EPFL

Thèse n° 9215

Making Computer Vision Models Robust and Adaptive

Présentée le 4 décembre 2023

Faculté informatique et communications
Laboratoire d'intelligence et d'apprentissage visuels
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Shuqing Teresa YEO

Acceptée sur proposition du jury

Prof. A. M. Alahi, président du jury
Prof. A. Roshan Zamir, Prof. P. Dillenbourg, directeurs de thèse
Prof. M. Cord, rapporteur
Prof. P. W. Koh, rapporteur
Dr M. Salzmann, rapporteur

École
polytechnique
fédérale
de Lausanne

2023

Acknowledgements

This thesis is the result of the help and support of my advisors, collaborators, lab members, friends, and family. To all of whom, I am immensely grateful.

I would first like to thank my advisor Amir Zamir. Amir took a chance on me when I was a lost and clueless PhD student. He taught me how to do research, how to present (with very, very long comments on every paper, and slide deck, including comments on the redundancy of the word very) and created a wonderfully collaborative and supportive lab environment. Amir would hold us to standards higher than ourselves, and it has greatly influenced the way I think about research directions and approach problems. I am also thankful to my co-advisor Pierre Dillenbourg. Pierre offered me a lifeline when I needed one, was a listening ear all these years, and saw me through many ups and downs. Both Amir and Pierre saved my PhD and I would not be here without them.

I am honoured to have on my jury, Alexandre Alahi, Mathieu Cord, Pang Wei Koh, and Mathieu Salzmann. I am grateful for their time, discussions, and feedback.

To the members of VILAB for teaching me invaluable life-long lessons, especially all things food related like how not to eat dumplings: to Oguzhan, the deputy boss of the lab, Roman, who has so many cool cameras, Andrei, the best kimchi maker in the world. To other long-term residents: Ainaz, David, Marius, Jiawei, and the new students: Alvin, Jason, Kunal, Rishubh, thank you for making the lab lively. To those I had the pleasure of working with: Zahra, Alex, Ruchira, Harold, Pooya, thank you for making deadlines more enjoyable. Finally, to Stephanie, for making administrative work seem easy.

I am also grateful to have been adopted by the (then) giant and incredibly social CHILI family: Alexis, Arzu, Ayberk, Elmira, Hala, Jauwairia, Jenny, Kevin, Louis, Ramtin, Sina, Stian, Thibault, Łukasz, Utku, Wafa. Special thanks to Kevin and Ramtin, who kept me going with their wise words and lame jokes respectively. Hala, for the delicious Lebanese desserts and for being contagiously cheerful. During my stint at LIONS, I was fortunate to have worked closely with Kamal. Kamal would patiently distill complicated concepts for me while sharing happy memories from his PhD days. And to Chen, for company amidst the misery. I am also thankful for the people who were always up for lunch, dinner, coffee, drinks, bouldering, or on rare occasions, hikes for making my time here eventful. And very much less grateful that almost everyone graduated and left so early.

Acknowledgements

Finally, to those most affected by my constant stress and lousy schedule. They were an unwavering source of support when the going gets tough and the most ardent cheerleaders when the tough gets going. To my parents, who are always worried that I am not eating enough, and to Federico, for making sure that I am. I could not have done it without you. This thesis is for you.

Teresa

Abstract

Visual perception is indispensable for many real-world applications. However, perception models deployed in the real world will encounter numerous and unpredictable distribution shifts, for example, changes in geographic locations, motion blur, and adverse weather conditions, among many others. Thus, to be useful in the real world, these models need to generalize to the complex distribution shifts that can occur. This thesis focuses on three directions aimed at achieving this goal.

For the first direction, we introduce two robustness mechanisms. They are training-time mechanisms as inductive biases are incorporated at training-time and at test-time, the weights of the models are frozen. The first robustness mechanism we introduce ensembles predictions from a diverse set of cues. As each cue responds differently to a distribution shift, we adopt a principled way of merging these predictions and show that it can result in a final robust prediction. The second mechanism is motivated by the rigidity and biases of existing datasets. Examples of dataset biases include containing mostly scenes from developed countries, professional photographs, and so on. Here, we aim to control pre-trained generative models to generate targeted training data to account for these biases, that we can use to fine-tune our models.

Training-time robustness mechanisms attempt to anticipate the shifts that can occur. However, distribution shifts can be unpredictable and models may return unreliable predictions if this shift was not accounted for at training time. Thus, for our second direction, we propose to incorporate test-time adaptation mechanisms so that models can adapt to shifts as they occur. To do so we create a closed-loop system that learns to use feedback signals computed from the environment. We show that this system is able to adapt efficiently at test time.

For the last direction, we introduce a benchmark for testing models on realistic shifts. These shifts are attained from a set of image transformations that take the geometry of the scene into account. Thus, they are more likely to occur in the real world. We show that they can expose the vulnerabilities of existing models.

Keywords: distribution shifts, robustness, adaptation, benchmarks

Résumé

La perception visuelle est indispensable pour de nombreuses applications du monde réel. Cependant, les modèles de perception déployés dans le monde réel seront confrontés à de nombreux changements de répartition imprévisibles, par exemple des changements de localisation géographique, du flou de mouvement et des conditions météorologiques défavorables, entre autres. Ainsi, pour être utiles dans le monde réel, ces modèles doivent être généralisés aux changements de distribution complexes qui peuvent survenir. Cette thèse se concentre sur trois directions visant à atteindre cet objectif.

Pour la première direction, nous introduisons deux mécanismes de robustesse. Ce sont des mécanismes de temps de formation car des biais inductifs sont incorporés au moment de la formation et au moment du test, les poids des modèles sont gelés. Le premier mécanisme de robustesse consiste à introduire des prédictions d'ensembles à partir d'un ensemble diversifié d'indices. Ceci est basé sur l'idée que les prédictions faites via différents signaux réagissent différemment à un changement de distribution, on devrait donc pouvoir les fusionner en une seule prédiction finale robuste. Le deuxième mécanisme est motivé par la rigidité et les biais des ensembles de données existants. Des exemples de biais dans les ensembles de données incluent le fait qu'ils contiennent principalement des scènes provenant de pays développés, des photographies professionnelles, etc. Ici, nous visons à contrôler des modèles génératifs pré-entraînés pour générer des données d'entraînement ciblées pour tenir compte de ces biais, que nous pouvons utiliser pour affiner nos modèles.

Les mécanismes de robustesse du temps de formation tentent d'anticiper les changements qui peuvent survenir. Cependant, les changements de distribution peuvent être imprévisibles et les modèles peuvent renvoyer des prédictions peu fiables si ce changement n'a pas été pris en compte au moment de la formation. Ainsi, pour notre deuxième direction, nous proposons d'incorporer des mécanismes d'adaptation au temps de test afin que les modèles puissent s'adapter aux changements au fur et à mesure qu'ils se produisent. Pour ce faire, nous créons un système en boucle fermée qui apprend à utiliser les signaux de rétroaction calculés à partir de l'environnement. Nous montrons que ce système est capable de s'adapter efficacement au moment du test.

Pour la dernière direction, nous introduisons un benchmark pour tester des modèles sur des changements réalistes. Ces décalages sont obtenus à partir d'un ensemble de

transformations d'image qui prennent en compte la géométrie de la scène. Ils sont donc plus susceptibles de se produire dans le monde réel. Nous montrons qu'ils peuvent exposer les vulnérabilités des modèles existants.

Mots-clés : décalages de distribution, robustesse, adaptation, benchmarks

Contents

Introduction

Visual perception models have been adopted in many real-world applications. These applications range from safety-critical ones like autonomous driving, and diagnosing diseases, to public policy and hiring decisions. However, when faced with distribution shifts from its training data, the predictions of these models are unreliable and can exhibit unintended biases. As the real world consists of rich, complex, and unpredictable distribution shifts, this is a challenge that needs to be solved before deploying these models in the real world.

This thesis presents three directions for improving the predictions of models under distribution shifts. The first involves robustness mechanisms. These mechanisms incorporate inductive biases into the model during training-time, in anticipation of the shifts that can occur at test time. The second direction studies adaptation mechanisms, which allows the model to adapt to shifts as they occur. Finally, for the last direction, we introduce a benchmark that can be used to evaluate models. See Figure 1 for a schematic overview of these directions.

Types of distribution shifts

One way to categorize distribution shifts is by *how the distributions differ at training and test-time*. Let us assume that the input data can be decomposed into a set of attributes. Thus, distribution shifts arise when the distribution over these attributes is not the same at training and test-time [1]. This gives rise to several types of shifts that commonly occur in the real world. *1. Unseen shifts:* This occurs when certain attributes of the data are not seen during training time but appear at test time e.g. noise or blur. *2. Long-tails:* Some examples of this are when data from certain regions or demographics are under-sampled. *3. Spurious correlations:* This occurs when the attributes that are correlated at training time are not correlated at test time e.g., an object that occurs with a certain background during training time may appear with a different background at test time. A model that learns to make predictions based on the background would perform poorly at test time.

Another way to categorize distribution shifts is by *how they are generated*. Popular

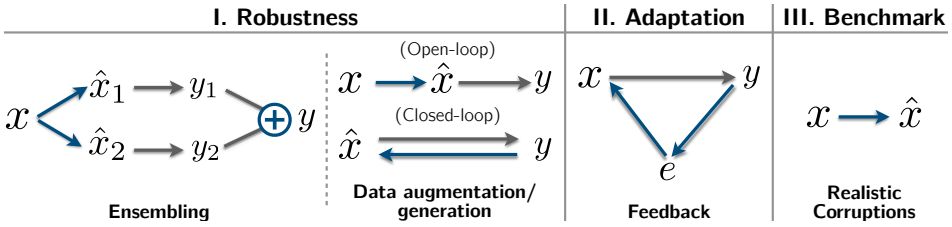

Figure 1: **A schematic overview of the three directions presented in this thesis.** The model of interest in all cases is $x \rightarrow y$ i.e., the mapping from the RGB image to the target task e.g., image classification or depth estimation. The blue arrows or symbols denote the key components of these mechanisms. **I. Robustness mechanisms** are training-time mechanisms as they incorporate inductive biases into the model at training time and at test time the weights of the model are frozen. Ensemble methods combine the predictions from several models to get a final strong prediction. We propose to apply transformations to the input x to get a set of cues, $\hat{x}_i$s. We then train a model to predict the target task from each of these cues. As each cue responds differently to distribution shifts, we adopt a principled way of merging these predictions and show that it can result in a final robust prediction. In Chapter 4, we introduce a set of realistic transformations i.e., $x \rightarrow \hat{x}$, and show that augmenting the training data with these transformations improves robustness. However, these augmentations are generated in an open-loop manner, i.e., without feedback from the model. Thus, in Chapter 2, instead of applying transformations to the input, we learn to control a generative model to generate training data for a given model by using its loss as a feedback signal. **II. Adaptation mechanisms** are test-time mechanisms as they aim to adapt to distribution shifts as they occur. To do so, they compute an adaptation signal from the environment, e, at test time and use it to update the model. Thus, creating a closed-loop system with the environment. In Chapter 3, we propose an efficient way of performing test-time adaptation. **III. Benchmarks** allow us to evaluate our models before deploying them in the real world. In Chapter 4, we introduce a set of realistic corruptions, informed by the geometry of the scene. We show that they can be computed efficiently and can expose the vulnerabilities of existing networks.

examples include low-level shifts like Gaussian blur, speckle noise [2] etc., and viewpoint changes that can result in occlusions [3, 4]. There can also be changes in geographic location [5, 6], time [7, 8] and so on. One can also *optimize for shifts* to fool a given model, creating an adversarial attack. The adversarial attack can be optimized to be imperceptible [9, 10, 11], over spatial transformation like rotation and translation parameters [12], or even over how the training and test data is split [13]. In this thesis, we focus mostly on unseen shifts, and in particular low-level and optimized ones.

Reasons for lack of robustness

Our training datasets tend to exhibit biases. Some examples include photographer's bias, i.e., where photographers tend to take pictures of objects in similar ways, and as a result, most objects tend to be centered or taken in their frontal view [14, 15, 16]. Another

example is location bias, when there is more data from certain parts of the world or ethnicity [17] compared to others e.g., satellite images from the Americas compared to that from Africa [18]. Thus, training our models on data with these biases can result in poor performance at test-time, when it is given a e.g., different viewpoint of an object or images from different countries. Furthermore, learning with stochastic gradient descent can return models that prefer certain attributes over others. [19] showed that networks exhibit simplicity bias, i.e., when there exist several attributes of the input data that are predictive of the target task, networks prefer to use the "simpler" attributes. These "simpler" attributes are e.g., background or texture, as opposed to semantic features of the object. This causes the model to be sensitive to perturbations in the input or style changes.

There are many other hypotheses for why our models are not robust, from learning from unnatural static I.I.D. data, typically of a single modality, being disconnected from the environment i.e., lack of interaction, and so on. A discussion of this can be found in the conclusion section, under future work. Thus, the above paragraph presents a narrow view of the possible reasons, as it assumes that we are constrained to training black box neural networks on I.I.D. inputs.

Making models robust

Robustness mechanisms *anticipate* the distribution shift that can occur and incorporate inductive biases into the model to help it generalize. They are *training-time mechanisms* as these inductive biases are incorporated at training-time and at test-time the weights of the models are frozen. There are several ways of making models more robust e.g., data augmentations, architecture changes, and so on. We will discuss the ones relevant to this thesis.

Ensemble methods combine the predictions of several models to attain a strong final prediction. Ensembles rely on the diversity of individual models to de-correlate errors. Sources of diversity include using different initializations [20], hyperparameters [21] or network architectures [22] for the ensemble components, or training the ensemble with additional loss terms [23, 24, 25, 26]. Other works average the weights of several models, to return one final model [27, 28, 29]. This has the benefit of having no additional cost at inference time. In Chapter 1, we propose to make our predictions from a diverse set of cues. This is based on the idea that predictions made via different cues respond differently to distribution shifts, hence one should be able to merge them into one robust final prediction. We perform the merging in a straightforward but principled manner based on the uncertainty associated with each prediction.

Learning with data from generative models. Sampling data from generative models such as generative adversarial networks (GANs) or diffusion models allows us to vastly expand the original training data. Recent works have made use of the latter to either

replace or expand the training data for image recognition tasks [30, 31, 32]. These works used pre-defined prompts or a language model to augment the pre-defined prompts, to generate data. Other works learn to generate training data for a given model by controlling the latent space of GANs [33, 34] or variational autoencoders (VAEs) [35]. We show in Chapter 2 that we can generate data for a given model but in contrast to these works, we employ diffusion models as the data generator.

Robustness benchmarks allow us to evaluate our models before deploying them in the real world. The WILDS benchmark [18] is a combination of 10 datasets that represents shifts that can arise in the real world e.g., shifts across time, location in satellite images, or hospitals. In contrast, the Common Corruptions benchmark [2] introduced synthetic corruptions on real images that expose the sensitivities of image recognition models. These corruptions are cheap to apply, however, they do not take into account the geometry of the scene, i.e., the corruptions are applied uniformly over the image. In Chapter 4, we propose 3D Common Corruptions, that modify real images *using 3D information* to generate realistic corruptions. Thus, leading to corruptions that are more likely to occur in the real world.

Data-augmentations. Increasing the diversity of the training data via data augmentation is a popular way of improving the model's robustness. Methods such as AugMix [36], AutoAugment [37], RandAugment [38] combine standard augmentations like color jitter, rotation, shearing. Methods like CutMix [39], Mixup [40] introduce new augmentations by replacing a portion of an image with a crop of another image or by linearly combining two images. Other works train with low-level shifts like noise or blur [41, 42], or adversarial attacks [11]. In Chapter 4, we introduce a set of realistic corruptions that incorporates the geometry of the scene into the transformations and show that training with these realistic corruptions results in more robust predictions.

Domain Generalization assume access to datasets from multiple domains during training [43, 44]. Some methods involve meta-learning [45], matching the features across the different domains [46, 47, 48] or enforcing invariance in the gradients [49]. Invariant risk minimization [50] learns features such that the optimal linear classifier on top of that representation matches across domains. Group distributionally robust optimization [51] minimizes the worst-case loss over domains. However, [52] showed that empirical risk minimization, when well implemented, achieves state-of-the-art performance across datasets. In this thesis, we assume access to only one domain during training, although extending this to multiple domains would be interesting for future work.

Making models adaptive

Distribution shifts that can occur in the real world are numerous and unpredictable. Training-time robustness mechanisms that attempt to take anticipatory measures (e.g., data augmentation or architecture changes) have inherent limitations. This is because the

learned model is frozen at test-time, thus upon encountering an out-of-distribution input, its predictions may collapse. This is the main motivation behind test-time adaptation methods, which instead aim to adapt to such shifts as they occur. These methods choose *adaptation over anticipation*.

Adaptive methods create a closed loop with the environment and use an adaptation signal at test time. The adaptation signal is a quantity that can be computed at test time from the environment. Optimizing a self-supervised signal e.g. entropy [53], mutual information [54] at test-time is a popular way of performing adaptation. While this can successfully adapt a network, it is inefficient as it does not make use of the learnable regularities in the adaptation process, and consequently, is unconducive for real-world applications. It also results in a rigid framework as the update mechanism is fixed to be the same as the training process of neural networks, i.e., using stochastic gradient descent. In Chapter 3, we show that this process can be effectively amortized using a controller network, which yields orders of magnitude faster results. In addition, it provides flexibility advantages as the controller is implemented using a neural network and can be engineered to include arbitrary inductive biases and desired features.

Robustness Mechanisms Part I

1 Ensembling diverse predictions

1.1 Introduction

We begin the first part on robustness mechanisms by demonstrating a way to use ensembling techniques to improve predictions. Suppose we want to learn a mapping from an input domain, e.g. RGB images, to a target domain, e.g. surface normals (see Fig. 1.1). A common approach is to learn this mapping with a *direct* path, i.e. *RGB* $\rightarrow$ *surface normals*. Since this path directly operates on the input domain, it is prone to being affected by any slight alterations in the RGB image, e.g. brightness changes. An alternative can be to go through a *middle domain*[1] that is invariant to that change. For example, the surface normals predicted via the *RGB* $\rightarrow$ *2D edges* $\rightarrow$ *surface normals* path will be resilient to brightness distortions in the input as the 2D edges domain abstracts that away. However, one does not know which middle-domain to use ahead of time as the distortions that a model may encounter are broad and apriori unknown, and some middle domains can be too lossy for certain downstream predictions. These issues can be mitigated by employing an *ensembling* approach where predictions made via a diverse *set* of middle domains are merged into one strong prediction on-the-fly.

This paper presents a general approach for the aforementioned process. We first use a set of K middle domains from which we learn to predict the final domain (Fig. 1.1). Each of the K paths reacts differently to a particular distribution shift due to its inherent biases, so its prediction may or may not degrade severely. Thus, we further estimate the *uncertainty* of each path's prediction which allows us to employ a principled way of combining these predictions into the one final prediction.

Prior knowledge of the relationship between middle domains is not needed as their

[1] or equivalently "middle task", as most vision tasks can be viewed as mapping an input onto some other domain.

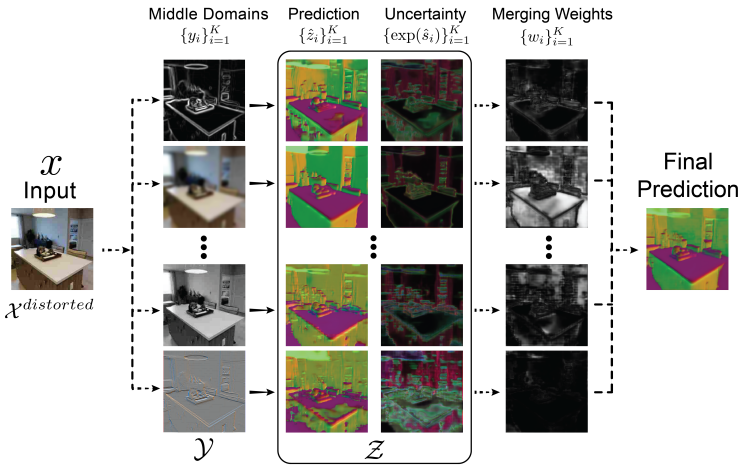

Figure 1.1: **An overview of the proposed method for creating a robust and diverse ensemble of predictions.** A set of K networks predict a target domain (here surface normals) given an input image that has undergone an unknown distribution shift (here JPEG compression degradation), via K middle domains (e.g. 2D texture edges, low-pass filtering, greyscale image, emboss filtering, etc). The predictions from the K paths are then merged into one final strong prediction using weights that are based on the uncertainty associated with each prediction. This method is shown to be significantly robust against adversarial and non-adversarial distribution shifts for several tasks. In the figure above, solid (–) and dashed (- -) arrows represent learned and analytical functions, respectively.

contribution to the final prediction is guided by their predicted uncertainties in a fully computational manner independent of the definition of the middle domain. In other words, no manual modification or re-design is needed upon a change in these domains. Moreover, the middle domains we adopt can all be *programmatically extracted*. Thus, this framework does not require any additional supervision/labeling than what a dataset already comes with. The proposed method would be equally applicable if the middle domains were also obtained using a learning based approach, e.g. predicting surface normals from the output of another network such as a depth estimator. We show in Sec. 4.5 that the method performs well insensitive to the choice of middle domains and it generalizes to completely novel non-adversarial and adversarial corruptions.

1.2 Related Work

This work has connections to a number of topics, including ensembling, uncertainty estimation and calibration, inductive bias learning [55], or works in neuroscience that suggest the brain uses multiple, sometimes partially redundant, cues to perceive [56, 57]. We give an overview of some of them within the constraints of space.

Ensembling allows us to resolve the bias-variance trade-off which states that errors

in a models' prediction can be decomposed into bias, variance, and an irreducible data-dependent noise term [58, 59]. This is done by combining multiple models with low bias and high variance, e.g. bagging [60], or with high bias and low variance, e.g. boosting [60], to have predictions with both low bias *and* variance.

A primary challenge for ensemble methods is to ensure diversity. Sources of diversity include using different initializations [20], hyperparameters [21] or network architectures [22] for the ensemble components, or training the ensemble with additional loss terms [23, 24, 25]. However, under distribution shifts, reduction in performance can stem from an increase in the bias, rather than the variance term [61]. Our set of middle domains yields a more diverse ensemble by design and promotes invariance to different distortions to keep bias low (see Fig. 1.1).

