A., Aydin, C., Lebedeva, A., Okun, M., Pachitariu, M., Bauza, M., Beau, M., Bhagat, J., Böhm, C., Broux, M., et al. (2021). Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, *372*(6539), eabf4588 (p. 2).

Stevenson, I. H., & Kording, K. P. (2011). How advances in neural recording affect data analysis. *Nature neuroscience*, *14*(2), 139–142 (pp. 2, 7).

Subbotin, M. F. (1923). On the law of frequency of error. *Mat. Sb.*, *31*(2), 296–301 (p. 104).

Sugiyama, M., & Kawanabe, M. (2012). *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press. (P. 25).

Sun, B., Feng, J., & Saenko, K. (2017). Correlation alignment for unsupervised domain adaptation. In *Domain adaptation in computer vision applications* (pp. 153–171). Springer. (P. 33).

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., … Anguelov, D. (2020). Scalability in perception for autonomous driving: Waymo open dataset. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 79).

Sun, Y., Tzeng, E., Darrell, T., & Efros, A. A. (2019a). Unsupervised domain adaptation through self-supervision. *ArXiv preprint*, *abs/1909.11825*. https://arxiv.org/abs/1909.11825 (pp. 47–49, 78).

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., & Hardt, M. (2019b). Test-time training for out-of-distribution generalization. *CoRR*, *abs/1909.13231* (pp. 33, 40, 41, 50, 69, 78, 244, 246, 249).

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:16747630 (pp. 176, 180, 182).

Svoboda, K., Denk, W., Kleinfeld, D., & Tank, D. W. (1997). In vivo dendritic calcium dynamics in neocortical pyramidal neurons. *Nature*, *385*(6612), 161–165 (pp. 2, 7).

Symvoulidis, P., Lauri, A., Stefanoiu, A., Cappetta, M., Schneider, S., Jia, H., Stelzl, A., Koch, M., Perez, C. C., Myklatun, A., et al. (2017). Neubtracker—imaging neurobehavioral dynamics in freely behaving fish. *Nature methods*, *14*(11), 1079–1082 (pp. 5, 10, 14).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 26, 231).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Conference on computer vision and pattern recognition (CVPR)* (pp. 26, 231).

Takei, T., Lomber, S. G., Cook, D. J., & Scott, S. H. (2021). Transient deactivation of dorsal premotor cortex or parietal area 5 impairs feedback control of the limb in macaques. *Current Biology*, *31*(7), 1476–1487 (p. 7).

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. *Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 26, 231).

Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning (ICML)* (pp. 40, 46, 87, 241).

Tang, X., Shen, H., Zhao, S., Li, N., & Liu, J. (2023). Flexible brain–computer interfaces. *Nature Electronics*, *6*(2), 109–118 (p. 7).

Tange, O. (2011). Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, *36*(1), 42–47. http://www.gnu.org/s/parallel (pp. 212, 262).

Tangemann, M., Schneider, S., Von Kügelgen, J., Locatello, F., Gehler, P. V., Brox, T., Kuemmerer, M., Bethge, M., & Schölkopf, B. (2023). Unsupervised object learning via common fate. *2nd Conference on Causal Learning and Reasoning* (p. 20).

Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., & Schmidt, L. (2020). Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, *33*, 18583–18599 (p. 251).

Tian, L., Hires, S. A., Mao, T., Huber, D., Chiappe, M. E., Chalasani, S. H., Petreanu, L., Akerboom, J., McKinney, S. A., Schreiter, E. R., et al. (2009). Imaging neural activity in worms, flies and mice with improved gcamp calcium indicators. *Nature methods*, *6*(12), 875–881 (p. 7).

Tian, Y., Krishnan, D., & Isola, P. (2019). Contrastive multiview coding. *arXiv preprint arXiv:1906.05849* (pp. 98, 100, 101, 111).

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., & Isola, P. (2020). What makes for good views for contrastive learning. (Pp. 98, 100).

Todorov, E. (2004). Optimality principles in sensorimotor control. *Nature neuroscience*, *7*(9), 907–915 (p. 7).