Estimating uncertainty: Uncertainty in a model's prediction can be decomposed into two sources [62, 63]. *Epistemic* uncertainty accounts for uncertainty in the model parameters, while *aleatoric* uncertainty stems from the noise inherent in the data. There are many proposed methods to estimate the former, such as using dropout [64, 65], stochastic variational inference methods [66, 67, 68, 69, 70, 71], ensembling [20], and consistency energy [72] where a *single* uncalibrated uncertainty estimate is extracted from consistency of different paths. Most of the existing methods in this area solely estimate uncertainty without using it towards improving the predictions. In contrast, our formulation estimates a calibrated uncertainty for each path *and* uses it to produce a stronger prediction.

Improving calibration with auxiliary datasets: Neural networks tend to produce outputs that are miscalibrated, i.e. their estimated uncertainty does not reflect the true likelihood of being correct [73, 74]. In particular, their predictions tend to be *overconfident* for unfamiliar examples. This is usually handled by a calibration step. Similar to [75, 76, 77, 78], we use a separate dataset from the one at test time to train the model to output high sigmas (uncertainties) for unfamiliar cases. Previous papers focus on generalizing uncertainties for classification; in Section 1.3.1, we show this can be extended to dense regression problems.

Enforcing consistency constraints in the context of cross-task predictions involves ensuring that the output predictions remain the same regardless of the intermediate domain [72, 79, 80, 81]. Particularly in contrast to [72] which uses (non-probabilistic) training-time consistency constraints to improve a network's prediction and does not have any consolidation mechanism, our goal is to robustify the final prediction by *merging the output of multiple prediction paths at the test time.* Our formulation and the *training-time* consistency constraints are complimentary.

Robustness via data augmentation: One approach to addressing robustness involves using data augmentation during training [11, 40, 36, 82, 83]. Such methods usually

involve *training with a set of corruptions* to generalize to unseen ones [41]. However, performance gains can be non-uniform, e.g. Gaussian noise augmentation improves performance on other noise corruptions (e.g. impulse, shot noise) but hurts performance on fog and contrast [84]. Instead, our main mechanism uses a large set of middle domains (not corruptions) to be resistant to a wide range of diverse *unseen* corruptions. We do not use any corruptions during training, except to calibrate uncertainty.

Adversarial attacks add imperceptible worst case shifts to the input to fool a model [9, 10, 11]. In contrast to [23, 24, 25], which are ensemble based adversarial robustness methods with an additional loss term to promote diversity, the diversity of our ensembles is a natural consequence of using different middle domains. While our focus is not limited to robustness against adversarial attacks, it yields supportive evaluations against them as well (Sec. 1.4.1).

Figure 1.2: **Addressing overconfident inaccurate predictions under high distortions. Left**: Qualitative prediction results of image (re)shading and their corresponding uncertainty estimates (i.e. sigma) under two intensities of speckle noise distortion. This is shown for a single UNet model before and after *sigma training* (ST) as well as for deep ensembles (standard deviation of predictions in the ensemble). Darker denotes lower uncertainty/sigma. ST was done using Gaussian noise and Gaussian blur distortions. *Using other distortions yields similar performance* (see appendix). **Right**: Scatter plot of ℓ_1 error versus average sigma. Each point is computed from an average over 16k test images for one of the *unseen* distortions and one of 5 levels of shift intensity. Notice (qualitatively and quantitatively) that when the models without ST produce poor results, their uncertainty does *not* correspondingly increase. Our ST helps the model to have a stronger correlation between its uncertainty estimates and error when *tested on unseen distortions*. This indicates that sigma *after ST* can be an effective signal for merging multiple predictions. Note that the predicted mean ("Prediction") *does not change* with ST.

1.3 Method

We explain the technical details of our method below.

Notations: Define $\mathcal{X}$ as the RGB domain, $\mathcal{Y} = \{\mathcal{Y}_j\}_{j=1}^{K}$ as the K intermediate domains, $\mathcal{Z}$ as the desired prediction domain. A single datapoint n from these K domains is denoted as $(x_n, y_{1,n}, \ldots, y_{j,n}, \ldots, y_{K,n}, z_n)$. $\mathcal{F}_{\mathcal{X}\mathcal{Y}}$ is the set of functions that maps the

RGB images to their intermediate domains, $\mathcal{F}_{\mathcal{XY}} = \{f_j : \mathcal{X} \to \mathcal{Y}_j\}_{j=1}^{K}$, and $\mathcal{F}_{\mathcal{YZ}}$ is the set of functions mapping from the intermediate to the target prediction domain, $\mathcal{F}_{\mathcal{YZ}} = \{g_j : \mathcal{Y}_j \to \mathcal{Z}\}_{j=1}^{K}$. Given K predictions of domain $\mathcal{Z}$, they are merged using the function m to get a final single prediction, $m : \{g_j(\mathcal{Y}_j)\}_{j=1}^{K} \to \mathcal{Z}$.

1.3.1 Estimating Per-Path Predictions and Uncertainty

We learn the mappings g_j using a neural network. We model the noise in the predictions made by g_j with a Laplace distribution. Thus, for an input sample $y_{j,n}$, the network outputs two sets of parameters $[\hat{z}_{j,n}, \hat{s}_{j,n}] = g_j(y_{j,n})$ where we set $\hat{s}_{j,n} = \log \hat{b}_{j,n}$ for numerical stability and $\hat{b}_{j,n}$ is the scale parameter of the Laplace distribution. We remove the dependence on j for brevity. This leads to the following negative log-likelihood (NLL) loss for g:

$$\mathcal{L}_{g,NLL} \;=\; \frac{1}{N}\sum_{n=1}^{N} \exp\left(-\hat{s}_n\right)\|\hat{z}_n - z_n\|_1 + \hat{s}_n, \tag{1.1}$$

where N is the number of samples, and z_n is the label for the nth sample. This results in an ℓ_1-norm loss on the errors as opposed to an ℓ_2-norm loss with a Gaussian distribution, which has been shown to improve prediction quality [63, 72]. Finally, the *sigma* is given by $\sqrt{2}\exp(\hat{s}_n)$ and it captures the uncertainty in predictions.

Calibration via Sigma training (ST): Uncertainty estimates under distribution shifts are poorly calibrated [85], i.e. there is a tendency to output a poor prediction with high confidence. This can be seen in Fig. 1.2, "Before sigma training" columns. With a higher noise distortion, the prediction clearly degraded, but the uncertainty estimate did not increase correspondingly. This issue persists even with methods that estimate epistemic uncertainty (Fig. 1.2, "Deep Ensembles" columns) which are meant to detect these shifts.

To mitigate this, we adopt a two-stage training setup where the network trained on in-distribution data is further trained to output high uncertainty outside the training distribution. We denote this step as *sigma training* (ST). Here, $\hat{s}_n$ is trained to learn its maximum likelihood estimator, with the loss denoted as *sigma calibration* (SC). As the goal of this step is to maximize the likelihood by correcting the sigma $\hat{s}_n$, and not the mean $\hat{z}_n$, under a distortion (*dist*), we add a loss term to ensure that $\hat{z}_n$ does not deviate from its predictions at the start of ST, which we define as $\hat{z}_n^0$. We denote this loss as *mean grounding* (MG). Finally, we include the original NLL from Eq. 1.1 on undistorted data (*undist*) to prevent forgetting. This results in the following loss formulation:

$$\mathcal{L}_{g,ST} \;=\; \mathcal{L}_{g,NLL}^{undist} + \alpha_1 \mathcal{L}_{g,MG}^{dist} + \alpha_2 \mathcal{L}_{g,SC}^{dist}, \tag{1.2}$$

where α_1, α_2 controls the weighting between the loss terms. For a given $\hat{z}_n^0$, the MG loss

is defined as the ℓ_1-norm distance between the current prediction and the one at the start of sigma training, i.e. $\mathcal{L}^{dist}_{g,MG} = \|\hat{z}^0_n - \hat{z}_n\|_1$. The SC loss guides the scale parameter towards its maximum likelihood estimate, i.e. $\mathcal{L}^{dist}_{g,SC} = \|\exp(\hat{s}_n) - \arg\min_{\hat{s}_n} \mathcal{L}^{dist}_{g_j,NLL}\|_1 = \|\exp(\hat{s}_n) - |z_n - \hat{z}^0_n|\|_1$.

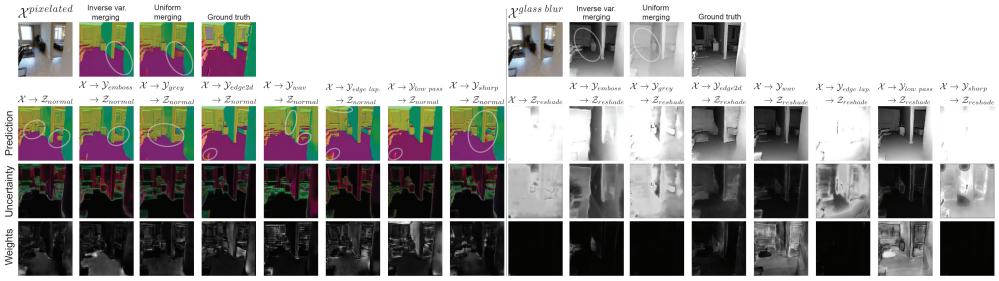

Figure 1.3: **How does the method work?** Each network in each path receives different cues for making a prediction, due to going through different middle domains. **Left**: Given a distorted pixelated query, each path (columns) is affected differently by the distortion, which is reflected in its prediction, uncertainty, and weights (lighter means higher weights/uncertainty). The inverse variance merging uses the weights to assemble a final prediction that is better than each of the individual predictions. (The uncertainties of surface normals look colorful as surface normals domain includes 3 channels, thus there are 3 uncertainty channels.) **Right**: Similarly, for a query with glass blur distortion, the method successfully disregards the degraded predictions and assembles an accurate final prediction. Note that the proposed method (*inverse var. merging*) obtains significantly better results than learning from the RGB directly (leftmost column of each example) which is the most common approach. The quality of the final prediction depends on the following elements: **1.** For each pixel, at least one middle domain is robust against the encountered distortion, and **2.** The uncertainty estimates are well correlated with error, allowing the merger to select regions from the best performing paths. *Uniform merging* does not take into account the uncertainties and consequently lead to worse predictions. The elliptical markers denote sample regions where the merged result is better than all individual predictions.

Following ST, the network outputs sigmas that are highly correlated with error (Fig. 1.2, rightmost plot). Given multiple predictions of the same target domain and their sigma estimates, this allows us to use the latter as a signal for merging to get a single strong prediction (Sec. 1.3.2).

As the objective of ST is to expose the network to inputs with high distortions as opposed to updating the final predicted mean, *any* corruption with high intensity will suffice. The distortions used for ST are not the same distortions as the ones at test time. Please see the appendix Section 2.5 for a detailed study. Furthermore, the experiments (Fig. 1.2, Fig. 1.6, Table 1.1) indicate that sigma clearly generalizes to unseen distortions.

1.3.2 Merging Predictions

After obtaining the set of mappings $\mathcal{F}_{\mathcal{XY}}$ and $\mathcal{F}_{\mathcal{YZ}}$ with the method described above, it remains to merge the predictions coming from multiple paths using a merging function m. We employ an analytical approach given by $m(\{g_j(y_{j,n})\}_{j=1}^{K}) = C\sum_{j=1}^{K}\exp(-2\hat{s}_{j,n})\hat{z}_{j,n}$ where C is a normalizing constant defined as $C = (\sum_{j=1}^{K}\exp(-2\hat{s}_{j,n}))^{-1}$. This performs a straightforward weighting of each pixel in each path by the inverse of its variance [86] which can be done with negligible computational cost. We denoted this as *Inverse variance merging* and will show in Section 1.4.1 that it performs better than other analytical and learning based variants of our method. Algorithm 1 summarizes our training procedure.

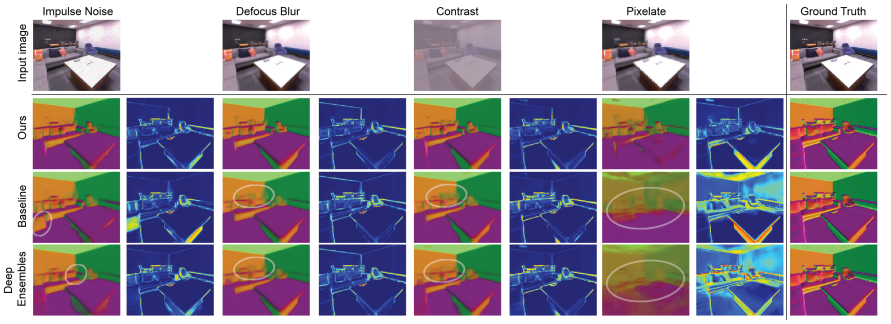

Figure 1.4: **Qualitative prediction results under 4 distribution shifts from Common Corruptions [2]** shown on a sample image from the Replica[87] dataset with shift intensity 3. Each prediction is followed by its corresponding error map. Our method is resistant to distortions compared to the baselines and provides better accuracy *especially over fine-grained regions and sharpness* (see the white markers). Best seen on screen.

Algorithm 1 Summary of the training procedure of our method

Require: Define $f_j \in \mathcal{F}_{\mathcal{XY}}$ and $g_j \in \mathcal{F}_{\mathcal{YZ}}$ $\forall j$.

1: **for** $j = 1 : K$ **do**
2: Train g_j using NLL loss in Eq. 1.1.
3: (Optional) Train g_j using consistency constraints [72]. (Sec. 1.4.1)
4: Perform sigma training over g_j with Eq. 1.2.
5: **end for**
6: Merge the K predictions from the $\mathcal{F}_{\mathcal{YZ}}$ networks using *Inv. var. merging* (Sec. 1.3.2).

A working example. Figure 1.3 illustrates our method with an example. For a given image, each path's prediction, uncertainty, and corresponding weights are shown. For the distorted (pixelated) query in the left, each path reacted differently to the distortion, and the final prediction is obtained by combining individual predictions based on their uncertainties. Similar observations can be made for the glass blurred image in the right, where the method learned weights in a way such that the degraded paths are not

used in the final prediction. We also show the final prediction from a uniform average of each path. While it is better than simply using the direct path ($\mathcal{X} \rightarrow \mathcal{Z}_{normal}$ or $\mathcal{X} \rightarrow \mathcal{Z}_{reshade}$), using the uncertainty estimates as weights results in a notably more accurate prediction.

There are two key elements to the effectiveness of our method. **I.** With a diverse set of middle domains, it is more likely that one of them will be less affected by distortions and returns an accurate prediction. **II.** The error of the prediction correlates well with its corresponding uncertainty estimates, i.e. the uncertainty is low in the region of the image where the prediction is accurate. This allows us to use these uncertainty estimates as a signal to have a final prediction with parts of the image taken from different paths.

1.4 Experiments

We demonstrate that the proposed approach leads to robustness against different *distribution shifts*, over different *datasets*, and different *prediction tasks*. For pixel-wise prediction tasks, we train on the Taskonomy dataset [88]. To evaluate the robustness under corruptions, we report performance under Common Corruptions [2] and adversarial perturbations [9, 10, 11]. To evaluate against dataset shifts, we report on Replica [87] and Habitat [89] datasets. For classification, we train on ImageNet [90], CIFAR [91], and evaluate on ImageNet-C and CIFAR-C [2]. Please see the appendix for more extensive qualitative results.

1.4.1 Evaluations on Pixel-Wise Prediction Tasks

Training dataset: We use Taskonomy [88] as our training dataset which includes 4 million real images of indoor scenes with multiple annotations for each image. We report results for *surface normals*, *depth (zbuffer)*, and *reshading* prediction, as popular target domains.

Middle Domains: From the RGB images we extract *2D edges*, *Laplace edges*, *greyscale*, *embossed*, *low-pass filtered*, *sharpened*, and *wavelet* images as the middle domains (detailed definitions can be found in the appendix). These middle domains are commonly used for low-level image processing tasks with negligible computation cost [92, 93] and do not need any supervision. The performance was not sensitive to the choice of middle domains as the method consistently outperforms baselines and improves with more middle domains (Sec. 1.4.1, Fig. 1.7).

Evaluation datasets: Our goal is to have test data that has a distribution shift from the training data to evaluate the robustness of our method. All the results are reported on the test set of the following datasets:

Taskonomy with Common Corruptions [2]: We apply the Common Corruptions on the test set of Taskonomy. They include all corruptions except outdoor corruptions

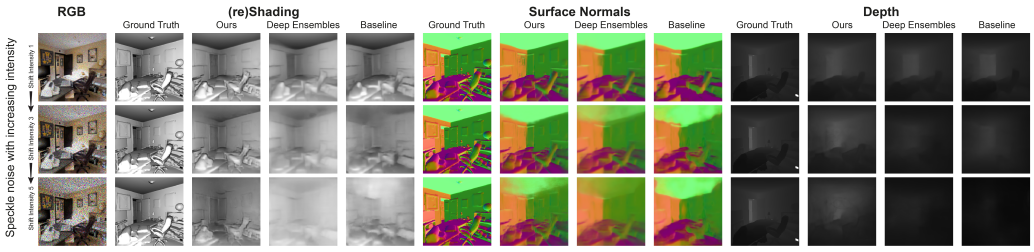

Figure 1.5: **Qualitative results under distribution shifts** for reshading, surface normals, and depth predictions. Each row shows the predictions from a query image from the Taskonomy test set under increasing speckle noise. Our method degrades less than the other baselines, demonstrating the effectiveness of using different cues to obtain a robust prediction. Notable improvements in the accuracy can be seen *especially in fine-grained regions.*

(snow, frost, fog) and the ones that change the geometry of the scene (elastic transform, motion, and zoom blur). We exclude Gaussian noise and blur from evaluations as they were used for ST, to keep training and testing fully separate. Visualizations of a subset of distortions are shown in Figure 1.4 and for all severities in the appendix .

Taskonomy with Adversarial corruptions [9, 10, 11]: We generate adversarial examples using Iterative-Fast Gradient Sign Method (I-FGSM) [10].

Other datasets: Replica [87] consists of 1227 images from high quality 3D reconstructions of indoor scenes. Similar to Taskonomy, we also apply common corruptions on these images. Habitat [89] consists of 1116 images from mesh renderings with a substantial shift from Taskonomy. We test on both datasets without fine-tuning (see appendix).

Training details: All networks for our method and baselines use the same UNet backbone architecture [94] and were trained with AMSGrad [95]. We used a learning rate of 5×10^{-4}, weight decay of 2×10^{-6}, and batch size of 64. The upsampling blocks of all networks resize the activation maps using bilinear interpolation.

We also augment the network training with “cross-task consistency constraints" (X-TC) [72] for generally better results, but this is not a fundamental requirement (ablation results provided in Sec. 1.4.1). We follow [72] and apply non-probabilistic perceptual losses on the predicted mean.

Baselines: We evaluate the following baselines. They are trained with NLL loss (Eq. 1.1), i.e. the models output both mean and sigma.

Baseline UNet: It is a single network that maps from RGB to the target domain without going through a middle domain (i.e. direct). This is the main baseline.

Multi-domain baseline: It is a network model with *RGB* image *and* all middle domains as inputs. Since this model is not *forced* to use different middle domains as opposed to

the proposed method, it reveals if learning from middle domains needs to be explicit and distributed.

Multi-task baseline: It is a single model that maps from *RGB* to *depth*, *reshading*, and *normals*. This is to reveal if learning additional tasks improved robustness.

Data augmentation baselines: We consider a baseline UNet adversarially trained to defend against I-FGSM attacks with $\epsilon = (0, 16]$. This baseline shows how well adversarial robustness translates to non-adversarial distortions. We also include style augmentation [96] as another baseline, which has been shown to reduce the texture bias that are less robust than shape cues.

Blind guess is a single prediction that captures the overall statistics of the domain, i.e. it returns the best guess of what the prediction should be independent of the input. Hence, it shows what can be learned from general dataset regularities (further details are in the appendix).

Deep ensembles [20] creates an ensemble by training the same exact networks with different initializations. Although there are recent papers proposing new ways to enforce diversity in ensembles, their improvement in performance against deep ensembles has not been found significant under non-adversarial shifts [97, 21]. Thus, deep ensembles remains the most relevant ensemble baseline. We use the same number of paths, i.e. ensemble components, as in our method. The predictions from each path are weighted equally to attain the final prediction. This baseline reveals if learning from different cues yields diverse predictions that results in a stronger final estimator.

Cross-domain ensemble setups evaluated: We evaluate several variants of our merging method. In all variants, different paths goes through different middle domain to produce a prediction along with one path being the *direct* prediction. They are then merged into the final prediction. We show the proposed analytical merging is superior to others.

Inverse variance merging: Each path's prediction is weighted inversely proportional to its variance, as proposed in Section 1.3.2.

Uniform merging: A simplified merging where each path is weighted equally, i.e. uncertainty is not used.

Network merging: A neural network is used to merge the predictions. Specifically, we consider a stacking model [98] that learns the final predictions given the outputs from each path and models the final output as a mixture of Laplacians. It has the advantage that the loss is over the entire image, thus, taking into account its spatial structure (see the appendix for details).

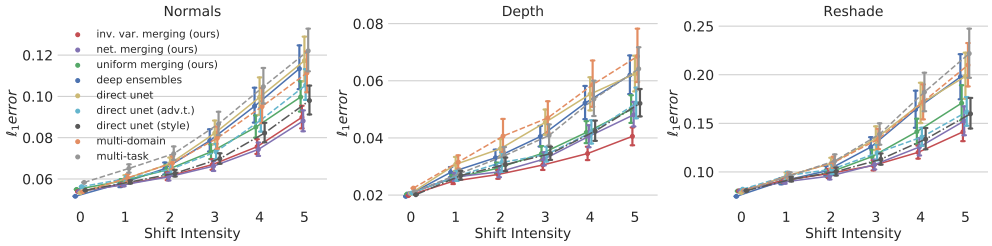

Figure 1.6: **Quantitative robustness evaluations using Common Corruption distortions applied on Taskonomy test set: Average ℓ_1 errors over 11 *unseen* distortions.** Our main method *inv. var. merging*, and frequently its simplified variant *uniform merging* and *network merging*, are more robust against shifts compared to the baselines. Error bars indicate one 'standard error' from the mean (via bootstrapping). Plots for additional perceptual error metrics and individual distortions are provided in the appendix.