Tong, Z., Song, Y., Wang, J., & Wang, L. (2022). Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, *35*, 10078–10093 (p. 17).

Tralie, C., Mease, T., & J.Perea. (2018a). Dreimac: Dimension reduction with eilenberg-maclane coordinates. *GitHub*. https://github.com/ctralie/DREiMac (p. 152).

Tralie, C., Saul, N., & Bar-On, R. (2018b). Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, *3*(29), 925. https://doi.org/10.21105/joss.00925 (p. 151).

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., & Lucic, M. (2020). On mutual information maximization for representation learning. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* (p. 100).

Tsypkin, Y. (1968). Self-learning–what is it? *IEEE Transactions on Automatic Control*, *13*(6), 608–612. https://doi.org/10.1109/TAC.1968.1099015 (pp. 39, 43).

Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., & Li, Y. (2022). Maxvit: Multi-axis vision transformer. *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, 459–479 (pp. 75, 76).

Turishcheva, P., Fahey, P. G., Hansel, L., Froebe, R., Ponder, K., Vystrčilová, M., Willeke, K. F., Bashiri, M., Wang, E., Ding, Z., et al. (2023). The dynamic sensorium competition for predicting large-scale mouse visual cortex activity from videos. *arXiv preprint arXiv:2305.19654* (p. 197).

Tybrandt, K., Khodagholy, D., Dielacher, B., Stauffer, F., Renz, A. F., Buzsáki, G., & Vörös, J. (2018). High-density stretchable electrode grids for chronic neural recording. *Advanced Materials*, *30*(15), 1706520 (p. 9).

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7167–7176 (pp. 38, 78).

Urai, A. E., Doiron, B., Leifer, A. M., & Churchland, A. K. (2022a). Large-scale neural recordings call for new insights to link brain and behavior. *Nature neuroscience*, *25*(1), 11–19 (pp. 2, 5, 198, 361).

Urai, A. E., Doiron, B., Leifer, A. M., & Churchland, A. K. (2022b). Large-scale neural recordings call for new insights to link brain and behavior. *Nature Neuroscience*, *25*, 11–19 (pp. 114, 117, 154, 176).

Van de Ven, G. M., & Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734* (p. 66).

van den Oord, A., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *ArXiv*, *abs/1807.03748* (pp. 178, 182, 183).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(11) (p. 199).

Van Der Maaten, L., Postma, E., Van den Herik, J., et al. (2009). Dimensionality reduction: A comparative. *J Mach Learn Res*, *10*(66-71), 13 (pp. 117, 150).

Vargas-Irwin, C. E., Shakhnarovich, G., Yadollahpour, P., Mislow, J. M., Black, M. J., & Donoghue, J. P. (2010). Decoding complete reach and grasp actions from local primary motor cortex populations. *Journal of neuroscience*, *30*(29), 9659–9669 (p. 114).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30* (pp. 5, 195, 200).

Villani, C. (2008). *Optimal transport: Old and new* (Vol. 338). Springer Science & Business Media. (P. 206).

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Contributors, S. 1. 0. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/https://doi.org/10.1038/s41592-019-0686-2 (pp. 212, 262).

Walt, S. v. d., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22–30 (p. 145).

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., & Darrell, T. (2020a). Fully test-time adaptation by entropy minimization. *CoRR*, *abs/2006.10726* (pp. 33, 40–42, 44, 48, 50, 52–55, 72).

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., & Darrell, T. (2020b). Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726* (pp. 66, 67, 70, 74, 78, 269, 271).

Wang, H., Ge, S., Lipton, Z., & Xing, E. P. (2019). Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 10506–10518 (p. 60).

Wang, Q., Fink, O., Van Gool, L., & Dai, D. (2022). Continual test-time domain adaptation. *arXiv preprint arXiv:2203.13591* (pp. 66–69, 72, 74, 78, 193, 271).

Wang, T., & Isola, P. (2020). Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, *119*, 9929–9939 (pp. 98, 100, 102, 104, 140, 141, 158–160, 183, 196, 290, 295, 304).