	Normal				Reshade				Depth			
Methodε	2	4	8	16	2	4	8	16	2	4	8	16
Baseline UNet	8.23	11.53	13.03	14.37	17.92	22.78	27.26	34.40	5.50	6.76	8.36	9.80
Deep ensembles	**7.49**	11.13	13.36	15.65	15.66	21.95	27.75	34.98	5.45	6.68	8.27	10.52
Inv. var. merging	7.60	**8.89**	**10.40**	**12.77**	**15.56**	**16.55**	**18.93**	**22.01**	**4.94**	**4.99**	**5.93**	**6.75**
Adv. T. (lower bound error)	5.78	5.74	5.45	5.53	9.39	8.98	8.07	8.20	2.23	2.27	2.39	2.74

Table 1.1: **Robustness against adversarial corruptions.** ℓ_1 errors for surface normals, reshade, and depth under adversarial attacks are reported. (Lower is better. Errors are multiplied by 100 for readability.) The proposed method significantly improves robustness against I-FGSM [10] based attacks *without adversarial training*, compared to the baselines. The last row shows the error for a model that has undergone adversarial training [11] with the *same attacks as those evaluated at test time*, hence it gives a lower bound on the error (see supplementary for additional details).

Robustness to Common Corruptions

Figures 1.4 and 1.5 show the qualitative results of our method against the baselines. Performance under various distortions is demonstrated in Figure 1.4 for the surface normals predictions of a sample image from Replica dataset. The proposed method consistently outperforms the baselines and provides more accurate predictions especially in fine-grained regions. This is further supported by quantitative results in Figure 1.6 where the ℓ_1 error over these distortions are notably lower for the proposed method compared to the baselines in all three target domains and shift intensities.

Among the evaluated baselines the data augmentation methods are the most competitive, e.g. adversarial robustness partially transferred to non-adversarial distortions, though *inverse variance merging* performs notably better.

We also observe *inverse variance merging* does much better than *uniform merging* and also better or comparable to *network merging* (Fig. 1.6) despite being simpler, more lightweight, and interpretable. Moreover, it does not demand fixing the number of paths

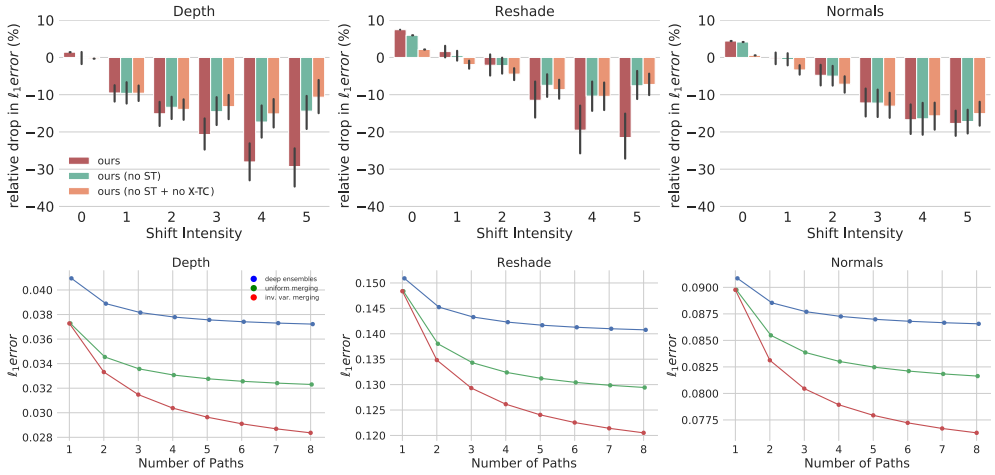

Figure 1.7: **Effect of sigma and/or consistency training. (Top row)** The plots show the relative change in ℓ_1 error compared to deep ensembles (i.e. negative means outperforming deep ensembles). The proposed method outperforms deep ensembles under distribution shifts even without ST and X-TC (consistency).
Robustness as a function of number of paths. (Bottom row) The plots show the average ℓ_1 error as we increase the number of paths (or ensemble components in the case of deep ensembles). The proposed method (*inv. var. merging*) and its simplified variant (*uniform merging*) consistently outperforms deep ensembles which plateaus much faster.

beforehand (unlike *network merging*), thus the number of paths can be decided by taking computational considerations into account on the fly.

Robustness to Adversarial Attacks

We demonstrate the effectiveness of the proposed method under adversarial attacks. The attacks are generated by I-FGSM. Following [10], we use attack strengths $\epsilon = [2, 4, 8, 16]$, with the number of iterations given by $N = \min(4 + \epsilon, 1.25\epsilon)$. The results are shown in Table 1.1. Neither our method nor the baselines utilize explicit adversarial defense mechanisms – while deep ensembles perform nearly as poorly as baseline UNet, the proposed method performs significantly better. This indicates that *using middle domains* promotes ensemble diversity in a way that *makes it more challenging to create one attack that fools all paths simultaneously*, hence this approach can be a promising remedy for adversarial attacks as well. Moreover, the proposed method also outperforms *Uniform merging* (see the appendix for the results) which does not use uncertainty estimates during merging. This indicates that the additional uncertainty output did not create an additional avenue for attack that I-FGSM could exploit.

Note that we do not obfuscate gradients by e.g. intentionally making certain operations non-differentiable, or using stochastic transforms [99]. The analytical operations to obtain the middle domains are deterministic and differentiable.

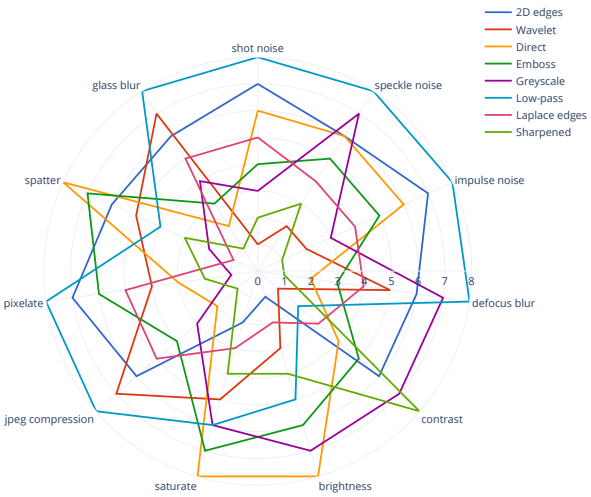

Figure 1.8: **Importance of each middle domain for different distortions.** The chart shows the order of the best performing paths for surface normal prediction for different distortions, with 8 denoting the most important and 1 the least important path. The plot shows, for instance, "noise" distortions benefited most from the *low-pass* middle domain while "contrast" distortion benefited most from the *sharpened* middle domain.

Additional Ablation Studies

Contribution of ST/X-TC: To quantify the contribution of each stage of training to the overall robustness of our setup, we study the performance of our proposed method without sigma training (ST) and/or cross-task consistency constraints (X-TC) in the first row of Figure 1.7 (and appendix). Our method, with or without ST or X-TC constraints, still outperformed deep ensembles as almost all bars are below the 0 line.

In Section 2.3 of the appendix, we compare the effect of equipping deep ensembles with ST and X-TC, and perform uniform and inverse variance merging. Thus, the only difference with our method is the use of middle domains. Our method still outperforms.

Robustness vs number of employed paths: In Figure 1.7, we investigate performance as a function of number of paths. Each point shows the average ℓ_1 error of *all possible combinations* for a given number of paths. Although all methods improve as more paths are added, our proposed methods has a much steeper downward trend than deep ensembles and our uniform merging variant, indicating that performance gap increases with more paths.

Sensitivity to choice of middle domains: Figure 1.7 also shows that the performance of our method is not sensitive to a particular set of middle domains. For a fixed number of paths n, our method outperforms deep ensembles for all possible combinations of n paths on average.

Path importance: We show the importance of each middle domain for the final prediction under each distortion in Fig. 1.8. For number of paths $n = 1, \ldots, 8$, we compute the set of best performing paths, i.e. the set of n paths with the lowest ℓ_1 error, denoted by $P_n = \{p_{(i)}\}_{i=1}^{n}$. The n^{th} best performing path is given by $P_n \backslash P_{n-1}$. The plot shows different paths indeed react differently to a given corruption, e.g. noise distortions substantially benefited from *low-pass*, while contrast distortion did not – thus the benefit

is not attributed to one or few middle domains under all distortions.

1.4.2 Robustness of Sigmas to Distribution Shifts

We have showed that our method returns predictions that are robust under a wide range of distribution shifts. Are our predicted uncertainties also able to generalize under distribution shifts, i.e. do we get high uncertainties when predictions get worse? To investigate this, we consider epistemic uncertainty which is used to capture the model's uncertainty and is an indicator of distribution shifts [63]. The left plot of Figure 1.9 shows a scatter plot of average epistemic uncertainties against error and the right shows the average error for all epistemic sigma values less than a threshold τ. The predicted uncertainties from deep ensembles initially increase with error but do not increase past 0.15 sigma despite an increase in error, thus is overconfident, while our method shows an increasing trend.

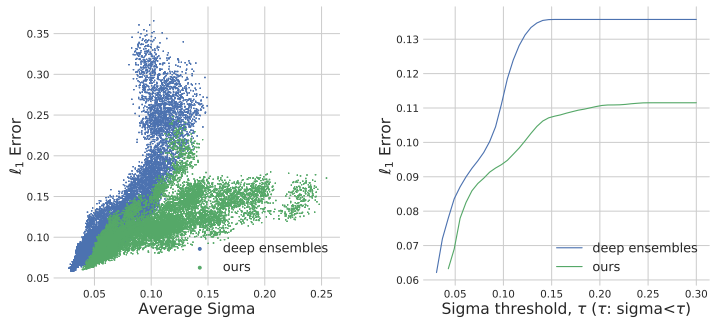

Figure 1.9: **Generalization of predicted uncertainties.** We compute the average ℓ_1 error and epistemic uncertainties for 11 unseen distortions and 5 levels of intensities for Reshade. Each point is an average over 64 images. Our method is able to return *high uncertainty when predictions are poor*.

Performance on undistorted data

In order to demonstrate that the robustness of our method on out-of-distribution data did not come at the cost of degraded performance on in-distribution data, we provide quantitative evaluations on the *undistorted* Taskonomy and Replica datasets in appendix. The results show the performance of our method, *when tested on undistorted data*, is indeed comparable to or better than the methods that are trained to perform well only on undistorted data.

1.4.3 Evaluation on Classification Tasks

The benefits of the proposed method is not limited to regression or dense pixel-wise tasks. We performed an experiment on ImageNet-C to evaluate the robustness against Common Corruptions (Table 1.2). Our method and deep ensembles both use 8 paths with identical ResNet-50 [100] network architecture. In addition, in this experiment our method does a simple averaging of the output probabilities from each path, similar to deep ensembles, and no ST or X-TC training was involved. The superior results show the

Table 1.2: **Robustness on ImageNet-C.** Error on clean and distorted data (mean Corruption Error, mCE). Following [2], the mCE is relative to AlexNet [101]. All methods are trained on clean ImageNet training data. Our method performs noticeably better under distortions compared to deep ensembles and a single model baseline ResNet. See the appendix for a detailed breakdown and additional results on CIFAR.

Method	Clean error	mCE
Baseline ResNet-50	24.37	76.21
Deep ensembles	21.50	70.43
Ours	21.61	67.85

basic value in using a diverse set of middle domains. Similar conclusions were obtained for CIFAR-10-C and CIFAR-100-C datasets (full results in the appendix).

1.5 Conclusion and Discussion

We presented a general framework for making robust predictions based on creating a diverse ensemble of various middle domains. Experiments demonstrated that this approach indeed leads to more robust predictions compared to several baselines.

We also showed that our method is not sensitive to the choice of middle domains (Sec. 1.4.1) or the corruptions used for ST (supplementary). Furthermore, even after equipping deep ensembles with ST and consistency training (Sec. 1.4.1, supplementary), our method still outperforms, confirming the effectiveness of using middle domains.

Below we briefly discuss some of the limitations:

Uncertainty under distribution shift: Our method relies on having reasonable uncertainty estimates (i.e. sigma) in presence of distribution shifts. While we observed sigma training to be helpful for this purpose, and also, uniform merging which does not rely on uncertainty estimates to still outperform the baselines, our method will benefit from better uncertainty estimation techniques.

Choice of middle domains: We adopted a fixed set of middle domains, and, as discussed in Sec. 1.4.1, the final performance was not sensitive to the adopted dictionary. However, *learning* or computationally *selecting* such middle domains with the objective of downstream robustness could be a worthwhile future direction.

Multi-modal distributions: We modeled our individual path outputs with single-modal distributions for convenience and considered multi-modal distributions only at merging step. Allowing for multi-modality in each path's output may further help with ambiguous data points.

Computational cost: While the computational complexity of our method and deep ensembles [20] are virtually the same, the methods based on ensembling generally increase the computational complexity as they involve turning one estimator into multiple. Investigating if the models in the ensemble can be compressed would be worthwhile – especially for our method since the diversity in the ensemble is by structure and owed to adopting different middle-domains, rather than stochasticities that often assume independence among models.

This chapter is based on the paper: T. Yeo*, O. F. Kar*, A. Zamir, *Robustness via Cross-Domain Ensembles,* ICCV 2021. I was the main contributor of this work.

2 Controlled Training Data Generation

2.1 Introduction

In this chapter, we present another robustness mechanism based on generating targeted training data. Large datasets have been a driving force in the progress of deep learning. It has also been shown that training on large and diverse datasets tends to improve robustness [102, 103]. However, *many datasets exhibit biases*, e.g., they are collected from certain geographic regions or with images that tend to be unoccluded and front facing [14, 15, 16]. Neural networks trained on these datasets can return *unreliable predictions* when tested on inputs with distribution shifts e.g., different geographic locations or unusual viewpoints or occlusions. This chapter presents a method to control generative models to mitigate the bias of training datasets.

An example of generative models is text-to-image models that generate an image conditioned on a given text prompt. These models have displayed remarkable generation and compositional capabilities [104, 105]. Several works have adopted these models to generate training data [30, 31, 32]. However, most of these works make use of pre-defined prompt templates. Thus, they are open-loop systems that generate data without any feedback and to some extent require anticipating the kinds of shifts that will be seen at test-time when defining the prompts.

Early works have attempted to control GANs or VAEs to generate targeted training data [33, 34, 35]. These methods create a closed-loop system by making use of feedback from the model that it aims to improve. Our work falls into this category, with the difference that we employ diffusion models for generation. We aim to learn a suitable prompt to generate images using feedback from the loss of a given pre-trained model. To do so, we borrow the idea of textual inversion [106] that aims to learn a word embedding, with placeholder string S^*, that represents a concept e.g., object, individual, style. While

textual inversion learns S^* from a set of images with that concept, our work aims to find S^* that results in generations that *maximize the loss of a given pre-trained model* f. See the left of Fig. 2.1 for how our method works and the right for some examples of S^* found by our method.

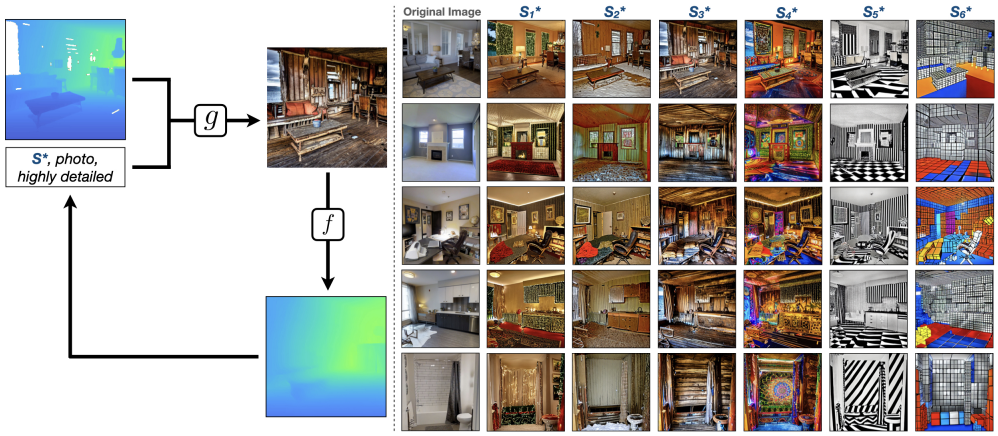

Figure 2.1: **Controlling generative models to generate targeted training data.** *Left:* An overview of how we generate training data. g is a generative model e.g., ControlNet [105] that takes in text prompts and additional conditioning (e.g. depth, segmentation, edge maps), to generate an RGB image. The additional conditioning in this example is depth maps. f is a pre-trained model, in this case, trained to perform depth estimation. We *maximize the loss* of f on the generated data with respect to a word embedding that has the corresponding string S^*. This results in S^* representing a style that is distinctly different from the original training data (see examples on the right). As g takes as input both the text prompt and depth map (as additional conditioning) and as this conditioning is the same as the target task, as long as the generations remain faithful to the condition, *this results in paired data* (RGB image and its corresponding label) that we can use to finetune f. See Sec. 2.3.2 for more details. *Right:* We show the results from optimizing for several S^*. Each S^* represents a distinct concept, and results in generations different from the original training data, shown in the first column. These S^*s *can be applied to any image in the training data*, even those that it was not optimized on. S_1^* seems to represent a festive style, S_2^*, snow, S_3^* transforms the scene into a wooden cabin, and so on. See Sec. 2.4 for more results.

2.2 Related Work

This work focuses on improving model robustness by training on generated data. We give an overview of the relevant topics.

Closed-loop data generation guides the generation process via the loss of a given neural network. [33, 35] control the latent space of GANs or VAEs to generate data that maximizes the loss of the network on the generated data. Similarly, [3] controls a NeRF to generate unusual viewpoints. [107] uses an SVM to first identify the failure modes of a given model, then uses this information to generate training data with a diffusion

model. Our method also employs a diffusion model but controls the generation in the prompt space.

Open-loop data generation uses pre-defined perturbations or prompts to generate data. Recent works that adopted diffusion models to generate training data use pre-defined prompt templates [32, 31, 30] or use a language model to generative variations of a given prompt [31]. These methods, to some extent, require anticipating the kind of data that will be seen at test-time when defining the prompts. In contrast, our method learns the prompt to generate data. Thus, it does not anticipate the data that will be seen at test-time, and the data generation and training can be done end-to-end.

Data augmentation. Data augmentation techniques apply transformations to a given RGB image to increase the diversity of training datasets to improve generalization performance. Traditional augmentations apply transformations such as color jitter, random crop, flipping, and so on to the given image. RandAugment [38], AugMix [36] are ways of combining these augmentations. Mixup [40] and CutMix [39] are proposed augmentations that transform a given image using another image in the dataset. Our method results in images that are more diverse than standard augmentations and more realistic than mixing multiple images. We can also have closed-loop augmentations, i.e., transformations that are guided by the loss of a network. AutoAugment [37] learns the optimal mix of augmentations while adversarial training [11] learns the perturbations to be added to the input.

Controlling diffusion models. Methods like ControlNet [105] and T2I-Adapter [108] adapt a pre-trained diffusion model to allow for additional conditioning e.g., edge, segmentation, and depth maps. We employ these models for generation as it allows us to generate paired data for different tasks, given the labels from an existing dataset. Editing methods aim to modify a given image, either via the prompt [109], masks [110], instructions [111] or inversion of the latent space [112, 113]. In contrast, personalization methods aim to adapt models to a given concept e.g., an object, individual, or style. Popular examples include textual inversion [106] and DreamBooth [114], which aim to find a token to represent a concept given several images of that concept. The former freezes the diffusion model, while the latter finetunes it. Extensions of these works learn to represent multiple concepts [115, 116]. While our goal is not personalized generation, we borrow the idea of optimizing for a token to represent a concept.

2.3 Method

Notations. Let $\mathcal{X}$ be the input image domain, and $\mathcal{Y}$ the target domain. x and y are samples from these domains. We use $f : \mathcal{X} \to \mathcal{Y}$ to denote the pre-trained model. $\mathcal{L}(f(x), y)$ is the training loss and $\mathcal{D}$ the training dataset for f. Let g denote the generative model that takes in text prompt p and optionally, conditioning z, i.e., $g : \mathcal{P} \times \mathcal{Z} \to \mathcal{X}$.

2.3.1 Preliminaries on diffusion models and textual inversion

Diffusion models [117] learn the data distribution by iteratively denoising a normally distributed variable. These models either operate on the image [118, 119, 120] or to reduce computation cost, the latent space [104]. Diffusion models can also perform generation conditioned on inputs such as text prompts, p, i.e., text-to-image (T2I) generation. One example of a T2I latent diffusion model is Stable Diffusion [104]. It consists of an autoencoder that is used to map images into a latent code ($z = \mathcal{E}(x)$), and back to the original image ($\tilde{x} = \mathcal{D}(z)$) and a diffusion model that is trained to perform denoising in the latent space. The diffusion model is trained with the following loss: $\mathcal{L}_{LDM} := \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}[||\epsilon - \epsilon_\theta(z_t, t, p)||_2^2]$, where z_t is the latent noised to time t, ϵ is the unscaled noised sample, ϵ_θ is the denoising network.

ControlNet [105] adds additional layers to the Stable Diffusion model to allow for additional conditioning, in addition to text prompts. This conditioning can be depth, edge, segmentation maps, etc. We employ ControlNet for generation as it allows us to condition on the *labels of existing datasets* e.g., depth or segmentation maps, to generate RGB images. This results in paired data i.e., RGB images with their corresponding labels, that can be used for training. See Sec. 2.3.2 for further discussion.

Textual inversion [106] aims to learn a concept e.g. style, object, represented by a placeholder word S^*, given a set of images. Each word (or sub-word) in the prompt is converted to a token, after which, an embedding vector, v. More concretely, textual inversion aims to learn an embedding vector, v^*, by optimizing $\mathcal{L}_{LDM}$ while keeping ϵ_θ fixed. It learns an embedding v^* to reconstruct the given set of images. The optimization objective is given by, $v^* = \arg\min_v \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}[||\epsilon - \epsilon_\theta(z_t, t, p)||_2^2]$.

2.3.2 Optimizing for a S^* that maximizes loss

Our goal is to generate training data that reflects the *failure modes* of a given pre-trained model, f. To do this, we *optimize for inputs that maximize the loss* of f. There are several ways of attaining these inputs. We can e.g., optimize for an imperceptible perturbation, constrained by an ℓ_p norm, to be added to the input image [9], select pre-defined groups in the training data [51] or optimize the latent space of generative models [33, 35]. To generate data, we adopt T2I diffusion models [104, 105] as they exhibit powerful generative capabilities. We then perform this *optimization in the space of text prompts* as it can potentially result in more interpretable and realistic generations.

Similar to textual inversion [106], we aim to *learn a v^* that represents a concept.* However, instead of the objective of reconstructing a set of images, we aim to fool a given pre-trained model, f. As described above, ControlNet is able to take domains such as depth as conditioning, thus, we can *generate data conditioned on the depth labels from the original training dataset $\mathcal{D}$*. More concretely, we use the label y, as conditioning while optimizing for v^* to generate an RGB image x. The optimization objective is given by,

$v^* = \arg\max_v \mathbb{E}_{x \sim g(y,p(v))} \mathcal{L}(f(x), y)$. g in this case is ControlNet and encapsulates the autoencoder, $\mathcal{D} \odot \mathcal{E}$, text encoder, and conditional denoising diffusion model, ϵ_θ.

Performing the above optimization gives us a v^*, which corresponds to a placeholder word S^*, which we can then use to generate data with a prompt like "S^*" and condition y. Furthermore, we observe the following: **1.** the resulting generations tend to follow the conditioning well (see Sec. 2.3.3 for more details), **2.** S^* *can be applied on any image* in the dataset, even those that were not optimized over. Thus, generating data with S^* and labels y allows us to attain paired data i.e., $(g(y, p), y)$ for dense tasks e.g., depth, semantic segmentation. These generated data are also distinct from the training data and result in a much higher loss of f (see Fig. 2.3). Thus, they represent the failure modes of f and we finetune f on this generated data to improve its robustness (see Sec. 2.4.2 for results).

2.3.3 Constraining the optimization of S^*

The above formulation results in the generation of diverse data that is distinctly different from the original training data (see Fig. 2.1, left, Fig. 2.3, 5th column). While most generations are faithful to the conditioning, it is possible that the optimization returns a degenerate solution i.e., a blank image as that would successfully maximize the loss. Thus, we also consider several modifications to the above formulation.

Image to image generation [121]. Instead of generating data from randomly sampled noise, this technique encodes a given image into a latent representation. Denoising is then done on this latent representation and results in generations that are visually closer to the given image. While this technique results in generations that maximize the loss of f and is less likely to return a collapsed generation, it comes at the expense of lower diversity in the generations (see Fig. 2.2). Thus, the following experiments did not adopt this technique. Balancing the diversity of the generations with its faithfulness is left to future work.

Optimizing for tokens [122]. Instead of optimizing for an embedding v^*, we can *optimize for tokens* from the existing vocabulary, similar to [122]. This constrains the possible generations to make use of the limited number of existing tokens, and can potentially result in generations that are more interpretable. However, optimizing for up to 128 tokens at the same time did not seem to result in a significant decrease in the loss.

2.4 Experiments

In this section, we will demonstrate that our proposed method is able to generate training data that can be used to improve the robustness of a pre-trained model.

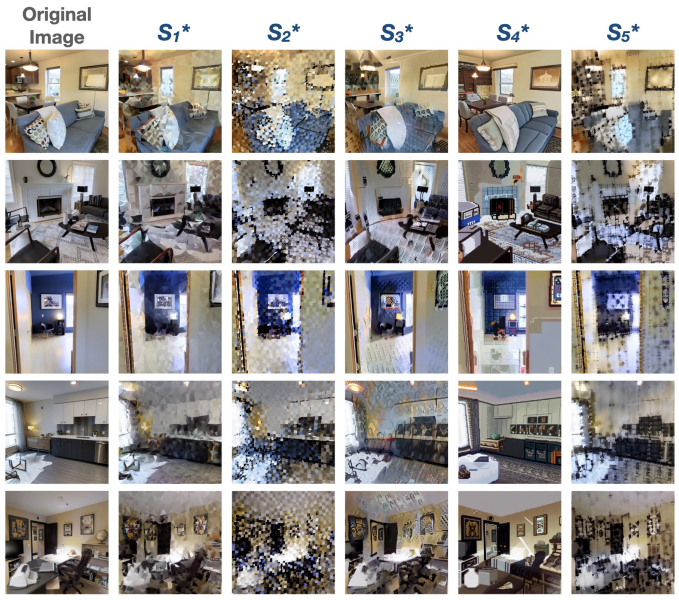

Figure 2.2: **Optimizing for S^* with image to image generations.** We show the results form optimizing for several S^* with image to image (img2img) generation. The generations are now more similar to the original image, shown in the first column, thus, less diverse than without img2img (see Fig. 2.1).

2.4.1 Experimental setup

We consider the depth prediction task, with the model f trained on the Taskonomy dataset [88, 72]. As mentioned in Sec. 2.3.2, we optimize for n embeddings, $\{v_i^*\}_{i=1}^n$. For a datapoint, (x, y), in the Taskonomy dataset, we randomly sample an embedding vector, v_i^*, with the corresponding placeholder word S_i^*, and generate an RGB image with the prompt "S_i^*" and the conditioning y. We then finetune f on the generated data, for a range of dataset sizes, 15k, 30k, and 60k datapoints.

Baselines. We evaluate the following baselines. They all involve fine-tune the same model f but on different datasets:

Control: This fine-tunes f on the original training data. This baseline is to ensure that the difference in performance is due to the generated data, rather than e.g., longer training or a change in optimization hyperparameters.

Prompt "room": As the Taskonomy dataset consists of indoor images from mostly residential buildings, we generate data with the prompt "room".

Prompt "": Here we do not constrain the generations to look like rooms. Thus, we can potentially get generations that although still resemble indoor scenes but are not living spaces e.g., an indoor pool, cafe, etc.

Sampled from $\mathcal{N}(\mu_{emb}, \sigma_{emb})$: In our proposed method, we optimize for n embedding vectors. Here, we fit a distribution to the embeddings in the vocabulary and sample n embeddings to be used in the data generation. Thus, this baseline controls for the number of embedding vectors used in the data generation and can potentially result in more diverse data.

For all generations, we use ControlNet version 1.0 with depth conditioning [105]. The text prompt always includes "photorealistic, highly detailed". The proposed method and

baselines are fine-tuned on the same datapoints i.e., all generations use the same set of labels, for conditioning and the control baseline uses the same datapoints from the original dataset.

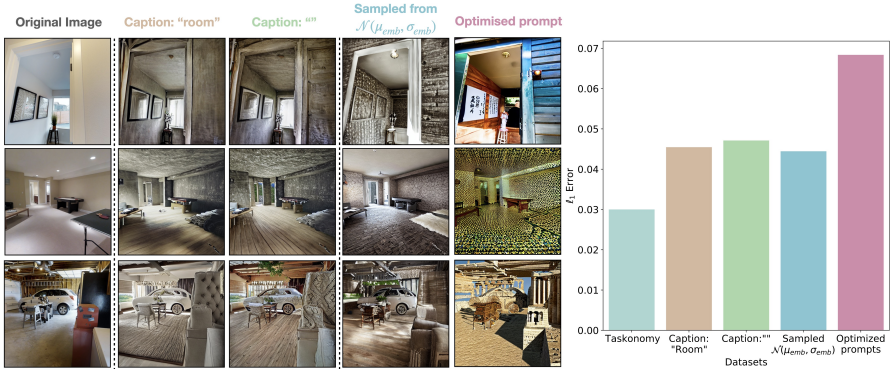

Figure 2.3: **A comparison of generations with different prompts.** *Left:* A comparison of the generated data for the baselines and our proposed method. Although the baselines (columns 2-4) result in generations that are different from the original data (first column), they tend to be similar to each other. Optimizing for word embeddings results in styles that are clearly distinct from the original data. Furthermore, as we optimized for several different word embeddings, we were able to generate different styles for each image, as seen in the different rows. Note that sampling different embeddings (second last column) does not result in generations that are as diverse. *Right:* The ℓ_1 error of f on the different generated data. Evaluating on data generated from optimized word embedding results in the highest loss.

2.4.2 Fine-tuning on generated data

Comparing the generated images with different prompts. Figure 2.3 shows the results of the generations for the baseline and proposed method. The generations from the baselines (columns 2-4), although they result in styles different from that of the original training data, tend to have similar styles, while those from our proposed method (last column) exhibit more distinct and diverse styles. Furthermore, the generations seem to keep the geometry of the scene. Thus, it allows us to use the generated RGB images and depth conditioning as training data. The loss of f on the generated data is shown on the right of Fig. 2.3. The generated data for the baselines results in a higher loss compared to that on the original training data and optimizing for prompts results in the highest loss.

Performance on OOD data. We evaluate our method and the baselines i.e., after finetuning on their respective generated datasets, qualitatively on random query images (Fig. 2.4, left) and quantitively on the Taskonomy dataset under Common Corruptions and Replica dataset (Fig. 2.4, right). Our proposed method, shown in the last row, returns more accurate predictions in general and even on outdoor scenes. Furthermore, under common corruptions and cross-dataset shifts, when fine-tuned on dataset sizes 30k

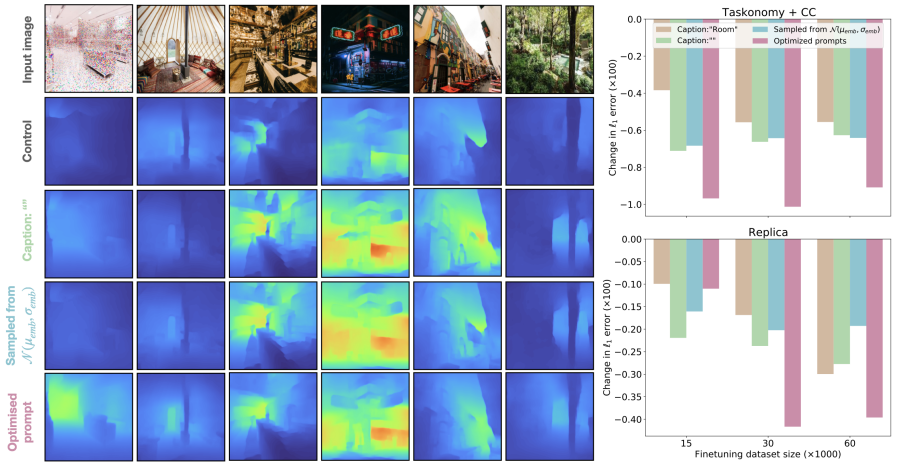

Figure 2.4: **Results under distribution shifts.** *Left:* Qualitative results of the models fine-tuned on the different generated data. The input images are query images from the internet and the following rows show the predictions from the different fine-tuned models. Our proposed method, shown in the last row, results in more robust predictions, even on outdoor scenes. *Right:* ℓ_1 error relative to the control baseline (multiplied by 100 for readability), lower is better. The different models fine-tuned on different datasets and on different dataset sizes, are evaluated on Taskonomy under Common Corruptions (averaged over all corruptions and severity levels) and the Replica dataset. In both cases, finetuning on generated data from optimized prompts tends to result in better performance.

or larger, our method shows clear improvements.

2.5 Discussion on target domain informed vs uninformed generation

The proposed framework generates data that is *uninformed about the target domain* i.e., the training objective is to generate data that maximizes the loss of f and does not make use of any knowledge of the target domain e.g., unlabelled data, domain description. How well would this framework do if we are interested in a certain target domain?

We experimented with the iWildCam dataset [123] from the WILDS benchmark [18]. The dataset consists of camera trap images and the task is multi-species classification. As the images are taken in the wild, the animals can be highly occluded and have unusual poses or perspectives. Thus, as it may be challenging to generate animals with unusual poses, we generate data using an in-painting diffusion model[1] [104] to keep the region of the image with the animal. Thus, we also make use of the bounding boxes of the animals from [124] for the generation. We run the optimization process described in 2.3.2 to get several S^*. The generations from these S^* are shown in Fig. 2.5. We use the ERM model

[1] https://huggingface.co/runwayml/stable-diffusion-inpainting

from [18] as the pre-trained model, f.

The generations are able to fool the model, however, they are not relevant to the target domain as the target domain consists of animals in the wild. The distribution shift in the target domain is due to different camera traps i.e., there can be shifts due to different illuminations, camera parameters, background, and so on. Thus, while it is possible to constrain our generation, e.g., by restricting the search space of the optimization to only natural domains, alternative approaches may be easier. For example, [125] generates descriptions of a dataset's domains and uses these descriptions to edit the training images. Thus, as these images are edited with text prompts to e.g., change the background from a grassy field to a body of water, they are more likely to look like images taken in the wild.

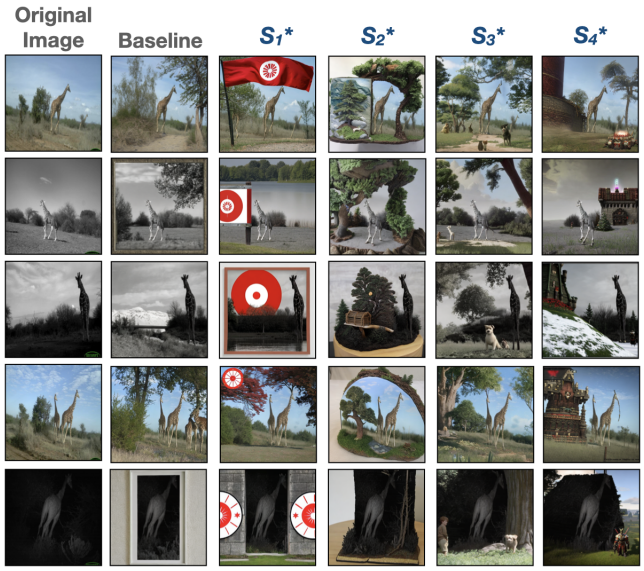

Figure 2.5: **Optimizing for S^* on the iWildCam dataset.** We show the results from optimizing for several S^* on the iWildCam dataset for multi-species classification. These generations, while they are able to fool the pre-trained model, are not relevant to the target domain.

2.6 Conclusion

We presented a way of controlling diffusion models to generate targeted training data. This results in generations that are diverse and distinctly different from the original training data (see Fig. 2.1, 2.3). Fine-tuning on this data results in improved robustness compared to the baselines.

Below we briefly discuss some of the limitations:

Computational cost: Performing the optimization described in Sec. 2.3.2 is expensive, compared to e.g., generating adversarial attacks by optimizing for additive perturbations. The optimization of each S^* takes about an hour on 3 V100s. This is partly due to the nature of diffusion models i.e., they require several denoising iterations to generate an image. Learning to predict an S^* given the state of the model f is an interesting future direction.

Fixed conditioning: In this work we assume access to labels, that can be used as conditioning inputs to ControlNet to generate new RGB images. However, there can also be shifts in the labels when e.g., going from indoor to outdoor scenes or changes in viewpoint or field of view. Learning a generative model to predict the labels/conditioning from e.g. noise or meta-data like camera parameters or proportion of each class in a scene [126] would allow us to control both the label and RGB input domain.

Curated demonstrations: This framework can potentially address other types of distribution shifts, e.g., datasets with spurious correlations, and reveal the biases of different models and training data. Attaining such demonstrations is a worthwhile future direction.

This chapter is based on a soon-to-be-published work: T. Yeo, A. Atanov, A. Alekseev, H. Benoit, R. Ray, P. Esmaeil, A. Zamir, *Controlled Training Data Generation with Diffusion Models for Robustness.* I was the main contributor of this work.

Adaptation Mechanisms Part II

3 Fast adaptation using test-time feedback

3.1 Introduction

The previous chapters presented *training-time* mechanisms. These mechanisms attempt to anticipate the distribution shifts that can occur at test-time. As distribution shifts that occur in the real world are *numerous* and *unpredictable* and the models are frozen at test-time, training-time mechanisms have inherent limitations. This is the main motivation behind *test-time* adaptation methods, which instead aim to adapt to such shifts as they occur. In other words, these methods choose adaptation over anticipation. In this chapter, we propose a test-time adaptation framework that aims to perform *efficient* adaptation of a main network using a feedback signal.

One can consider performing test-time training for this purpose, similar to previous works [53, 127, 128]. While this can successfully adapt a network, it is unnecessarily *inefficient* as it does not make use of the learnable regularities in the adaptation process, and consequently, is unconducive for real-world applications. It also results in a *rigid* framework as the update mechanism is fixed to be the same as the training process of neural networks (SGD). We show this process can be effectively amortized using a learning-based feed-forward controller network, which yields orders of magnitude *faster* results (See Fig. 3.1, Sec. 3.4.3). In addition, it provides *flexibility* advantages as the controller is implemented using a neural network and can be engineered to include arbitrary inductive biases and desired features.

3.2 Related Work

Our work focuses on how to adapt a neural network in an efficient way at test-time on a range tasks and adaptation signals. We give an overview of relevant topics.

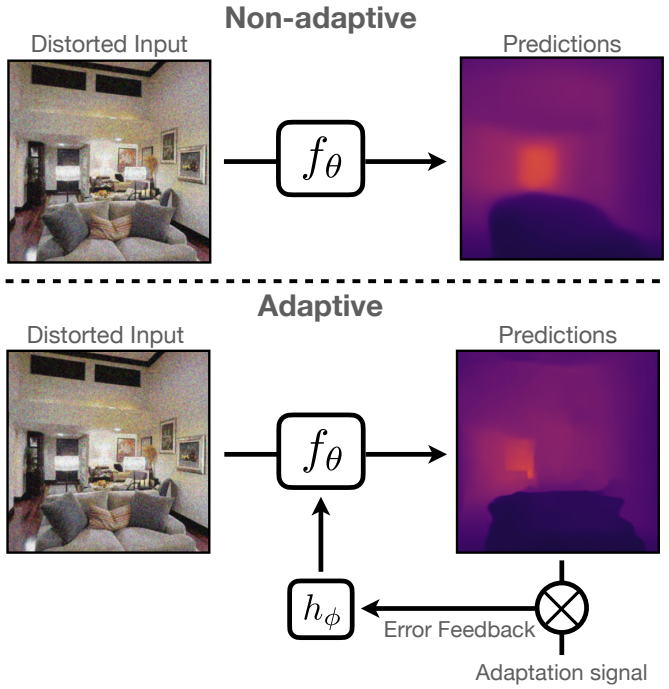

Figure 3.1: **Adaptive vs non-adaptive neural network pipelines.** ***Top:*** In order to be robust, non-adaptive methods include training-time interventions that *anticipate and counter* the distribution shifts that will occur at test-time (e.g., via data augmentation). The learned model, f_θ, is frozen at test-time, thus upon encountering an out-of-distribution input, its predictions may collapse. ***Bottom:*** Adaptive methods create a *closed loop* and use an *adaptation signal* at test-time. The adaptation signal is a quantity that can be computed at test-time from the environment. h_ϕ acts as a "controller" by taking in an error feedback, computed from the adaptation signal and model predictions, to adapt f_θ accordingly. It can be implemented as a **(i)** standard optimizer (e.g., using SGD) or **(ii)** neural network. The former is equivalent to test-time optimization (TTO), while the latter aims to *amortize* the optimization process, by training a controller network to adapt f_θ – thus, it can be more efficient and powerful. In this chapter, we study the latter approach and show its efficiency and flexibility.

Robustness methods *anticipate* the distribution shift that can occur and incorporate inductive biases into the model to help it generalize. Popular methods include data augmentation [11, 40, 129, 36, 39, 42, 7, 4], self-/pre-training [130, 131, 132, 133, 134, 135, 136], architectural changes [137, 138, 139, 140, 141] or ensembling [20, 85, 142, 26, 143, 144]. We focus on adaptation mechanisms and identifying practical adaptation signals that can be used at *test-time*.

Conditioning methods use *auxiliary inputs* to adapt a model. Some examples include using HyperNetworks [145, 146] or cross-attention [104, 147]. A popular method that has been adopted in different problem settings, e.g., style transfer [148, 149, 150], few-shot learning [151, 152, 153, 154, 155], is performing feature-wise modulation [156, 157]. It

involves training a model to use the auxiliary information to predict affine transformation parameters that will be applied to the features of the target model. Our formulation can be viewed to be a particular form of conditioning, and we show it results in a framework that is expressive, efficient, and generalizable.

Amortized optimization methods make use of learning to improve (e.g., speed-up) the solution of optimization problems, particularly for settings that require repeatedly solving similar instances of the same underlying problem [158, 159, 160, 161, 162, 163, 164, 165, 166]. Fully amortized optimization methods model the shared structure between past instances of solved problems to regress the solution to a new problem [145, 167, 168]. As adapting to distribution shifts can be cast as solving an optimization problem at test-time, our method can be interpreted as an amortized solution.

Test-time adaptation methods for geometric tasks. Many existing frameworks, especially in geometric tasks such as aligning a 3D object model with an image of it, in effect instantiate a task-specific case of closed-loop optimization for each image [164, 161, 169]. Common sources of their adaptation quantity include sensor data [170, 171, 172, 173, 174, 175], structure from motion (SFM) [176, 177], motion [178], and photometric and multi-view consistency constraints (MVC) [179, 180]. Many of the latter methods often focus on depth prediction and they introduce losses that are task-specific, e.g., [176] optimize a photometric consistency loss. We differ by aiming to investigate a more general framework for test-time adaptation that can be applied to several tasks. For MVC, while we adopt the same losses as [179], we show under collapsed predictions, optimizing only MVC constraints is not sufficient for recovering predictions; depth predictions need to be adapted and this can be done efficiently using our proposed framework (see Sec. 3.4.3).

Test-time adaptation methods for semantic tasks. Most of these works involve optimizing a self-supervised objective at test-time [181, 53, 182, 128, 183, 127, 184, 54, 185]. They differ in the choice of self-supervised objectives, e.g., prediction entropy [53], mutual information [54], and parameters optimized [184]. However, as we will discuss in Sec. 3.3.2, and as shown by [184, 128, 186], existing methods can ***fail silently***, i.e. successful optimization of the adaptation signal loss does not necessarily result in better performance on the target task. We aim to have a more general method and also show that using proper adaptation signals results in improved performance.

Weak supervision for semantic tasks uses imperfect, e.g., sparse and noisy supervision, for learning. In the case of semantic segmentation, examples include scribbles [187] and sparse annotations [188, 189, 190, 191, 192]. For classification, coarse labels are employed in different works [193, 194]. We aim to have a more general method and adopt these as test-time adaptation signals. Further, we show that self-supervised vision backbones, e.g., DINO [195], can also be used to generate such signals and are useful for adaptation (See Sec. 3.3.2).

Multi-modal frameworks are models that can use the information from multiple sources, e.g., RGB image, text, audio, etc., [196, 197, 198, 199, 200, 201, 133, 202, 203, 204]. Schematically, our method has similarities to multi-modal learning (as many amortized optimization methods do) since it simultaneously uses an input RGB image and an adaptation signal. The main distinction is that our method implements a particular process toward adapting a network to a shift using an adaptation signal from the environment – as opposed to a generic multi-modal learning.

3.3 Method

In Fig. 3.1, we schematically compared methods that incorporate robustness mechanisms at training-time (thus anticipating the distribution shift) with those that adapt to shifts at test-time. Our focus is on the latter. In this section, we first discuss the benefits and downsides of common adaptation methods (Sec. A.1.3). We then propose an adaptation method that is fast and can be applied to several tasks (Sec. 3.3.1). To adapt, one also needs to be able to compute an adaptation signal, or *proxy*, at the test-time. In the second part of the section, we implement a number of practical adaptation signals for a number of tasks (Sec. 3.3.2).