Wei, C., Shen, K., Chen, Y., & Ma, T. (2020). Theoretical analysis of self-training with deep networks on unlabeled data. *ICLR* (p. 42).

Weisstein, E. (2020). Standard deviation distribution. https://mathworld.wolfram.com/StandardDeviationDistribution.html (p. 223).

Wightman, R. (2019). Pytorch image models. https://doi.org/10.5281/zenodo.4414861 (pp. 259, 262).

Williams, A. H., Kunz, E., Kornblith, S., & Linderman, S. (2021). Generalized shape metrics on neural representations. *Advances in Neural Information Processing Systems*, *34*, 4738–4750 (p. 197).

Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., Haddock, S. H., Huff, K. D., Mitchell, I. M., Plumbley, M. D., et al. (2014). Best practices for scientific computing. *PLoS biology*, *12*(1), e1001745 (pp. 3, 15).

Wolpert, D. M., Diedrichsen, J., & Flanagan, J. R. (2011). Principles of sensorimotor learning. *Nature reviews neuroscience*, *12*(12), 739–751 (p. 7).

Wu, A., Roy, N. A., Keeley, S., & Pillow, J. W. (2017). Gaussian process based nonlinear latent structure discovery in multivariate spike train data. *Advances in neural information processing systems*, *30* (p. 13).

Wu, M., Zhuang, C., Yamins, D., & Goodman, N. (2020). On the importance of views in unsupervised representation learning (p. 98).

Wu, Y., & He, K. (2018). Group normalization. *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 30, 51, 53, 232).

Wu, Z., Xiong, Y., Yu, S. X., & Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 3733–3742. https://doi.org/10.1109/CVPR.2018.00393 (pp. 98, 100, 101, 111).

Wulfmeier, M., Bewley, A., & Posner, I. (2018). Incremental adversarial domain adaptation for continually changing environments. *2018 IEEE International conference on robotics and automation (ICRA)*, 4489–4495 (p. 78).

Xie, C., & Yuille, A. L. (2020). Intriguing properties of adversarial training. *International Conference on Learning Representations (ICLR)* (p. 33).

Xie, Q., Luong, M.-T., Hovy, E., & Le, Q. V. (2020a). Self-training with noisy student improves imagenet classification. *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, 10687–10698 (pp. xxii, 33, 38, 40, 43, 46, 48, 66, 194, 240–242, 247).

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Conference on computer vision and pattern recognition (CVPR)* (pp. xxii, 26, 29, 46, 48, 75, 76, 231, 232, 242).

Xie, S. M., Kumar, A., Jones, R., Khani, F., Ma, T., & Liang, P. (2020b). In-n-out: Pre-training and self-training using auxiliary information for out-of-distribution robustness. *arXiv preprint arXiv:2012.04550* (p. 42).

Yang, H., Zhang, R., & Robinson, P. (2016). Human and sheep facial landmarks localisation by triplet interpolated features. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1–8 (p. 84).

Ye, S., Filippova, A., Lauer, J., Vidal, M., Schneider, S., Qiu, T., Mathis, A., & Mathis, M. W. (2023a). Superanimal models pretrained for plug-and-play analysis of animal behavior. *arXiv preprint arXiv:2203.07436* (pp. 20, 194).

Ye, S., Lauer, J., Zhou, M., Mathis, A., & Mathis, M. W. (2023b). Amadeusgpt: A natural language interface for interactive animal behavioral analysis. *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=9AcG3Tsyoq (p. 203).

Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S., Shenoy, K. V., & Sahani, M. (2008a). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Advances in neural information processing systems*, *21* (p. 13).

Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., & Sahani, M. (2008b). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of neurophysiology*, *102 1*, 614–35 (p. 114).

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 79).

Yu, K. J., Kuzum, D., Hwang, S.-W., Kim, B. H., Juul, H., Kim, N. H., Won, S. M., Chiang, K., Trumpis, M., Richardson, A. G., et al. (2016). Bioresorbable silicon electronics for transient spatiotemporal mapping of electrical activity from the cerebral cortex. *Nature materials*, *15*(7), 782–791 (p. 9).