3.3.1 How to adapt at test-time?

An adaptive system is one that can respond to changes in its environment. More concretely, it is a system that can acquire information to characterize such changes, e.g., via an adaptation signal that provides an error feedback, and make modifications that would result in a reduction of this error (see Fig. 3.1). The methods for performing the adaptation of the system range from gradient-based updates, e.g., just using SGD to fine-tune the parameters [181, 53, 128], – to the more efficient semi-amortized [153, 205] and amortized approaches [206, 151, 152] (see Fig. 6 of [152] for an overview). As amortization methods train a controller network to substitute the explicit optimization process, they only require a forward pass at test-time. Thus, they are computationally efficient. Gradient-based approaches, e.g., TTO, can be powerful adaptation methods when the test-time signal is robust and well-suited for the task (see Fig. 3.4). However, they are inefficient and also have the risk of overfitting and the need for carefully tuned optimization hyperparameters [184]. In this work, we focus on an amortization-based approach, RNA.

Notation. We use $\mathcal{X}$ to denote the input image domain, and $\mathcal{Y}$ to denote the target domain for a given task. We use $f_\theta : \mathcal{X} \to \mathcal{Y}$ to denote the model to be adapted, where θ denotes the model parameters. We denote the model before and after adaptation as f_θ and $f_{\hat{\theta}}$ respectively, where the latter is used to get the final predictions after adaptation. $\mathcal{L}$ and $\mathcal{D}$ are the original training loss and training dataset of f_θ, e.g., for classification, $\mathcal{L}$ will be the cross-entropy loss and $\mathcal{D}$ the ImageNet training data. As shown in Fig. 3.1, h_ϕ is a controller for f_θ. It can be an optimization algorithm, e.g., SGD, or a neural network.

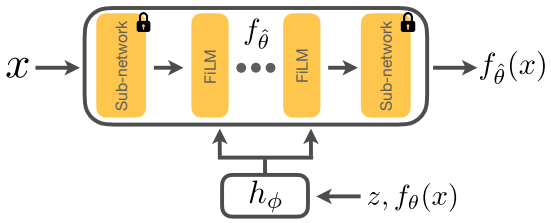

Figure 3.2: **Architecture of RNA.** x is the input image, f_θ is the model to be adapted and $f_\theta(x)$ the corresponding prediction. To perform adaptation, we freeze the parameters of f_θ and insert several FiLM layers into f_θ. We then train h_ϕ to take in z, the adaptation signal, and $f_\theta(x)$ to predict the parameters of these FiLM layers. This results in an adapted model $f_{\hat{\theta}}$ and the improved predictions, $f_{\hat{\theta}}(x)$.

ϕ denotes the optimization hyperparameters or the network's parameters. The former case corresponds to TTO, and the latter is the proposed RNA, which will be explained in the next subsection. Finally, the function $g : \mathcal{X}^M \to \mathcal{Z}$ returns the adaptation signal by mapping a set of images $\mathcal{B} = \{I_1, ..., I_M\} \in \mathcal{X}^M$ to a vector $g(\mathcal{B}) = z \in \mathcal{Z}$. This function g is given, e.g., for depth, g returns the sparse depth measurements computed via SFM.

Rapid Network Adaptation (RNA)

For adaptation, we choose to use a neural network for h_ϕ. The adaptation signal and model predictions are passed as inputs to h_ϕ and it is trained to regress the parameters $\hat{\theta}(\phi) = h_\phi(f_\theta(\mathcal{B}), z)$. This corresponds to an objective-based amortization of the TTO process [166]. Using both the adaptation signal z and model prediction $f_\theta(\mathcal{B})$ informs the controller network about the potential *errors* of the model. The training objective for h_ϕ is $\min_\phi \ \mathbb{E}_\mathcal{D}\left[\mathcal{L}(f_{\hat{\theta}(\phi)}(\mathcal{B}), y)\right]$, where $(\mathcal{B}, y) \sim \mathcal{D}$ is a training batch sampled from $\mathcal{D}$. Note that the original weights of f are frozen and h_ϕ is a small network, having only 5-20% of the number of parameters of f, depending on the task. We call this method as *rapid network adaptation* (RNA) and experiment with different variants of it in Sec. 3.4.

There exist many options for implementing the amortization process, e.g., h_ϕ can be trained to update the input image or the weights of f_θ. We choose to modulate the features of f_θ as it has been shown to work well in different domains [156] and gave the best results. To do this, we insert k Feature-wise Linear Modulation (FiLM) layers [157] into f_θ. Each FiLM layer performs: $\text{FiLM}(\mathbf{x}_i; \gamma_i, \beta_i) = \gamma_i \odot \mathbf{x}_i + \beta_i$, where $\mathbf{x_i}$ is the activation of layer i. h_ϕ is a network that takes as input the adaptation signal z and model predictions and outputs the coefficients $\{\gamma_i, \beta_i\}$ of all k FiLM layers. h_ϕ is trained on the same dataset $\mathcal{D}$ as f_θ, therefore, unlike TTO, it is *never exposed to distribution shifts during training.* Moreover, it is able to generalize to unseen shifts (see Sec. 3.4.3). See the supplementary for the full details as well as the other RNA implementations we investigated.

3.3.2 Which test-time adaptation signals to use?

Independent of the RNA method and while developing adaptation signals is not the main focus of this study, we need to choose some for experimentation. Existing test-time adaptation signals, or proxies, in the literature include prediction entropy [53], spatial autoencoding [128] and self-supervised tasks like rotation prediction [181], contrastive [182] or clustering [184] objectives. The more aligned the adaptation signal is to target task, the better the performance on the target task [181, 182]. More importantly, a poor signal can cause the adaptation to fail silently [184, 128]. Figure 3.3 shows how the original loss on the target task changes as different proxy losses from the literature, i.e. entropy [53], consistency between different middle domains [142, 72] are minimized. In all cases, the proxy loss decreases, however, the improvement in the target loss varies. Thus, successful optimization of existing proxy losses does not necessarily lead to better performance on the target task. In this chapter, we focus on adopting a few practical and real-world signals for our study. Furthermore, RNA turns out to be less susceptible to a poor adaptation signal vs TTO (see sup. mat. Tab. 1). This is because RNA is a neural network *trained* to use these signals to improve the target task, as opposed to being fixed at being SGD as TTO is.

Employed test-time adaptation signals

We develop test-time adaptation signals for several geometric and semantic tasks as shown in Fig. 3.4. Our focus is not on providing an extensive list of adaptation signals, but rather on using practical ones for experimenting with RNA as well as demonstrating the benefits of using signals that are rooted in the known structure of the world and the task in hand. For example, geometric computer vision tasks naturally follow the multi-view geometry constraints, thus making that a proper candidate for approximating the test-time error, and consequently, an informative adaptation signal.

Geometric Tasks. The field of multi-view geometry and its theorems, rooted in the 3D structure of the world, provide a rich source of adaptation signals. We demonstrate our results on the following target tasks: monocular depth estimation, optical-flow estimation, and 3D reconstruction. For all, we first run a standard structure-from-motion (SFM) pipeline [207]. Then, for depth estimation, we employ the z-coordinates of resulting sparse 3D keypoints from each image as the adaptation signal. For optical flow, we perform keypoint matching across images (which returns sparse optical flow). Lastly, for 3D reconstruction, in addition to the previous two signals, we employ consistency between depth and optical flow predictions as another signal.

Semantic Tasks. For semantic segmentation, we first experiment with using a low number of click annotations for each class, similar to the works on active annotation tools [192, 190, 189]. Likewise, for classification, we use the hierarchical structure of semantic classes, and use coarse labels generated from the WordNet tree [208], similar to [209]. Although these signals (click annotations and coarse labels) are significantly

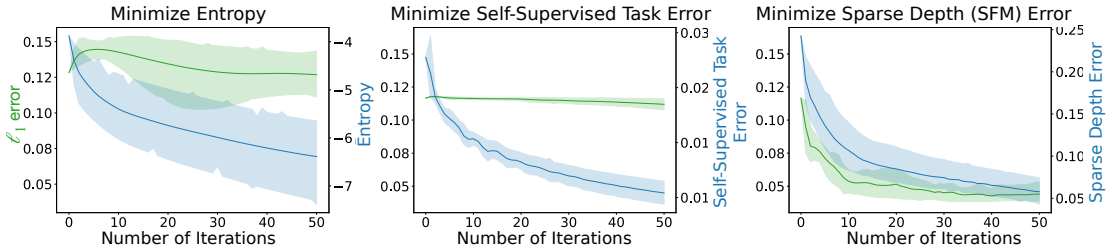

Figure 3.3: **Adaptation using different signals. Not all improvements in proxy loss translates into improving the target task's performance.** We show the results of adapting a pre-trained depth estimation model to a defocus blur corruption by optimizing different adaptation signals: prediction entropy [53], a self-supervised task (sobel edge prediction error [142]), and sparse depth obtained from SFM. The plots show how the ℓ_1 target error with respect to ground-truth depth (green, left axis) changes as the proxy losses (blue, right axis) are optimized (shaded regions represent the 95% confidence intervals across multiple runs of stochastic gradient descent (SGD) with different learning rates). Only adaptation with the sparse depth (SFM) proxy leads to a reduction of the target error. This signifies the importance of employing proper signals in an adaptation framework. Furthermore, we show that RNA is less susceptible to poorer adaptation signal, which results in comparable or improved performance while being significantly faster (see sup. mat. Table 1).

weaker versions of the actual ground truth, thus being cheaper to obtain, it may not be realistic to assume access to them at test-time for certain applications, e.g., real-time ones. Thus, we also show how these can be obtained via k-NN retrieval from the training dataset and patch matching using spatial features obtained from a pre-trained self-supervised vision backbone [195] (see Fig. 3.4).

3.4 Experiments

We demonstrate that our approach consistently outperforms the baselines for adaptation to **different distribution shifts** (2D and 3D Common Corruptions [2, 4], cross-datasets), over **different tasks** (monocular depth, image classification, semantic segmentation, optical flow) and **datasets** (Taskonomy [88], Replica [87], ImageNet [210], COCO [211], ScanNet [212], Hypersim [213]).

3.4.1 Experimental Setup

We describe our experimental setup, i.e. the different adaptation signals, adaptation mechanisms, datasets and baselines, for different tasks. Please see Tab. 3.1 for a summary.

Baselines. We evaluate the following baselines:

Pre-Adaptation Baseline: The network f that maps from RGB to the target task, e.g.,

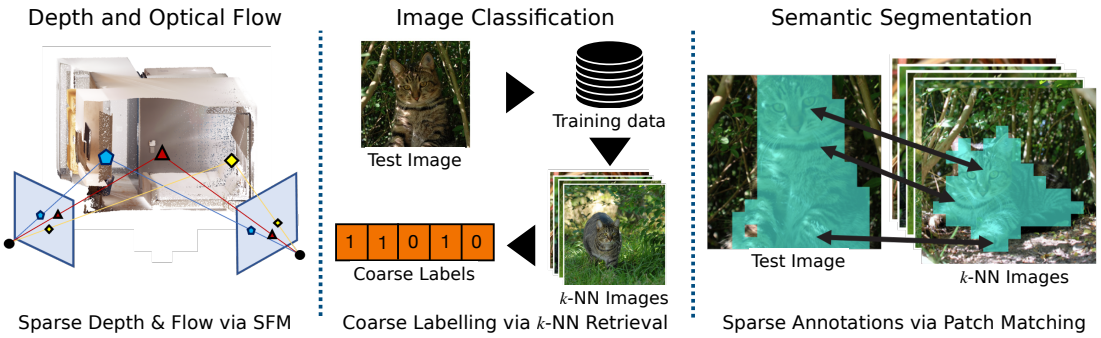

Figure 3.4: **Examples of employed test-time adaptation signals.** We use a range of adaptation signals in our experiments. These are practical to obtain and yield better performance compared to other proxies. In the left plot, for depth and optical flow estimation, we use sparse depth and optical flow via SFM. In the middle, for classification, for each test image, we perform k-NN retrieval to get k training images. Each of these retrieved image has a one hot label associated with it, thus, combining them gives us a coarse label that we use as our adaptation signal. Finally, for semantic segmentation, after performing k-NN as we did for classification, we get a pseudo-labelled segmentation mask for each of these images. The features for each patch in the test image and the retrieved images are matched. The top matches are used as sparse supervision. See Sec. 3.4.1 for more details.

Task	Adaptation signal	Adapted model	Training data	OOD evaluation data	Baselines
Depth	SFM, masked GT	UNet [94], DPT [214]	Taskonomy	*SFM*: Replica, Replica-CC, ScanNet, *masked GT*: Taskonomy-CC,-3DCC, Hypersim	Pre-adaptation, densification, TENT, TTO-edges, TTO
Optical flow	Keypoint matching	RAFT [215]	FlyingChairs, FlyingThings [215]	Replica-CC	Pre-adaptation
3D reconstruction	SFM, keypoint matching,consistency	Depth, optical flow models	Depth, optical flow data	Replica-CC	Pre-adaptation, TTO
Semantic segmentation	Click annotations, patch matching	FCN [216]	COCO (20 classes from Pascal VOC)	*Click annotations*: COCO-CC, *Patch matching*: ImageNet-C	Pre-adaptation, densification, TENT, TTO
Classification	Coarse labels (WordNet, DINO k-NN)	ResNet50 [100], ConvNext [141]	ImageNet	ImageNet-C, ImageNet-3DCC, ImageNet-V2	Pre-adaptation, DINO k-NN, densification, TENT, TTO

Table 3.1: **Overview of the experiments for different target tasks, adaptation methods, and adaptation signals.** For each task, we list the adaptation signal (Sec. 3.3.2) that we use for adaptation. We also list the models that we adapt, and the out-of-distribution (OOD) data used for evaluations and the relevant baselines. When there are different options for adaptation signal, e.g., in the case of depth, the signal is denoted in italics followed by the corresponding OOD dataset. The weights for the semantic segmentation, classification and optical flow models were taken from PyTorch [217].

depth estimation, with no test-time adaptation. We denote this as Baseline for brevity.

Densification: A network that maps from the given adaptation signal for the target task to the target task, e.g., sparse depth from SFM to dense depth. This is a control baseline and shows what can be learned from the test-time supervision alone, without employing input image information or a designed adaptation architecture. See Sec. 3.4.3 for a variant which includes the image.

TTO (episodic): We adapt the Baseline model to each episode by optimizing the proxy loss (see Tab. 3.1 for the adaptation signal used for each task.) at test-time. Its weights

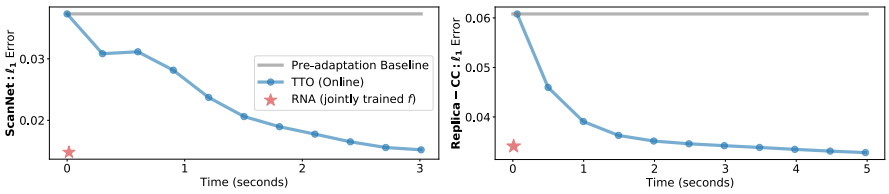

Figure 3.5: **RNA can achieve similar performance as TTO in a much shorter time.** We compare how the ℓ_1 errors of the adaptation mechanisms decrease over wall-clock time (s). The errors are averaged over all episodes (and all corruptions for Replica-CC). RNA only requires a forward pass at test-time, while TTO requires multiple forward and backward passes. On ScanNet and Replica-CC, RNA takes 0.01s, while TTO takes 3s to achieve similar performance. Furthermore, RNA is *not trained with test-time shifts* unlike TTO, thus, it learned to use the additional supervision to adapt to *unseen shifts*.

are reset to the Baseline model's after optimizing each batch, similar to [53, 127].

TTO (online): We continually adapt to a distribution shift defined by a corruption and severity. Test data is assumed to arrive in a stream, and each data point has the same distribution shift, e.g., noise with a fixed standard deviation [53, 181]. The difference with TTO (episodic) is that the model weights are not reset after each iteration. We denote this as TTO for brevity.

TTO with Entropy supervision (TENT [53]): We adapt the Baseline model trained with log-likelihood loss by optimizing the entropy of the predictions. This is to reveal the effectiveness of entropy as a signal as proposed in [53].

TTO with Sobel Edges supervision (TTO-Edges): We adapt the Baseline model trained with an additional decoder that predicts a self-supervised task, similar to [181]. We choose to predict Sobel edges as it has been shown to be robust to certain shifts [142]. We optimize the error of the edges predicted by the model and edges extracted from the RGB image to reveal the value of edge error as a supervision.

RNA configurations. At test-time, we first get the predictions of the Baseline model and compute an adaptation signal. The predictions and adaptation signal are then passed to h_ϕ which adapts f_θ to $f_{\hat{\theta}}$. The test images are then passed to $f_{\hat{\theta}}$ to get the final predictions. We evaluate following variants of RNA.

RNA (frozen f): Baseline model weights, f_θ, are frozen when training h_ϕ. We call this variant *RNA* for brevity.

RNA (jointly trained f): In contrast to *frozen f* variant, here we train RNA jointly with the Baseline network. This variant requires longer training.

Adaptation signal. As described in Sec. 3.3.2, we compute a broad range of test-time signals from the following processes. Each case describes a process applied on query image(s) in order to extract a test-time quantity.

Structure-from-motion (SFM): Given a batch of query images, we use COLMAP [207] to run SFM, which returns sparse depth. The percentage of valid pixels, i.e. depth measurements, is about 0.16% on Replica-CC and 0.18% on Replica. For ScanNet we use the pre-computed sparse depth from [218], which has about 0.04% valid pixels. As running SFM on corrupted images results in noisy sparse depth, we train h_ϕ to be invariant to noise [174, 218].

Masked ground truth (GT): We apply a random mask to the GT depth of the test image. We fixed the valid pixels to 0.05% of all pixels, i.e. similar sparsity as SFM (see the supplementary for other values). This a *control* proxy as it enables a better evaluation of the adaptation methods without conflating with the shortcomings of adaptation signals. It is also a scalable way of simulating sparse depth from real-world sensors, e.g., LiDAR, as also done in [170, 219, 220].

Click annotations: We generate click annotations over random pixels for each class in a given image using GT – simulating an active annotation pipeline. The number of pixels ranges from 3 to 25, i.e. roughly 0.01% of the total pixels, similar to [188, 189, 190, 191, 192].

Patch matching: To not use GT click annotations, for each test image, we first retrieve its k-NN images from the original clean training dataset using DINO features [195]. We then get segmentation masks on these k images. If the training dataset has labels for segmentation we use them directly, otherwise we obtain them from a pretrained network. For each of the k training images and test image, we extract non-overlapping patches. The features for each patch that lie inside the segmentation masks of the k training images are matched to the features of every patch in the test image. These matches are then filtered and used as sparse segmentation annotations. See Fig. 3.4 for illustration.

Coarse labels (WordNet): We generate 45 coarse labels from the 1000-way ImageNet labels, i.e. making the labels 22x coarser, using the WordNet tree [208], similar to [194]. See supplementary for more details on the construction and results for other coarse label sets.

Coarse labels (DINO k-NN): For each test image, we retrieve the k-NN images from the training dataset using DINO features [195]. Each of these k training images is associated with an ImageNet class, thus, combining k one-hot labels gives us a coarse label.

Keypoint matching: We perform keypoint matching across images to get sparse optical flow.

3.4.2 Adaptation with RNA vs TTO

Here we summarize our observations from adapting with RNA vs TTO. As described earlier, TTO represents the approach of closed-loop adaptation using the adaptation signal but without benefiting from any amortization (the adaptation process is fixed to be standard SGD). These observations hold across different tasks (see Sec. 3.4.3 for results).

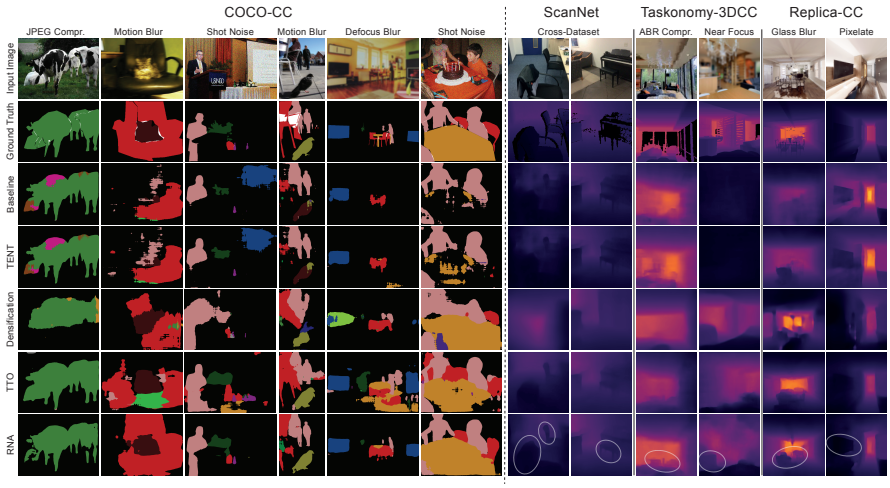

Figure 3.6: **Qualitative results of RNA vs the baselines** for semantic segmentation on random query images on COCO-CC (left) and depth on images from ScanNet, Taskonomy-3DCC and Replica-CC (right). For semantic segmentation, we use 15 pixel annotations per class. For Taskonomy-3DCC, we use sparse depth with 0.05% valid pixels (30 pixels per image). See Fig. 3.7 for results on different adaptation signal levels. For ScanNet and Replica-CC, the adaptation signal is sparse depth measurements from SFM [207] with similar sparsity ratios to Taskonomy-3DCC. The predictions with proposed adaptation signals are shown in the last two rows. They are noticeably more accurate compared to the baselines. Comparing TTO and RNA, RNA's predictions are more accurate for segmentation, and sharper than TTO for depth (see the ellipses) while being significantly faster. See supplementary for more results.

RNA is efficient. Being able to adapt efficiently at test-time is crucial for many real-world problems. As RNA only requires a forward pass at test-time, it is orders of magnitude faster than TTO and is able to attain comparable performance to TTO. In Fig. 3.5, we compare the runtime of adaption with RNA and TTO for depth prediction. On average, for a given episode, RNA obtains similar performance as TTO in 0.01s, compared to TTO's 3-5s. Similarly, for dense 3D reconstruction, RNA is able to adapt in 0.008s compared to TTO's 66s (see Fig. 3.9). This suggests a successful amortization of the adaptation optimization by RNA.

Furthermore, RNA's training is also efficient as it only requires training a small model, i.e. 5-20% of the Baseline model's parameters, depending on the task. Thus, RNA has a fixed overhead, and small added cost at test-time.

RNA's predictions are sharper than TTO for dense prediction tasks. From the last two rows of Fig. 3.6, it can be seen that RNA retains fine-grained details. This is a noteworthy point and can be attributed to the fact that *RNA benefits from a neural network, thus its inductive biases can be beneficial (and further engineered) for such advantages*. This is a general feature that RNA, and more broadly using a learning-based function to amortize adaptation optimization, brings – in contrast to limiting the

Adaptation Signal	SFM			Sparse GT				Relative Runtime
Dataset	Replica		ScanNet	Taskonomy			Hypersim	
Shift	CDS	CC	CDS	None	CC	3DCC	CDS	
Pre-adaptation Baseline	1.75	6.08	3.30	2.68	5.74	4.75	33.64	1.00
Densification	2.50	4.19	2.35	1.72	1.72	1.72	17.25	1.00
TENT [53]	2.03	6.09	4.03	5.51	5.51	4.48	35.45	15.85
TTO-Edges [181]	1.73	6.14	3.28	2.70	5.69	4.74	33.69	20.98
RNA (frozen f)	1.72	4.26	**1.77**	**1.12**	1.68	1.49	16.17	**1.56**
RNA (jointly trained f)	**1.66**	3.41	**1.74**	**1.11**	**1.50**	**1.37**	17.13	**1.56**
TTO (Episodic)	1.72	3.31	1.85	1.62	2.99	2.31	17.77	14.85
TTO (Online)	1.82	**3.16**	**1.76**	**1.13**	**1.48**	**1.34**	**14.17**	14.85

Table 3.2: **Quantitative adaptation results on depth estimation.** ℓ_1 errors on the depth prediction task. (Lower is better. Multiplied by 100 for readability. The best models within 0.0003 error are shown in bold.) We generate distribution shifts by applying Common Corruptions (CC), 3D Common Corruptions (3DCC) and from performing cross-dataset evaluations (CDS). The results from CC and 3DCC are averaged over all distortions and severity levels on Taskonomy and 3 severity levels on Replica data. The adaptation signal from Taskonomy is masked GT (fixed at 0.05% valid pixels) while that from Replica and ScanNet is sparse depth from SFM. RNA and TTO notably outperform the baselines. **RNA successfully matches the performance of TTO while being around 10 times faster**. See supplementary for the losses for different corruption types, sparsity levels, and the results of applying RNA to other adaptation signals.

adaptation process to be SGD, as represented by TTO.

RNA generalizes to unseen shifts. RNA performs better than TTO for low severities (see supplementary for more details). However, as it was *not exposed to any corruptions*, the performance gap against TTO narrows at high severities as expected, which is *exposed to corruptions* at test-time.

We hypothesize that the generalization property of RNA is due to the following reasons. **1.** Even though f_θ was trained to convergence, it does not achieve exactly 0 error. Thus, when h_ϕ is trained with a frozen f_θ with the training data, it can still learn to correct the errors of f_θ, thus, adapting f_θ. **2.** Adaptation signals, by definition, are expected to be relatively robust to distribution shifts. Even though the RGB image has been corrupted or from a new domain, the adaptation signal is more aligned with the target task. Thus, the input to h_ϕ does not undergo a significant shift and is able to adapt f_θ.

3.4.3 Experiments using Various Target Tasks

In this section, we provide a more comprehensive set of evaluations covering various target tasks and adaptation signals. In all cases, RNA is a fixed general framework without being engineered for each task and shows supportive results.

Depth. We demonstrate the results quantitatively in Tab. 3.2 and Fig. 3.5 and qualita-

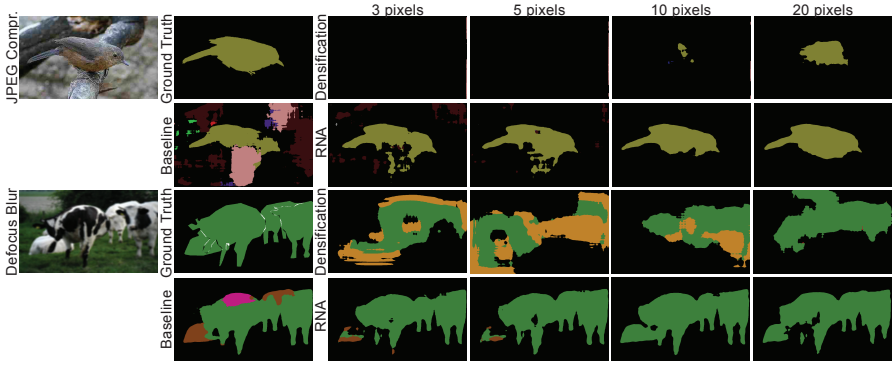

Figure 3.7: **Qualitative adaptation results on semantic segmentation** on random query images on COCO-CC. RNA notably improves the prediction quality using error feedback from as few as 3 random pixels.

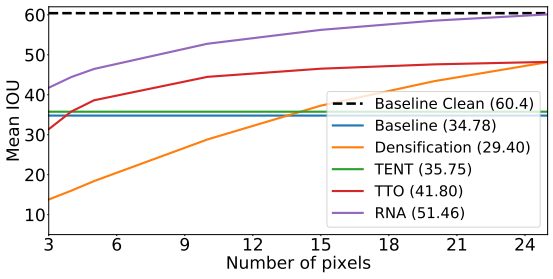

Figure 3.8: **Quantitative adaptation results on semantic segmentation**. Each point shows the mean IOU over 15 corruptions and 5 severities. RNA significantly improves over baselines. Black dashed line shows the mean IOU of baseline model for *clean* validation images, and is provided as a reference. Numbers in the legend denote averages over all supervision pixel counts. See supplementary for a breakdown.

tively in Fig. 3.6. In Tab. 3.2, we compare RNA against all baselines, and over several distribution shifts and different adaptation signals. Our RNA variants outperform the baselines overall. TTO (online) has a better performance than TTO (episodic) as it assumes a smoothly changing distribution shift, and it continuously updates the model weights. RNA (jointly trained f) has a better performance among RNA variants. This is reasonable as the target model is not frozen, thus, is less restrictive.

As another baseline, we trained a single model that takes as input *a concatenation of the RGB image and sparse supervision*, i.e. multi-modal input. However, its average performance on Taskonomy-CC was 42.5% worse than RNA's (see sup. mat. Sec. 3.2). Among the baselines that do not adapt, densification is the strongest under distribution shift due to corruptions. This is expected as it does not take the RGB image as input, thus, it is not affected by the underlying distribution shift. However, as seen from the qualitative results in Figs. 3.6, 3.7, it is unable to predict fine-grained details, unlike RNA. We also show that the gap between RNA and densification widens with sparser supervision (see sup. mat. Fig. 1), which confirms that RNA is making use of the error feedback signal, to adapt f.

Dense 3D Reconstruction. Here, we combine multiple adaptation signals from

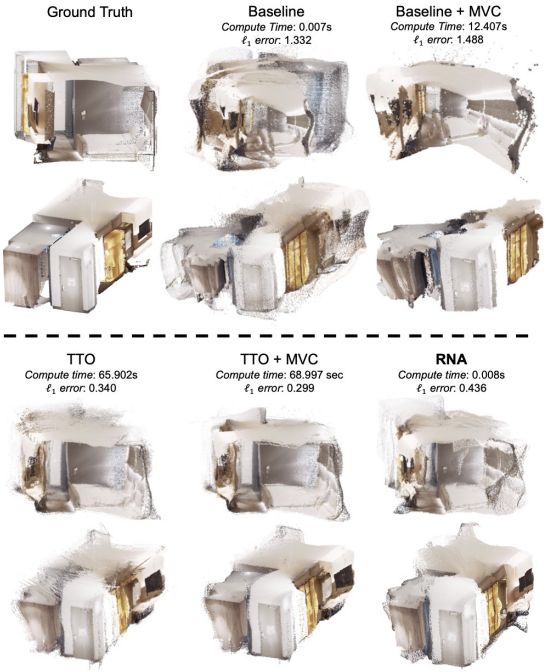

Figure 3.9: **Adaptation results for 3D reconstruction.** Using appropriate adaptation signals from multi-view geometry can recover accurate 3D reconstructions. We report the average ℓ_1 error between ground truth 3D coordinates and the estimated ones. The titles above each column refers to the depth model used to get the reconstruction. TTO+MVC corresponds to the predictions after multi-view consistency optimization. It can be seen that RNA and TTO improve the reconstructions over the baselines with RNA being significantly faster. See supplementary for more results.

multi-view geometry. First, we adapt the weights of the depth and optical flow models independently. The results from this adaptation can be found in the previous paragraph (for depth) and supplementary (for optical flow). Next, the two models are adapted to make their predictions consistent with each other. This is achieved using the same process as [179], i.e. multi-view-consistency (MVC). See supplementary for details.

Figure 3.9 shows the point cloud visualizations on Replica with Gaussian Noise corruption. This results in collapsed depth predictions, thus, reconstructions are unusable (Baseline column) and performing MVC is not helpful (Baseline+MVC). Adapting the depth predictions using TTO and MVC improves the reconstruction notably while RNA achieves a similar performance significantly faster.

Semantic Segmentation. We experiment with click annotations and DINO patch matching as adaptation signals.

Click annotations: In Fig. 3.8, we show how the IoU changes with the adaptation signal level on COCO-CC. As the Baseline and TENT do not make use of this signal, their IoU is a straight line. RNA clearly outperforms the baselines for all levels of adaptation signal. Figure 3.7 shows the qualitative results with increasing supervision, and Fig. 3.6 (left) a comparison against all baselines, demonstrating higher quality predictions with RNA.

DINO patch matching: We perform patch matching on DINO features (described in Sec. 3.4.1) to get the adaptation signal. As the patch matching process can be computationally expensive, we demonstrate our results on all cat classes in ImageNet and over

Adaptation Signal	Dataset	Clean	IN-C	IN-3DCC	IN-V2	Rel. Runtime
-	Pre-adaptation Baseline	23.85	61.66	54.97	37.15	1.00
Entropy	TENT	24.67	46.19	47.13	37.07	5.51
Coarse labels (wordnet)	Densification	95.50	95.50	95.50	95.50	-
	TTO (Online)	24.72	**40.62**	42.90	36.77	5.72
	RNA (frozen f)	**16.72**	41.21	**40.37**	**25.53**	**1.39**
Coarse labels (DINO)	DINO (k-NN)	25.56	52.64	48.24	37.39	-
	TTO (Online)	24.59	51.59	49.18	36.96	5.72
	RNA (frozen f)	24.36	54.86	52.29	36.88	**1.39**

Table 3.3: **Quantitative adaptation results on on ImageNet (IN) classification task.** We evaluate on the clean validation set, ImageNet-{C,3DCC,V2}. We report average error (%) for 1000-way classification task over all corruptions and severities. For the coarse labels with WordNet supervision, we use 45-coarse labels. For DINO k-NN, we set $k = 20$.

one noise, blur and digital corruption for 3 levels of severity. We used the predictions of a pre-trained FCN on the clean images as pseudolabels to compute IoU. The mean IoU averaged over these corruptions and severities is 48.98 for the baseline model, 53.45 for TTO. RNA obtains a better IOU of 58.04, thus it can make use of the sparse annotations from DINO patch matching.

Image Classification. We experiment with coarse labels from WordNet and DINO k-NN as adaptation signals.

Coarse labels (WordNet): Table 3.3 shows the results from using 45-coarse labels on ImageNet-{C,3DCC,V2}. This corresponds to 22x coarser supervision compared to the 1000 classes that we are evaluating on. TENT seems to have notable improvements in performance under corruptions for classification, unlike for semantic segmentation and depth. We show that using coarse supervision results in even better performance, about a further 5 pp reduction in error. Furthermore, on uncorrupted data, i.e. clean, and ImageNet-V2 [221], RNA gives roughly 10 pp improvement in performance compared to TTO. Thus, coarse supervision provides a useful signal for adaptation while requiring much less effort than full annotation [193]. See supplementary for results on other coarse sets.

Coarse labels (DINO k-NN): We also show results from using coarse sets generated from DINO k-NN retrieval. This is shown in the last 3 rows of Tab. 3.3. Both RNA and TTO use this coarse information to outperform the non-adaptive baselines. However, they do not always outperform TENT, which could be due to the noise in retrieval.

3.4.4 Ablations and additional results

Adaptation on other architectures. Table 3.4 shows the results of incorporating RNA to different architectures, namely the dense prediction transformer (DPT) [214] for depth and ConvNext [141] for image classification. In both cases, RNA is able to improve on the error and runtime of TTO. Thus, RNA can be applied to a range of

Task (Arch.)	Depth (DPT [214])			Classification (ConvNext [141])		
Shift	**Clean**	**CC**	**Rel. Runtime**	**Clean**	**IN-C**	**Rel. Runtime**
Pre-adaptation Baseline	2.23	3.76	1.00	18.13	42.95	1.00
TTO (Online)	1.82	2.61	13.85	17.83	41.44	11.04
RNA (frozen f)	**1.13**	**1.56**	**1.01**	**14.32**	**38.04**	**1.07**

Table 3.4: **RNA works across different architectures.** Quantitative adaptation results on depth estimation and image classification on Taskonomy and ImageNet datasets, respectively. (Lower is better. ℓ_1 errors for depth estimation are multiplied by 100 for readability.)

architectures.

Controlling for number of parameters. We ran a control experiment where all methods have the same architecture, thus, same number of parameters. The results are in supplementary Table 2. RNA still returns the best performance. Thus, its improvement over the baselines is not due to a different architecture or number of parameters but due to its test-time adaptation mechanism.

Different implementations of RNA. We experiment with different controller architectures e.g., HyperNetworks [145], other FiLM variants, or adapting the input instead of the model parameters. See supplementary Sec. 2.2 for the details and a conceptual discussion on the trade-offs of the choices of implementing this closed-loop "control" system, namely those that make stronger model-based assumptions.

3.5 Discussions of RNA compared to other approaches

There are many methods that aim to handle distribution shifts. Figure 3.10 gives an overview of how these methods can be characterized. Open-loop systems predict y by only using their inputs *without receiving feedback*. Training-time robustness methods, image modifications, and multi-modal methods fall into this category. These methods assume the learned model is frozen at test-time. Thus, they aim to incorporate inductive biases at training time that could be useful against distribution shifts at test-time. The closed-loop systems, on the other hand, are *adaptive* as they make use of an error feedback signal that can be computed at test-time from the model predictions and an adaptation signal.

The closed-loop systems can be instantiated as *model-based* or *model-free* adaptation methods. The former performs adaptation by estimating the parameters of the distribution shift using the feedback signal. For this purpose, different forms of feedback can be useful, e.g. an error feedback and the feedback from the input image itself can lead to successful estimation of the shift parameters. This is a form of inductive bias that can help the method generalize for similar shifts. Furthermore, explicitly modelling the distribution shift parameters results in an interpretable system that will not fail

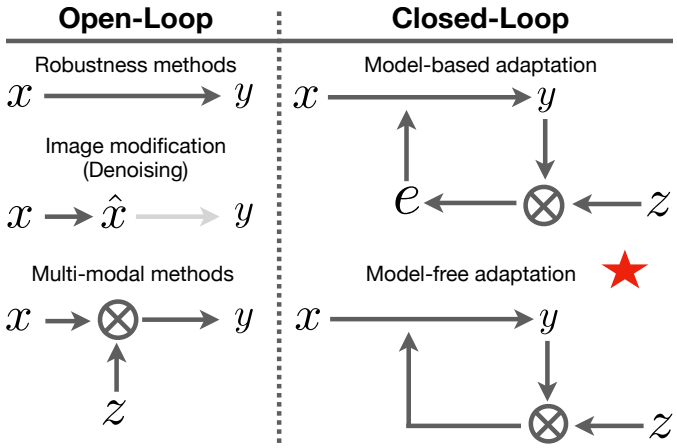

Figure 3.10: **An overview of methods that aim to handle distribution shifts.** On the left we have open-loop systems. They predict y by only using their inputs *without receiving feedback*. The first and popular example of open-loop systems is training-time robustness methods (data augmentation, architectural changes, etc.). The next example is the methods that modify the input x, e.g. denoising or style changes, independent of y. Furthermore, there are multi-modal methods that use additional input z. As the learned model is frozen at test-time, these methods need to *anticipate* the distribution shift by incorporating inductive biases at training time (See also Fig. 3.1). In contrast, closed-loop systems on the right make use of its current output, y, and an adaptation signal, z, to form an *error feedback signal* that can be used to update its predictions. Thus, they *adapt* to the shifts as they occur. We can then group closed-loop systems into model-based and model-free methods. The former performs adaptation by estimating the parameters of the distribution shift, e, while the latter performs adaptation without explicitly predicting e. Furthermore, this adaptation can be performed via running an optimization, i.e. test-time optimization (TTO) via SGD, or via amortization, i.e., training a side-network to predict TTO updates that minimizes the error feedback. Our proposed method, RNA, belongs to the model-free adaptation approaches that makes use of amortization for efficiency.

silently. In [222], to learn a policy for legged robots, they defined defined a model of the environment e.g., ground friction, center of mass, etc., and showed that it generalized well. However, our experiments with a model-based approach, where we defined the parameters of the environment as the possible distribution shifts can occur e.g., intensity of a noise or blur corruption, did not show promising results. It can because our model is not an accurate reflection of the real world, and/or that such a model is not needed for static vision tasks.

In contrast, model-free methods do not estimate these parameters and learn to adapt based only on the error feedback signal. Our proposed method RNA belongs to model-free approaches, and as we have shown in this chapter, it generalizes well to a diverse set of unseen distribution shifts. Note that model-free adaptation methods, including RNA, has schematic similarities to multi-modal learning approaches as they simultaneously use

an RGB input image and an adaptation signal. The main distinction is that our method implements a particular process toward adapting a network to a shift using an adaptation signal from the environment – as opposed to a generic multi-modal learning.

Closed-loop systems aim to minimize an error feedback signal. There are different ways to implement this. Running an optimization, e.g. SGD, at test-time is a popular choice, as done in several other works [181, 53, 128, 54]. Since this may be unnecessarily expensive, another way is to *amortize* this optimization process [166]. This results in our proposed method, RNA.

RNA vs Training-time Robustness Methods. As discussed before, training-time robustness methods, e.g. data augmentation, aim to anticipate distribution shifts by building invariances at training-time. On the other hand, RNA performs adaptation at test-time using error feedback. Furthermore, as RNA makes use of an error signal, it is able to handle cases where there are multiple possible predictions for a given input, e.g., scale ambiguity for monocular depth estimation, while the robustness methods may not.

RNA vs Denoising. The denoising methods, and in general the methods performing modification in the input image, e.g., domain adaptation methods that aims to map an image in the target domain to the style of the source domain [80], are concerned with reconstructing plausible images *without* taking the downstream prediction $x \rightarrow y$ into account (shown as gray in Fig. 3.10). Moreover, it has been shown that imperceptible artifacts in the denoised & modified image could result in degraded predictions [2, 142, 183]. In contrast, RNA performs updates with the goal of reducing the error of the target task.

RNA vs Model-based Adaptation. As explained before, model-based approaches require building a well-defined model of the distribution shifts that can be faced at test-time. While this can effectively work for the modeled shifts, the performance can quickly deteriorate for the ones that are outside the scope of the model. Thus, the effectiveness for a more general and practical adaptation setting is limited. In contrast, RNA adopts a model-free adaptation approach that does not require a tedious modelling of distribution shifts at the expense of lack of interpretability. As our experiments on a diverse set of tasks and distribution shifts show, RNA is able to generalize and outperforms the baselines.

RNA vs TTO. RNA aims to amortize the optimization process of TTO. Thus, it aims to simulate TTO by learning to reduce errors in the predictions using the error feedback signal. In Sec. 3.4, we show that RNA is able to adapt orders of magnitude faster than running test-time optimization (TTO). See Sec. 3.4.2 for detailed discussions and results.

Using different forms of feedback signal as input to RNA. In the case where

only the adaptation signal, z, is passed as input, it is possible that the side-network is implicitly modelling an error feedback signal. This is because it is trained alongside the main model ($x \rightarrow y$), thus, it sees and learns to correct the main model's errors during training. We found that having an error feedback signal as input results in better performance on average, thus, we adopted this as our main method.

3.6 Conclusion and Limitations

We presented RNA, a method for efficient adaptation of neural networks at test-time using a closed-loop formulation. It involves training a side network to use a test-time adaptation signal to adapt a main network. This network acts akin to a "controller" and adapts the main network based on the adaptation signal. We showed that this general and flexible framework can generalize to unseen shifts, and as it only requires a forward pass at test-time, it is orders of magnitude faster than TTO. We evaluated this approach using a diverse set of adaptation signals and target tasks. We briefly discuss the limitations and potential future works:

Different implementations of RNA and amortization methods. While we experimented with several RNA variants (see supplementary for details), further investigations using other techniques, e.g. cross-attention [147] or building in a more explicit "model" of the shifts and environment, could be worthwhile. In general, as the role of the controller network is to amortize the training optimization of the main network, the amortized optimization literature [166] is an apt resource to consult for this purpose.

Hybrid mechanism for activating TTO in RNA. TTO constantly adapts a model to a distribution shift, hence, in theory, it can adapt to any shift despite being comparatively inefficient. To have the best of both worlds, investigating mechanisms for selectively activating TTO within RNA when needed can be useful.

Finding adaptation signals for a given task. While the focus of this study was not on developing new adaption signals, we demonstrated useful ones for several core vision tasks, but there are many more. Finding these signals requires either knowledge of the target task so a meaningful signal can be accordingly engineered or core theoretical works on understanding how a proxy and target objectives can be "aligned" for training.

This chapter is based on the paper: T. Yeo, O. F. Kar, Z. Sodagar, A. Zamir, *Fast Adaptation of Neural Networks using Test-Time Feedback,* ICCV 2023. I was the main contributor of this work.

Benchmarks Part III

4 3D Common Corruptions

4.1 Introduction

This chapter presents a set of distribution shifts to test models' robustness before deploying these models in the real world. In contrast to previously proposed shifts which perform uniform 2D modifications over the image, such as Common Corruptions (2DCC) [2], our shifts incorporate 3D information to generate corruptions that are consistent with the scene geometry. This leads to shifts that are more likely to occur in the real world (See Fig. 4.1). The resulting set includes 20 corruptions, each representing a distribution shift from training data, which we denote as *3D Common Corruptions* (3DCC). 3DCC addresses several aspects of the real world, such as camera motion, weather, occlusions, depth of field, and lighting. Figure 4.2 provides an overview of all corruptions. As shown in Fig. 4.1, the corruptions in 3DCC are more diverse and realistic compared to 2D-only approaches.