Yue, X., Zheng, Z., Zhang, S., Gao, Y., Darrell, T., Keutzer, K., & Vincentelli, A. S. (2021). Prototypical cross-domain self-supervised learning for few-shot unsupervised domain adaptation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13834–13844 (p. 78).

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *CoRR*, *abs/1605.07146* (pp. 26, 46, 48, 54, 231).

Zalta, E. N. (Ed.). (2020). Scientific Research and Big Data. In *The Stanford encyclopedia of philosophy* (Summer 2020). Metaphysics Research Lab, Stanford University. (P. 192).

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. https://openreview.net/forum?id=Sy8gdB9xx (p. 44).

Zhang, H., Dauphin, Y. N., & Ma, T. (2019). Fixup initialization: Residual learning without normalization. *CoRR, abs/1901.09321* (pp. 30, 233).

Zhang, M., Levine, S., & Finn, C. (2021). Memo: Test time robustness via adaptation and augmentation. *arXiv preprint arXiv:2110.09506* (pp. 41, 55, 249).

Zhang, M., Levine, S., & Finn, C. (2022). Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems*, *35*, 38629–38642 (pp. 67, 78, 269).

Zhang, R. (2019). Making convolutional networks shift-invariant again. *International Conference on Machine Learning (ICML)* (p. 33).

Zhang, Z., & Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31: Annual conference on neural information processing systems 2018, neurips 2018, december 3-8, 2018, montréal, canada* (pp. 8792–8802). https://proceedings.neurips.cc/paper/2018/hash/f2925f97bc13ad2852a7a551802feeao-Abstract.html (p. 44).

Zhao, C., Chen, S., Zhang, L., Zhang, D., Wu, R., Hu, Y., Zeng, F., Li, Y., Wu, D., Yu, F., et al. (2023). Miniature three-photon microscopy maximized for scattered fluorescence collection. *Nature Methods*, *20*(4), 617–622 (p. 8).

Zhao, H., Jia, J., & Koltun, V. (2020a). Exploring self-attention for image recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10076–10085 (p. 48).

Zhao, Y., & Park, I. M. (2017). Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural computation*, *29*(5), 1293–1316 (p. 13).

Zhao, Z., Klindt, D. A., Maia Chagas, A., Szatko, K. P., Rogerson, L., Protti, D. A., Behrens, C., Dalkara, D., Schubert, T., Bethge, M., et al. (2020b). The temporal structure of the inner retina at a single glance. *Scientific reports*, *10*(1), 4399 (p. 193).

Zhou, D., & Wei, X. (2020). Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-vae. *Advances in Neural Information Processing Systems 33*. https://proceedings.neurips.cc/paper/2020/hash/510f2318f324cf07fce24c3a4b89c771-Abstract.html (pp. 12, 114, 117–119, 124, 137, 138, 148–150, 157).

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2019). A comprehensive survey on transfer learning. *arXiv preprint arXiv:1911.02685* (p. 84).

Zimmermann, R. S., Sharma, Y., Schneider, S., Bethge, M., & Brendel, W. (2021a). Contrastive learning inverts the data generating process. *Proceedings of the 38th International Conference on Machine Learning*, *139*, 12979–12990. http://proceedings.mlr.press/v139/zimmermann21a.html (pp. 118, 140, 153, 158, 163, 169–173).

Zimmermann, R. S., Sharma, Y., Schneider, S., Bethge, M., & Brendel, W. (2021b). Contrastive learning inverts the data generating process. *International Conference on Machine Learning*, 12979–12990 (pp. 178, 179, 184, 185).

Zoltowski, D., Pillow, J., & Linderman, S. (2020). A general recurrent state space framework for modeling neural dynamics during decision-making. *International Conference on Machine Learning*, 11680–11691 (p. 13).

Zou, Y., Yu, Z., Liu, X., Kumar, B., & Wang, J. (2019). Confidence regularized self-training. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5982–5991 (p. 42).

Index

Photographic credits

Figure 1.1 Adapted based on code from Urai et al. (2022a), licensed under CC-BY.

Figure 1.2 uses https://doi.org/10.5281/zenodo.3925903 by Luigi Petrucco and https://doi.org/10.5281/zenodo.3926119 by Agustin Carpaneto, both under CC-BY.

Figure 1.3: Reprinted from *Nature neuroscience*, 20(3), Bassett, D. S., and Sporns, O. (2017), "Network neuroscience." pp. 353-364. Reproduced with permission from Springer Nature.

Figure 1.4 Reprinted from Hong et al. (2021), licensed under CC-BY.

Figure 1.5: Reprinted from *Neuron* 93(3), Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A., and Poeppel, D., "Neuroscience needs behavior: correcting a reductionist bias." pp. 480-490, Copyright (2017), with permission from Elsevier.

Figure 1.6 Reprinted from *Current Opinion in Neurobiology* 60, Mathis, M.W., and Mathis, A., "Deep learning tools for the measurement of animal behavior in neuroscience.". pp. 1-11. with permission from Elsevier.

All Figures and Tables in Chapter 7 and Appendix F Reprinted from Schneider et al. (2023), which is an open access article distributed under the terms of the Creative Commons CC BY license, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Publisher: Springer Nature.

Curriculum Vitae

Steffen Schneider

Contact Information

Website	stes.io
Email	stes@hey.com
Google Scholar	scholar.google.com/citations?user=KR5dj44AAAAJ
Github	github.com/stes

Education

11/19 – 01/24	Ph. D. Candidate, Neuroscience, EPFL EDNE (since 01/2021) and International Max Planck Research School for Intelligent Systems Tübingen (via ELLIS PhD Program)
10/16 – 12/18	M. Sc. Neuroengineering, with high distinction, Technical University Munich
10/13 – 09/16	B. Sc. Electrical Engineering, Information Technology & Computer Engineering, with excellence, RWTH Aachen University

Research and Professional Experience

02/23 – now	Co-Founder and CTO, Kinematik AI, LLC
04/21 – now	Co-Founder and CEO (Geschäftsführer), KI macht Schule gUG (haftungsbeschränkt)

05/22 – 07/22	Research Scientist Intern, Computer Vision, FAIR at Meta AI, New York
09/20 – 01/21	Applied Science Intern, Adaptation and object centric learning, Amazon Web Services, Tübingen
02/20 – 03/20	Visiting PhD student, The Rowland Institute at Harvard, Cambridge, Massachusetts
10/18 – 09/19	AI Resident, Speech Recognition, Facebook AI Research, Menlo Park, California
04/18 – 09/18	Master's Thesis Student, International Max Planck Research School & University of Tübingen
06/17 – 04/18	Research Assistant, Multivariate Data Analysis for Molecular Imaging, Institute of Biological and Medical Imaging, Helmholtz Zentrum Munich & Klinikum rechts der Isar, TU Munich
05/16 – 02/18	Research Assistant, Medical Computer Vision, Institute of Imaging & Computer Vision, RWTH Aachen University
03/17 – 05/17	Research Intern, Deep Learning and EEG Signal Analysis, School of Computing, University of Kent and Data Science Institute, Imperial College London
10/14 – 06/15	Teaching Assistant, Electrical Engineering, RWTH Aachen University

Additional academic activities

Student supervision and mentoring

Theses	Bethge Lab: Shubham Krishna, Jan Hansen-Palmus (as co-supervisor), Khushdeep Singh Mann; Mathis Lab: Jin H Lee, Rodrigo González, Anastasiia Filippova
Projects	Bethge Lab: Shubham Krishna, Bozidar Antic, Jugoslav Stojcheski; Mathis Lab: Célia Benquet, Xingying Chen, Pauline Lauwers

Reviewing

Journals	Cell (co-reviewing)