We show in Sec. 4.5 that the performance of the methods aiming to improve robustness, including those with diverse data augmentation, reduce drastically under 3DCC. Furthermore, we observe that the robustness issues exposed by 3DCC well correlate with corruptions generated via photorealistic synthesis. Thus, 3DCC can serve as a challenging testbed for real-world corruptions, especially those that depend on scene geometry.

Motivated by this, our framework also introduces new *3D data augmentations*. They take the scene geometry into account, as opposed to 2D augmentations, thus enabling models to build invariances against more realistic corruptions. We show in Sec. 4.5.3 that they significantly boost model robustness against such corruptions, including the ones that cannot be addressed by the 2D augmentations.

The proposed corruptions are *generated programmatically* with *exposed parameters*, enabling fine-grained analysis of robustness, e.g. by continuously increasing the 3D

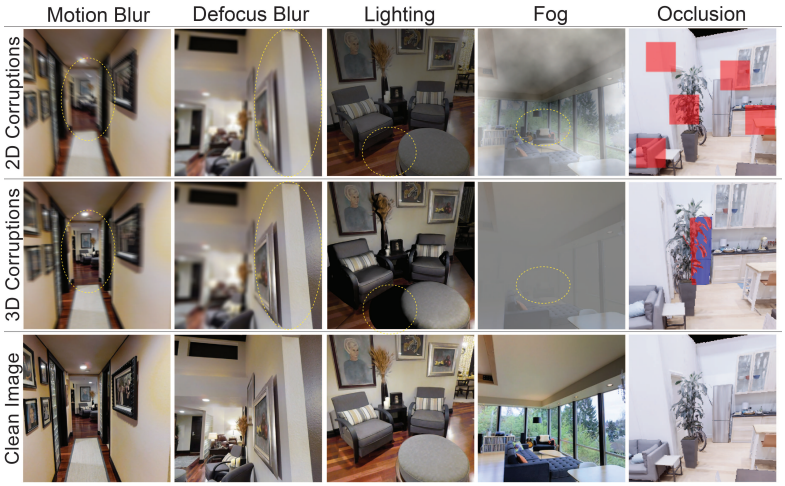

Figure 4.1: **Using 3D information to generate real-world corruptions.** The top row shows sample 2D corruptions applied uniformly over the image, e.g. as in Common Corruptions [2], disregarding 3D information. This leads to corruptions that are unlikely to happen in the real world, e.g. having the same motion blur over the entire image irrespective of the distance to camera (top left). Middle row shows their 3D counterparts from 3D Common Corruptions (3DCC). The circled regions highlight the effect of incorporating 3D information. More specifically, in 3DCC, **1. motion blur** has a *motion parallax* effect where objects further away from the camera seem to move less, **2. defocus blur** has a *depth of field* effect, akin to a large aperture effect in real cameras, where certain regions of the image can be selected to be in focus, **3. lighting** takes the scene geometry into account when illuminating the scene and casts shadows on objects, **4. fog** gets denser further away from the camera, **5. occlusions** of a target object, e.g. fridge (blue mask), are created by changing the camera's viewpoint and having its view *naturally obscured by another object*, e.g. the plant (red mask). This is in contrast to its 2D counterpart that randomly discards patches [223].

motion blur. They are *efficient* to compute and can be computed on-the-fly during training as data augmentation with a small increase in computational cost. They are also *extendable*, i.e. they can be applied to standard vision datasets, e.g. ImageNet [210], that do not come with 3D labels.

4.2 Related Work

This chapter presents a data-focused approach [224, 225] to robustness. We give an overview of some of the related topics within the constraints of space.

Robustness benchmarks based on corruptions: Several studies have proposed robustness benchmarks to understand the vulnerability of models to corruptions. A popular benchmark, Common Corruptions (2DCC) [2], generates synthetic corruptions on real images that expose sensitivities of image recognition models. It led to a series of

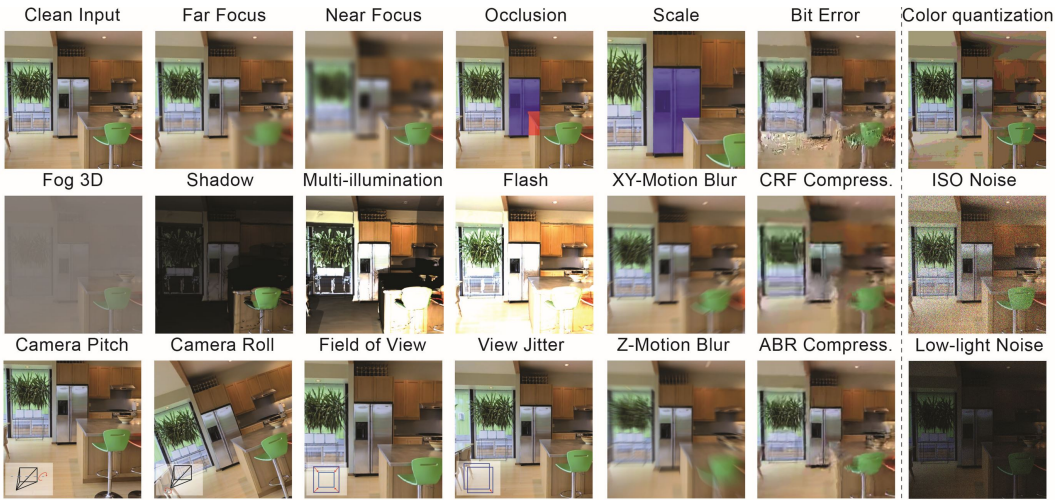

Figure 4.2: **The new corruptions.** We propose a *diverse* set of new corruption operations ranging from defocusing (near/far focus) to lighting changes and 3D-semantic ones, e.g. object occlusion. These corruptions are all *automatically generated*, *efficient* to compute, and can be applied to *most datasets* (Sec. 4.3.3). We show that they expose vulnerabilities in models (Sec. 4.5.2) and are a good approximation of realistic corruptions (Sec. 4.5.2). A subset of the corruptions marked in the last column are novel and commonly faced in the real world, but are not 3D based. We include them in our benchmark. For occlusion and scale corruptions, the blue and red masks denote the amodal visible and occluded parts of an object, e.g. the fridge.

works either creating new corruptions or applying similar corruptions on other datasets for different tasks [226, 227, 228, 229, 230, 231]. In contrast to these works, 3DCC modifies real images *using 3D information* to generate realistic corruptions. The resulting images are both perceptually different and expose different failure modes in model predictions compared to their 2D counterparts (See Fig. 4.1 and 4.8). Other works create and capture the corruptions in the real world, e.g. ObjectNet [232]. Although realistic, it requires significant manual effort and is not extendable. A more scalable approach is to use computer graphics based 3D simulators to generate corrupted data [233] which can lead to generalization concerns. 3DCC aims to generate corruptions *as close to the real world* as possible while staying *scalable*.

Robustness analysis works use *existing* benchmarks to probe the robustness of different methods, e.g. data augmentation or self-supervised training, under several distribution shifts. Recent works investigated the relation between synthetic and natural distribution shifts [102, 7, 103, 234] and effectiveness of architectural advancements [138, 139, 235]. We select several popular methods to show that 3DCC can serve as a challenging benchmark (Fig. 4.6 and 4.7).

Improving robustness: Numerous methods have been proposed to improve model robustness such as data augmentation with corrupted data [84, 11, 129, 236], texture

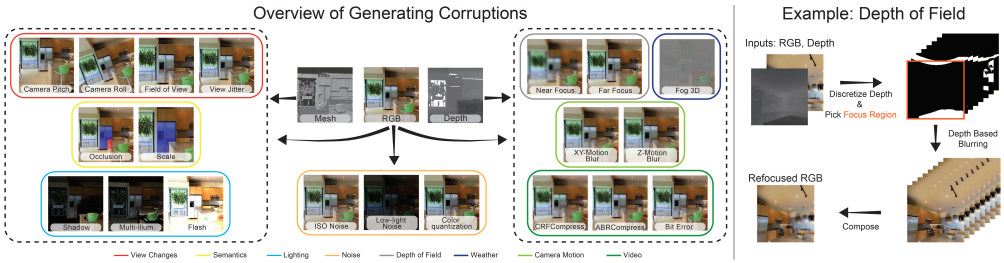

Figure 4.3: **Left:** We show the *inputs* needed to create each of our corruptions, e.g. the 3D information such as depth, and RGB image. These corruptions have also been grouped (in solid colored lines) according to their *corruption types*. For example, to create the distortions in the dashed box in the right, one only needs the RGB image and its corresponding depth. For the ones in the left dashed box, 3D mesh is required. Note that one can create *view changes* corruptions also from panoramic images if available, without a mesh. **Right:** As an example, we show an overview of generating depth of field effect *efficiently*. The scene is first split into multiple layers by discretizing scene depth. Next, a region is chosen to be kept in focus (here it is the region closest to the camera). We then compute the corresponding blur levels for each layer according to their distance from the focus region, using a pinhole camera model. The final refocused image is obtained by compositing blurred image layers.

changes [96, 7], image compositions [40, 39] and transformations [36, 42]. While these methods can generalize to some unseen examples, performance gains are non-uniform [41, 84]. Other methods include self-training [131], pre-training [130, 237], architectural changes [138, 139], and diverse ensembling [23, 24, 25, 142]. Here we instead adopt a data-focused approach to robustness by **i.** providing a large set of realistic distribution shifts and **ii.** introducing new 3D data augmentation that improves robustness against real-world corruptions (Sec. 4.5.3).

Photorealistic image synthesis involves techniques to generate realistic images. Some of these techniques have been recently used to create corruption data. These techniques are generally specific to a single real-world corruption. Examples include adverse weather conditions [238, 239, 240, 241, 242], motion blur [243, 244], depth of field [245, 246, 247, 132, 248], lighting [249, 250], and noise [251, 252]. They may be used for purely artistic purposes or to create training data. Some of our 3D transformations are instantiations of these methods, with the downstream goal of testing and improving model robustness in a unified framework with a wide set of corruptions.

Image restoration aims to undo the corruption in the image using classical signal processing techniques [253, 254, 255, 256] or learning-based approaches [257, 258, 259, 260, 261, 262, 263]. We differ from these works by generating corrupted data, rather than removing it, to use them for benchmarking or data augmentation. Thus, in the latter, we train with these corrupted data to encourage the model to be invariant to corruptions, as opposed to training the model to remove the corruptions as a pre-processing step.

Adversarial corruptions add imperceptible *worst-case* shifts to the input to fool a model [9, 10, 11, 264]. Most of the failure cases of models in the real world are not the result of adversarial corruptions but rather *naturally occurring distribution shifts*. Thus, our focus in this paper is to generate corruptions that are likely to occur in the real world.

4.3 Generating 3D Common Corruptions

4.3.1 Corruption Types

We define different corruption types, namely *depth of field*, *camera motion*, *lighting*, *video*, *weather*, *view changes*, *semantics*, and *noise*, resulting in 20 corruptions in 3DCC. Most of the corruptions require an RGB image and scene depth, while some needs 3D mesh (See Fig. 4.3). We use a set of methods leveraging 3D synthesis techniques or image formation models to generate different corruption types, as explained in more detail below. Further details are provided in the appendix.

Depth of field corruptions create refocused images. They keep a part of the image in focus while blurring the rest. We consider a layered approach [132, 248] that splits the scene into multiple layers. For each layer, the corresponding blur level is computed using the pinhole camera model. The blurred layers are then composited with alpha blending. Figure 4.3 (right) shows an overview of the process. We generate *near focus* and *far focus* corruptions by randomly changing the focus region to the near or far part of the scene.

Camera motion creates blurry images due to camera movement during exposure. To generate this effect, we first transform the input image into a point cloud using the depth information. Then, we define a trajectory (camera motion) and render novel views along this trajectory. As the point cloud was generated from a single RGB image, it has incomplete information about the scene when the camera moves. Thus, the rendered views will have disocclusion artifacts. To alleviate this, we apply an inpainting method from [244]. The generated views are then combined to obtain parallax-consistent motion blur. We define *XY-motion blur* and *Z-motion blur* when the main camera motion is along the image XY-plane or Z-axis, respectively.

Lighting corruptions change scene illumination by adding new light sources and modifying the original illumination. We use Blender [265] to place these new light sources and compute the corresponding illumination for a given viewpoint in the 3D mesh. For the *flash* corruption, a light source is placed at the camera's location, while for *shadow* corruption, it is placed at random diverse locations outside the camera frustum. Likewise, for *multi-illumination* corruption, we compute the illumination from a set of random light sources with different locations and luminosities.

Video corruptions arise during the processing and streaming of videos. Using the scene 3D, we create a video using multiple frames *from a single image* by defining a trajectory, similar to motion blur. Inspired by [229], we generate *average bit rate (ABR)* and *constant rate factor (CRF)* as H.265 codec compression artifacts, and *bit error* to capture corruptions induced by imperfect video transmission channel. After applying the corruptions over the video, we pick a single frame as the final corrupted image.

Weather corruptions degrade visibility by obscuring parts of the scene due to disturbances in the medium. We define a single corruption and denote it as *fog 3D* to differentiate it from the fog corruption in 2DCC. We use the standard optical model for fog [238, 239, 240]:

$$\mathbf{I}(\mathbf{x}) \;=\; \mathbf{R}(\mathbf{x})\mathbf{t}(\mathbf{x}) + \mathbf{A}(1 - \mathbf{t}(\mathbf{x})), \tag{4.1}$$

where $\mathbf{I}(\mathbf{x})$ is the resulting foggy image at pixel x, $\mathbf{R}(\mathbf{x})$ is the clean image, $\mathbf{A}$ is atmospheric light, and $\mathbf{t}(\mathbf{x})$ is the transmission function describing the amount of light that reaches the camera. When the medium is homogeneous, the transmission depends on the distance from the camera, $\mathbf{t}(\mathbf{x}) = \exp(-\beta\mathbf{d}(\mathbf{x}))$ where $\mathbf{d}(\mathbf{x})$ is the scene depth and β is the attenuation coefficient controlling the fog thickness.

View changes are due to variations in the camera extrinsics and focal length. Our framework enables rendering RGB images conditioned on several changes, such as *field of view*, *camera roll* and *camera pitch*, using Blender. This enables us to analyze the sensitivity of models to various view changes in a controlled manner. We also generate images with *view jitter* that can be used to analyze if models predictions flicker with slight changes in viewpoint.

Semantics: In addition to view changes, we also render images by selecting an object in the scene and changing its occlusion level and scale. In *occlusion* corruption, we generate views of an object occluded by other objects. This is in contrast to random 2D masking of pixels to create an unnatural occlusion effect that is irrespective of image content, e.g. as in [223, 235] (See Fig. 4.1). Occlusion rate can be controlled to probe model robustness against occlusion changes. Similarly, in *scale* corruption, we render views of an object with varying distances from the camera location. Note that the corruptions require a mesh with semantic annotations, and are generated automatically, similar to [266]. This is in contrast to [232] which requires tedious manual effort. The objects can be selected by randomly picking a point in the scene or using the semantic annotations.

Noise corruptions arise from imperfect camera sensors. We introduce new noise corruptions that do not exist in the previous 2DCC benchmark. For *low-light noise*, we decreased the pixel intensities and added Poisson-Gaussian distributed noise to reflect the low-light imaging setting [251]. *ISO noise* also follows a Poisson-Gaussian distribution, with a fixed photon noise (modeled by a Poisson), and varying electronic noise (modeled by a Gaussian). We also included *color quantization* as another corruption that reduces the bit depth of the RGB image. Only this subset of our corruptions is not based on 3D

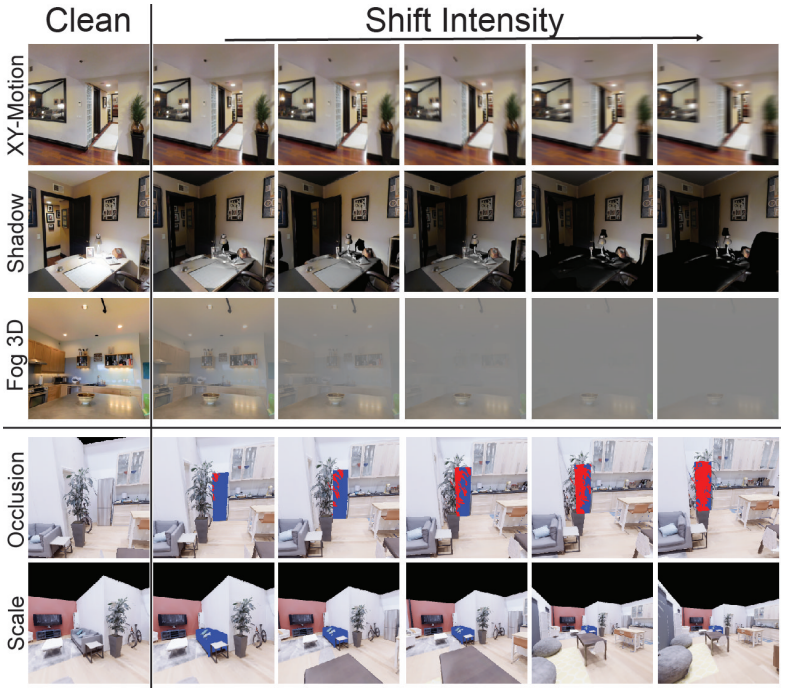

Figure 4.4: **Visualizations of 3DCC with increasing shift intensities**. **Top:** Increasing the shift intensity results in larger blur, less illumination, and denser fog. **Bottom:** The object becomes more occluded or shrinks in size using *calculated viewpoint changes*. The blue mask denotes the amodal visible parts of the fridge/couch, and the red mask is the occluded part. The leftmost column shows the clean images. Visuals for all corruptions for all shift intensities are shown in the appendix.

information.

4.3.2 Starter 3D Common Corruptions Dataset

We release the full open source code of our pipeline, which enables using the implemented corruptions on any dataset. As a starter dataset, we applied the corruptions on 16k Taskonomy [88] test images. For all the corruptions except the ones in *view changes* and *semantics* which change the scene, we follow the protocol in 2DCC and define 5 shift intensities, resulting in approximately 1 million corrupted images (16k×14 × 5). Directly applying the methods to generate corruptions results in uncalibrated shift intensities with respect to 2DCC. Thus, to enable aligned comparison with 2DCC on a more uniform intensity change, we perform a calibration step. For the corruptions with a direct counterpart in 2DCC, e.g. motion blur, we set the corruption level in 3DCC such that for each shift intensity in 2DCC, the average SSIM [267] values over all images is the same in both benchmarks. For the corruptions that do not have a counterpart in 2DCC, we adjust the distortion parameters to increase shift intensity while staying in a similar SSIM range as the others. For *view changes* and *semantics*, we render 32k images with smoothly changing parameters, e.g. roll angle, using the Replica [87] dataset.

Figure 4.4 shows example corruptions with different shift intensities.

4.3.3 Applying 3DCC to standard vision datasets

While we employed datasets with full scene geometry information such as Taskonomy [88], 3DCC can also be applied to standard datasets without 3D information. We exemplify this on ImageNet [210] and COCO [211] validation sets by leveraging depth predictions from the MiDaS [268] model, a state-of-the-art depth estimator. Figure 4.5 shows example images with *near focus*, *far focus*, and *fog 3D* corruptions. Generated images are physically plausible, demonstrating that 3DCC can be used for other datasets by the community to generate a diverse set of image corruptions. In Sec. 4.5.2, we quantitatively demonstrate the effectiveness of using predicted depth to generate 3DCC.

4.4 3D Data Augmentation

While benchmarking uses corrupted images as *test data*, one can also use them as augmentations of *training data* to build invariances towards these corruptions. This is the case for us since, unlike 2DCC, 3DCC is designed to capture corruptions that are more likely to appear in the real world, hence it has a sensible augmentation value as well. Thus, in addition to benchmarking robustness using 3DCC, our framework can also be viewed as new data augmentation strategies that take the 3D scene geometry into account. We augment with the following corruption types in our experiments: *depth of field*, *camera motion*, and *lighting*. The augmentations can be efficiently generated on-the-fly during training using parallel implementations. For example, the depth of field augmentations take 0.87 seconds (wall clock time) on a single V100 GPU for a batch size of 128 images with 224×224 resolution. For comparison, applying 2D defocus blur requires 0.54 seconds, on average. It is also possible to precompute certain selected parts of the augmentation process, e.g. the illuminations for lighting augmentations, to increase efficiency. We incorporated these mechanisms in our implementation. We show in Sec. 4.5.3 that these augmentations can significantly improve robustness against real-world distortions.

4.5 Experiments

We perform evaluations to demonstrate that 3DCC can expose vulnerabilities in models (Sec. 4.5.2) that are not captured by 2DCC (Sec. 4.5.2). The generated corruptions are similar to expensive realistic synthetic ones (Sec. 4.5.2) and are applicable to datasets without 3D information (Sec. 4.5.2) and for semantic tasks (Sec. 4.5.2). Finally, the proposed 3D data augmentation improves robustness qualitatively and quantitatively (Sec. 4.5.3).

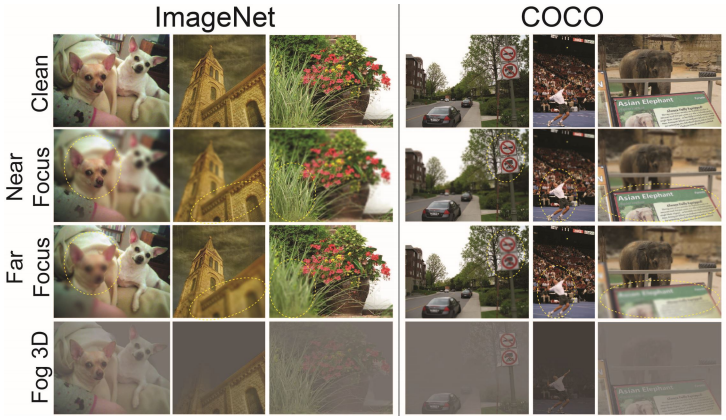

Figure 4.5: **3DCC can be applied to most datasets**, even those that do not come with 3D information. Several query images from the ImageNet [210] and COCO [211] dataset are shown with *near focus*, *far focus* and *fog 3D* corruptions applied. Notice how the objects in the circled regions go from sharp to blurry depending on the focus region and scene geometry. To get the depth information needed to create these corruptions, predictions from MiDaS [268] model is used. This gives a good enough approximation to generate realistic corruptions (as we will quantify in Sec. 4.5.2).