Conferences	NeurIPS 2021, ICLR 2022 (highlighted reviewer), ICLR 2023, CVPR 2023, NeurIPS 2023, ICLR 2024, AISTATS 2024, CVPR 2024, TMLR
Workshops	NeurIPS 2023 SSL; NeurIPS 2022 DistShift; ICML 2022 ShiftHappens; ICML 2021 UDL; BIA 2017, DLMIA 2017 and DLMIA 2018 workshops at MICCAI 2017 and 2018

Selected Scientific Workshops and Meetings

06/2022	Co-Organizer and Panel Moderation, Shift Happens Workshop at ICML'22. (>200 participants)
02/2021	Lead Organizer, Tuebingen AI Symposium 2020 (130 participants)
07/2021	Panel Moderation, ELLIS PhD and Postdoc Summit
10/2021	Talk and Workshop on robust vision at the MSNE project week, TU Munich
01/2020	Instructor, Introductory course on Quantum machine learning
06/2018	Co-Organizer, SmartStart Computational Neuroscience Retreat, Freiburg
09/2017	Co-Organizer, Biomodeling retreat, Barcelona

List of Publications

Articles in peer-reviewed journals and proceedings

2023	Learnable latent embeddings for joint behavioural and neural analysis. ***Nature***, *2023*. Steffen Schneider*, Jin Hwa Lee*, and Mackenzie Weygandt Mathis.
2023	Rdumb: A simple approach that questions our progress in continual test-time adaptation. ***Neural Information Processing Systems (NeurIPS)***, *2023*. Ori Press, Steffen Schneider, Matthias Kuemmerer, and Matthias Bethge.

2023 Unsupervised object learning via common fate.
***Conference on Causal Learning and Reasoning (CLeaR)**, 2023*.
Matthias Tangemann, Steffen Schneider, Julius von Kügelgen, Francesco Locatello, Peter Gehler, Thomas Brox, Matthias Kümmerer, Matthias Bethge, and Bernhard Schölkopf.

2022 If your data distribution shifts, use self-learning.
***Transactions on Machine Learning Research**, and contributed talk at the ICML 2021 WeaSuL Workshop, 2022*.
Evgenia Rusak*, Steffen Schneider*, George Pachitariu, Luisa Eck, Peter Gehler, Oliver Bringmann, Wieland Brendel, and Matthias Bethge.

2022 Multi-animal pose estimation, identification and tracking with DeepLabCut.
***Nature Methods** 19, 496–504, 2022*.
Jessy Lauer, Mu Zhou, Shaokai Ye, William Menegas, Steffen Schneider, Tanmay Nath, Mohammed Mostafizur Rahman, Valentina Di Santo, Daniel Soberanes, Guoping Feng, Venkatesh N. Murthy, George Lauder, Catherine Dulac, Mackenzie Weygandt Mathis, and Alexander Mathis.

2021 Contrastive learning inverts the data generating process.
***International Conference on Machine Learning (ICML)**, 2021*.
Roland Zimmermann*, Yash Sharma*, Steffen Schneider*, Matthias Bethge, and Wieland Brendel.

2021 Pretraining boosts out-of-domain robustness for pose estimation.
***Winter Conference on Applications of Computer Vision (WACV)**, 2021*.
Alexander Mathis*, Thomas Biasi*, Steffen Schneider, Mert Yüksekgönül, Byron Rogers, Matthias Bethge, and Mackenzie Mathis.

2020 Improving robustness against common corruptions by covariate shift adaptation.
***Neural Information Processing Systems (NeurIPS)**, 2020; also in: ICML UDL Workshop (**oral**, top 5%), 2020*.
Steffen Schneider*, Evgenia Rusak*, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge.

2020 A primer on motion capture with deep learning: Principles, pitfalls and perspectives.
***Neuron** 108(1), 44–65, 2020.*
Alexander Mathis, Steffen Schneider, Jessy Lauer, and Mackenzie Mathis.

2020 vq-wav2vec: Self-supervised learning of discrete speech representations.
***International Conference on Learning Representations (ICLR)**, 2020.*
Alexei Baevski*, Steffen Schneider*, and Michael Auli.

2019 wav2vec: Unsupervised pre-training for speech recognition.
***Interspeech**, 2019.*
Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli.

2019 Iron-sequestering nanocompartments as multiplexed electron microscopy gene reporters.
***ACS Nano** 13 (7), 8114–8123, 2019.*
Felix Sigmund, Susanne Pettinger, Massimo Kube, Fabian Schneider, Martina Schifferer, Michaela Aichler, Steffen Schneider, Axel Walch, Thomas Misgeld, Hendrik Dietz, et al.

2017 Neubtracker — imaging neurobehavioral dynamics in freely behaving fish.
***Nature Methods** 14 (11), 1079–1082, 2017.*
Panagiotis Symvoulidis, Antonella Lauri, Anca Stefanoiu, Michele Cappetta, Steffen Schneider, Hongbo Jia, Anja Stelzl, Maximilian Koch, Carlos Cruz Perez, Ahne Myklatun, Sabine Renninger, Andriy Chmyrov, Tobias Lasser, Vasilis Ntziachristos Wolfgang Wurst, and Gil G Westmeyer.

Reviews

2021 Motor control: Neural correlates of optimal feedback control theory.
***Current Biology** 31(7), R356-R358., 2021.*
Mackenzie W Mathis and Steffen Schneider.

Workshop publications and pre-prints

2023 Identifiable attribution maps using regularized contrastive learning.
NeurIPS Workshop: Self-Supervised Learning - Theory and Practice, 2023.
Steffen Schneider, Rodrigo González Laiz, Markus Frey, and Mackenzie W Mathis.

2022 SuperAnimal models pretrained for plug-and-play analysis of animal behavior
CoRR abs/2203.07436, 2022.
Shaokai Ye, Anastasiia Filippova, Jessy Lauer, Maxime Vidal, Steffen Schneider, Tian Qiu, Alexander Mathis, Mackenzie W Mathis

2021 Out-of-distribution generalization of internal models is correlated with reward.
Self-Supervision for Reinforcement Learning Workshop-ICLR 2021, 2021.
Khushdeep Singh Mann*, Steffen Schneider*, Alberto Chiappa, Jin Hwa Lee, Matthias Bethge, Alexander Mathis, and Mackenzie W Mathis.

2020 On the relationship between adaptive and invariant representation learning.
NeurIPS 2020 Preregistration Workshop, 2020.
Steffen Schneider*, Shubham Krishna*, Luisa Eck, Mackenzie W Mathis, and Matthias Bethge.

2018 Salad: A toolbox for semi-supervised adaptive learning across domains.
NeurIPS Workshop on Machine Learning Open Source Software (MLOSS), 2018.
Steffen Schneider, Alexander S. Ecker, Jakob H. Macke, and Matthias Bethge.

2018 Multi-task generalization and adaptation between noisy digit datasets: An empirical study.
NeurIPS Workshop on Continual Learning, 2018.
Steffen Schneider, Alexander S. Ecker, Jakob H. Macke, and Matthias Bethge.

2017 Context-based normalization of histological stains using deep convolutional features.
3rd Workshop on Deep Learning in Medical Image Analysis (DLMIA), 2017.
Daniel Bug*, Steffen Schneider*, Anne Grote, Eva Oswald, Friedrich Feuerhake, Julia Schüler, and Dorit Merhof.

Software

GitHub https://github.com/stes

2023 CEBRA, a method for non-linear data analysis in neuroscience, `https://github.com/stes/cebra`

2022 "Shift Happens" benchmark package for robustness evaluation (result of ICML 2022 workshop) `github.com/shift-happens-benchmark/icml-2022/`

2022 the DeepLabCut benchmark
`https://benchmark.deeplabcut.org/`

2021 robusta, a package for robustness and adaptation on ImageNet scale
`github.com/bethgelab/robustness`

2020 wav2vec and vq-wav2vec, self-supervised pre-training for speech recognition

2018 salad, a package for semi-supervised adaptive learning across domains
`domainadaptation.org`

Patents

2022 Dimensionality reduction of time-series data, and systems and devices that use the resultant embeddings. *Patent pending, filed on January 25, 2022*.