4.5.1 Preliminaries

Evaluation Tasks: 3DCC can be applied to any dataset, irrespective of the target task, e.g. dense regression or low-dimensional classification. Here we mainly experiment with surface normals and depth estimation as target tasks widely employed by the community. We note that the robustness of models solving such tasks is underexplored compared to classification tasks (See Sec. 4.5.2 for results on panoptic segmentation and object recognition). To evaluate robustness, we compute the ℓ_1 error between predicted and ground truth images.

Training Details: We train UNet [94] and DPT [214] models on Taskonomy [88] using learning rate 5×10^{-4} and weight decay 2×10^{-6}. We optimize the likelihood loss with Laplacian prior using AMSGrad [95], following [142]. Unless specified, all the models use the same UNet backbone (e.g. Fig. 4.6). We also experiment with DPT models trained on Omnidata [132] that mixes a diverse set of training datasets. Following [132], we train with learning rate 1×10^{-5}, weight decay 2×10^{-6} with angular & ℓ_1 losses.

Robustness mechanisms evaluated: We evaluate several popular data augmentation strategies: DeepAugment [7], style augmentation [96], and adversarial training [10]. We also include Cross-Domain Ensembles (X-DE) [142] that has been recently shown to improve robustness to corruptions by creating diverse ensemble components via input transformations. We refer to the appendix for training details. Finally, we train a model with augmentation with corruptions from 2DCC [2] (2DCC augmentation), and another model with 3D data augmentation on top of that (2DCC + 3D augmentation).

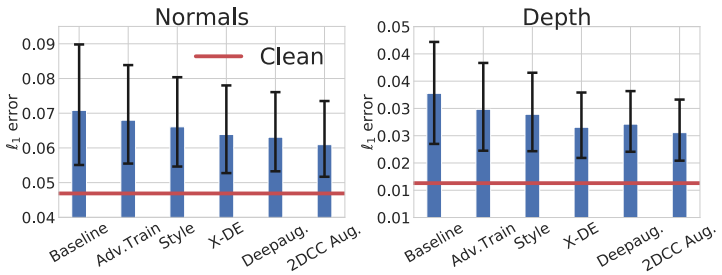

Figure 4.6: **Existing robustness mechanisms are found to be insufficient for addressing real-world corruptions approximated by 3DCC.** Performance of models with different robustness mechanisms under 3DCC for surface normals (left) and depth (right) estimation tasks are shown. All models here are UNets and are trained with Taskonomy data. Each bar shows the ℓ_1 error averaged over all 3DCC corruptions (lower is better). The black error bars show the error at the lowest and highest shift intensity. The red line denotes the performance of the baseline model on clean (uncorrupted) data. This denotes that existing robustness mechanisms, including those with diverse augmentations, perform poorly under 3DCC.

4.5.2 3D Common Corruptions Benchmark

3DCC can expose vulnerabilities

We perform a benchmarking of the existing models against 3DCC to understand their vulnerabilities. However, we note that our main contribution is not the performed analyses but the benchmark itself. The state-of-the-art models may change over time and 3DCC aims to identify the robustness trends, similar to other benchmarks.

Effect of robustness mechanisms: Figure 4.6 shows the average performance of different robustness mechanisms on 3DCC for surface normals and depth estimation tasks. These mechanisms improved the performance over the baseline but are still far from the performance on clean data. This suggests that 3DCC exposes robustness issues and can serve as a challenging testbed for models. The 2DCC augmentation model returns slightly lower ℓ_1 error, indicating that diverse 2D data augmentation only partially helps against 3D corruptions.

Effect of dataset and architecture: We provide a detailed breakdown of performance against 3DCC in Fig. 4.7. We first observe that baseline UNet and DPT models trained on Taskonomy have similar performance, especially on the view change corruptions. By training with larger and more diverse data with Omnidata, the DPT performance improves. Similar observations were made on vision transformers for classification [269, 138]. This improvement is notable with view change corruptions, while for the other corruptions, there is a decrease in error from 0.069 to 0.061. This suggests that combining architectural advancements with diverse and large training data can play an important role in robustness against 3DCC. Furthermore, when combined with 3D augmentations, they improve

robustness to real-world corruptions (Sec. 4.5.3).

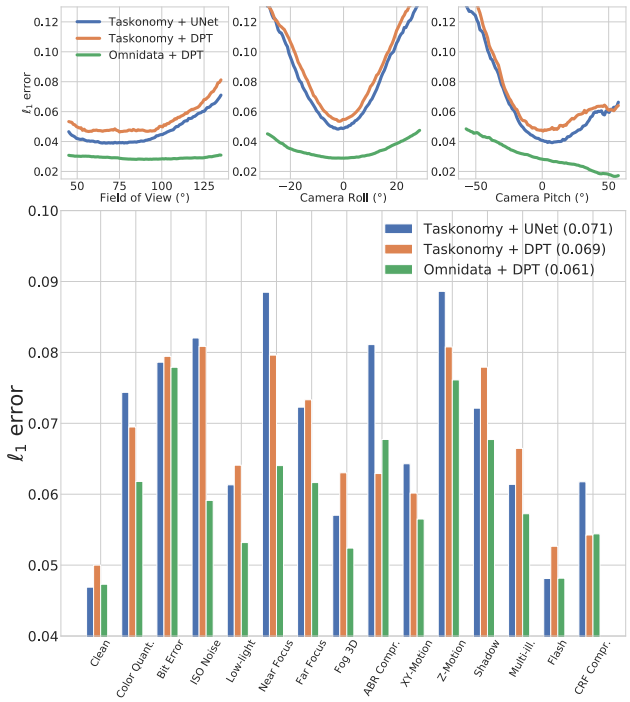

Figure 4.7: **Detailed breakdown of performance on 3DCC.** The benchmark can expose trends and models' sensitivity to a wide range of corruptions. We show this by training models on either Taskonomy [88] or Omnidata [132] and with either a UNet [94] or DPT [214] architecture. The average ℓ_1 error over all shift intensities for each corruption is shown (lower is better). **Top:** We observe that Taskonomy models are more susceptible to changes in field of view, camera roll, and pitch compared to Omnidata trained model, which is consistent with their methods. **Bottom:** The numbers in the legend are the average performance over all the corruptions. We can see that all the models are sensitive to 3D corruptions, e.g. *z-motion blur* and *shadow*. Overall, training with large diverse data, e.g. Omnidata, and using DPT is observed to notably improve performance.

Redundancy of corruptions in 3DCC and 2DCC

In Fig. 4.1, a qualitative comparison was made between 3DCC and 2DCC. The former generates more realistic corruptions while the latter does not take scene 3D into account and applies uniform modifications over the image. In Fig. 4.8, we aim to quantify the similarity between 3DCC and 2DCC. On the left of Fig. 4.8, we compute the correlations of ℓ_1 errors between clean and corrupted predictions made by the baseline model for a subset of corruptions (full set is in the appendix). 3DCC incurs less correlations both intra-benchmark as well as against 2DCC (Mean correlations are 0.32 for 2DCC-2DCC, 0.28 for 3DCC-3DCC, and 0.30 for 2DCC-3DCC). Similar conclusions are obtained for depth estimation (in the appendix). In the right, we provide the same analysis on the

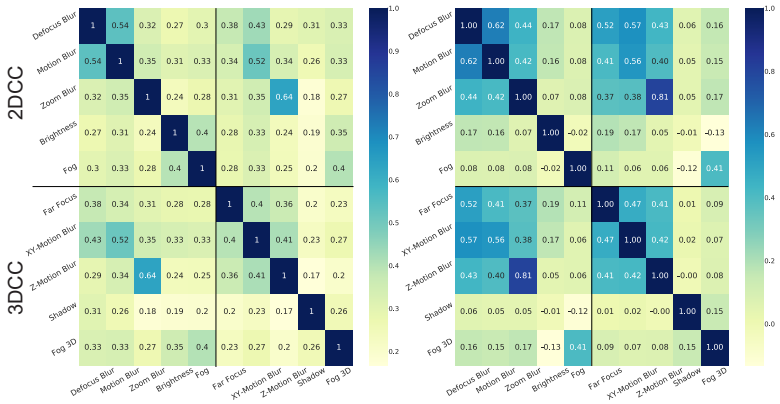

Figure 4.8: **Redundancy among corruptions.** We quantified the pairwise similarity of a subset of corruptions from 2DCC and 3DCC by computing their correlations in the ℓ_1 errors of the surface normals predictions (left) and RGB images (right). 3DCC incurs less correlations both intra-benchmark as well as against 2DCC. Thus, 3DCC has a diverse set of corruptions and these corruptions do not have a significant overlap with 2DCC. Using depth as target task yields similar conclusions (full affinity matrices are provided in the appendix).

RGB domain by computing the ℓ_1 error between clean and corrupted images, again suggesting that 3DCC yields lower correlations.

Figure 4.9: **Visual comparisons of 3DCC and expensive After Effects (AE) generated depth of field effect** on query images from Hypersim. 3DCC generated corruptions are visually similar to those from AE.

Soundness: 3DCC vs Expensive Synthesis

3DCC aims to expose a model's vulnerabilities to certain real-world corruptions. This requires the corruptions generated by 3DCC to be similar to real corrupted data. As generating such labeled data is expensive and scarcely available, as a proxy evaluation, we instead compare the realism of 3DCC to synthesis made by Adobe After Effects (AE) which is a commercial product to generate high-quality photorealistic data and often relies on expensive and manual processes. To achieve this, we use the Hypersim [213] dataset that comes with high-resolution z-depth labels. We then generated 200 images that are near- and far-focused using 3DCC and AE. Figure 4.9 shows sample generated images from both approaches that are perceptually similar. Next, we computed the prediction errors of a baseline normal model when the input is from 3DCC or AE. The scatter plot of ℓ_1 errors are given in Fig. 4.10 and demonstrates a strong correlation, 0.80, between the two approaches. For calibration and control, we also provide the scatter

plots for some corruptions from 2DCC to show the significance of correlations. They have significantly lower correlations with AE, indicating the depth of field effect created via 3DCC matches AE generated data reasonably well.

Effectiveness of applying 3DCC to other datasets

We showed qualitatively in Fig. 4.5 that 3DCC can be applied to standard vision datasets like ImageNet [210] and COCO [211] by leveraging predicted depth from a state-of-the-art model from MiDaS [268]. Here, we quantitatively show the impact of using predicted depth instead of ground truth. For this, we use the Replica [87] dataset that comes with ground truth depth labels. We then generated 1280 corrupted images using ground truth depth and predicted depth from MiDaS [268] *without fine-tuning on Replica*. Figure 4.11 shows the trends on three corruptions from 3DCC generated using ground truth and predicted depth. The trends are similar and the correlation of errors is strong (0.79). This suggests that the predicted depth can be effectively used to apply 3DCC to other datasets, and the performance is expected to improve with better depth predictions.

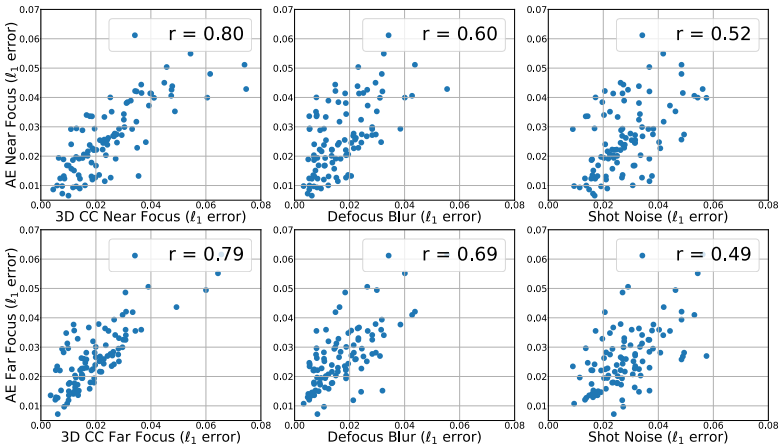

Figure 4.10: **Corruptions of 3DCC are similar to expensive realistic synthetic ones while being cheaper to generate.** Scatter plots of ℓ_1 errors from the baseline model predictions on 3DCC against those created by Adobe After Effects (AE). The correlation between 3DCC near (far) focus and those from AE near (far) focus is the *strongest* (numbers are in the legend of left column). We also added the most similar corruption from 2DCC (defocus blur) for comparison, yielding weaker correlations (middle). Shot noise (right) is a *control baseline*, i.e. a randomly selected corruption, to calibrate the significance of the correlation measure.

3DCC evaluations on semantic tasks

The previous benchmarking results were focusing on surface normals and depth estimation tasks. Here we perform a benchmarking on panoptic segmentation and object recognition tasks as additional illustrative 3DCC evaluations. In particular for panoptic segmentation, we use *semantic corruptions* from Sec. 4.3.1, and for object classification, we introduce

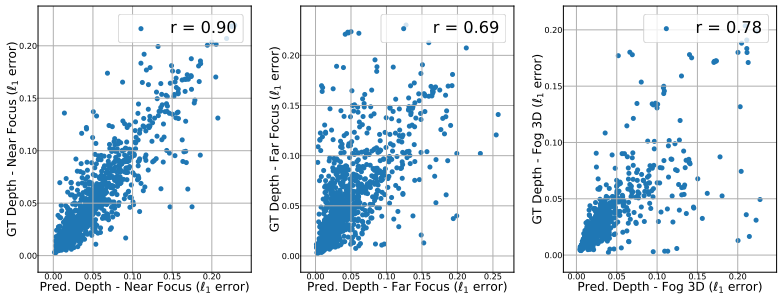

Figure 4.11: **Effectiveness of applying 3DCC without ground truth depth.** Three corruptions from 3DCC are generated using depth predictions from MiDaS [268] model on unseen Replica data. Scatter plots show the ℓ_1 errors from the baseline model when corruptions are generated using the predicted depth (x-axis) or the ground truth (y-axis). The trends are similar between two corrupted data results, suggesting the predicted depth is an effective approximation to generate 3DCC. See the appendix for more tests including control baselines.

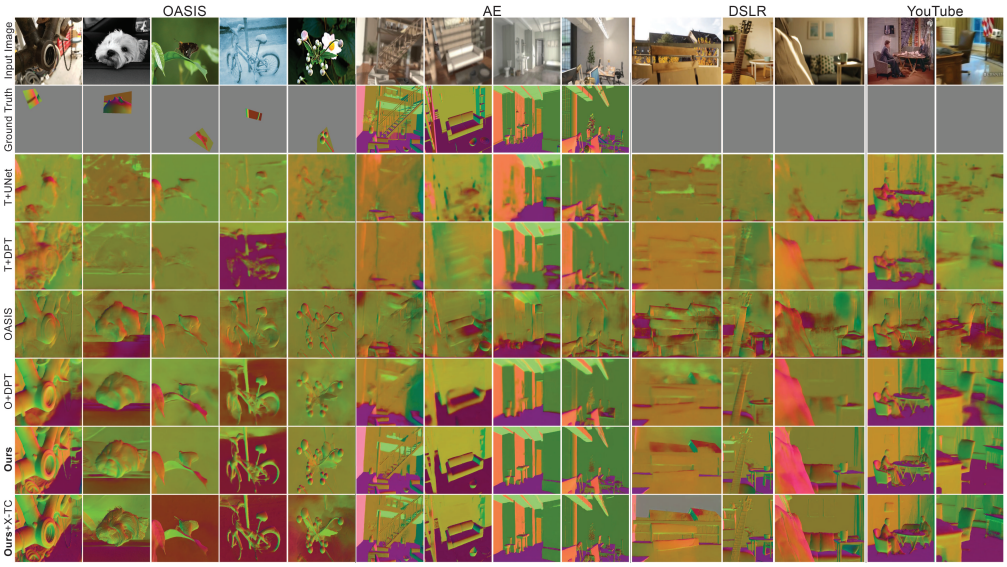

Figure 4.12: **Qualitative results of learning with 3D data augmentation** on random queries from OASIS [270], AE (Sec. 4.5.2), manually collected DSLR data, and in-the-wild YouTube videos for surface normals. The ground truth is gray when it is not available, e.g. for YouTube. The predictions in the last two rows are from the O+DPT+2DCC+3D (Ours) model. It is further trained with cross-task consistency (X-TC) constraints [72] (Ours+X-TC). They are noticeably sharper and more accurate.

ImageNet-3DCC by applying corruptions from 3DCC to ImageNet validation set, similar to 2DCC [2].

Semantic corruptions: We evaluate the robustness of two panoptic segmentation models from [132] against *occlusion corruption* of 3DCC. The models are trained on Omnidata [132] and Taskonomy [88] datasets with a Detectron [271] backbone. See the

appendix for details.

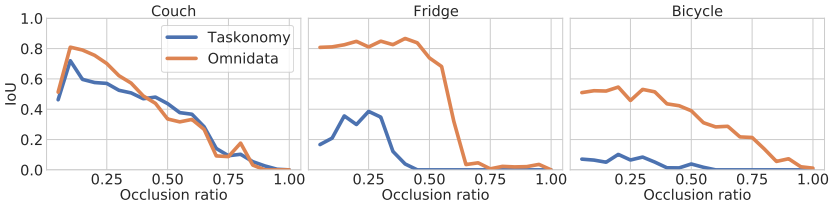

Figure 4.13: **Robustness against occlusion corruption of 3DCC.** The plot shows the intersection over union (IoU) scores of Detectron models [271] for different objects over a range of occlusion ratios. The models are trained on Taskonomy [88] or Omnidata [132] datasets. The occlusion ratio is defined as the number of occluded pixels divided by the sum of occluded and visible pixels of the object. This is computed over the test scenes of Replica [87]. The plots expose the occlusion handling capabilities of the models and show that the Omnidata trained model is generally more robust than the Taskonomy one. The degradation in model predictions is class-specific and becomes more severe with higher occlusion ratios.

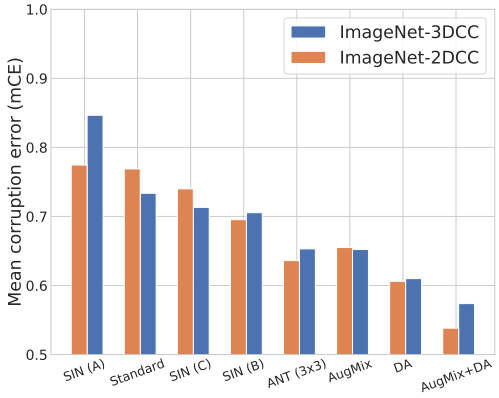

Figure 4.14: **Robustness on ImageNet-3DCC and ImageNet-2DCC.** Errors on ImageNet validation images corrupted by 3DCC and 2DCC are computed for the models in robustness leaderboards [2, 264]. Following [2], we compute the mean corruption error (mCE) relative to AlexNet [101]. The performance degrades significantly against ImageNet-3DCC, thus it can serve as a challenging benchmark. As expected, the general trends are similar between the two benchmarks as 2D and 3D corruptions are not completely disjoint. A similar observation was also made in [230] even when the corruptions are *designed to be dissimilar* to 2DCC. Still, there are notable differences that can be informative during model development by exposing trends and vulnerabilities that are not captured by 2DCC, e.g. ANT [41] has better mCE on 2DCC compared to AugMix [36], while they perform similarly on 3DCC. Likewise, combining DeepAugment [7] with AugMix improved the performance on 2DCC significantly more than 3DCC. See the appendix for more results.

Figure 4.13 quantifies the effect of occlusion on the predictions of models, i.e. how the models' intersection over union (IoU) scores change with increasing occlusion, for selected objects. This is computed on the test scenes from Replica [87]. The Omnidata trained

model is generally more robust than the Taskonomy one, though we see a decrease in IoU in both models as occlusion increases. The trends are class-specific possibly due to shape of the objects and their scene context, e.g. fridge predictions remain unchanged up until 0.50 occlusion ratio, while couch predictions degrade more linearly for Omnidata model. This evaluation showcases one use of semantic corruptions in 3DCC, which are notably harder to accomplish using other benchmarks that do not operate based on 3D scans.

ImageNet-3DCC: We compare performances of the robust ImageNet models [96, 41, 36, 7] from RobustBench [264] and ImageNet-2DCC [2] (i.e. ImageNet-C) leaderboards in Fig. A.39. Following 2DCC, we compute mean corruption error (mCE) by dividing the models errors by AlexNet [101] errors and averaging over corruptions. The performance of models degrade significantly, including those with diverse augmentations. Thus, ImageNet-3DCC can serve as a challenging benchmark for object recognition task. As expected, while the general trends are similar between the two benchmarks as 2D and 3D corruptions are not completely disjoint [230], 3DCC exposes vulnerabilities that are not captured by 2DCC, which can be informative during model development. See the appendix for further results.

4.5.3 3D data augmentation to improve robustness

We demonstrate the effectiveness of the proposed augmentations qualitatively and quantitatively. We evaluate UNet and DPT models trained on Taskonomy (T+UNet, T+DPT) and DPT trained on Omnidata (O+DPT) to see the effect of training dataset and model architecture. The training procedure is as described in Sec. 4.5.1. For the other models, we initialize from O+DPT model and train with 2DCC augmentations (O+DPT+2DCC) and 3D augmentations on top of that (O+DPT+2DCC+3D), i.e. our proposed model. We also further trained the proposed model using cross-task consistency (X-TC) constraints from [72], denoted as (Ours+X-TC) in the results. Lastly, we evaluated a model trained on OASIS training data from [270] (OASIS).

Qualitative evaluations: We consider **i.** OASIS validation images [270], **ii.** AE corrupted data from Sec. 4.5.2, **iii.** manually collected DSLR data, and **iv.** in-the-wild YouTube videos. Figure 4.12 shows that predictions made by the proposed models are significantly more robust compared to baselines.

Quantitative evaluations: In Table 4.1, we compute errors made by the models on 2DCC, 3DCC, AE, and OASIS validation set (no fine-tuning). Again, the proposed models yield lower errors across datasets showing the effectiveness of augmentations. Note that robustness against corrupted data is improved *without sacrificing performance on in-the-wild clean data*, i.e. OASIS.

BenchmarkModel	T+UNet	T+DPT	OASIS [270]	O+DPT	O+DPT+2DCC	**Ours**	**Ours**+X-TC [72]
2DCC [2] (ℓ_1 error)	8.15	7.47	15.31	6.43	5.78	5.32	**5.29**
3DCC (ℓ_1 error)	7.08	6.89	15.11	6.13	5.94	5.42	**5.35**
AE (Sec. 4.5.2) (ℓ_1 error)	12.86	12.39	16.85	7.84	6.50	**4.94**	5.47
OASIS [270] (angular error)	30.49	32.13	24.63	24.42	**23.67**	24.65	23.89

Table 4.1: **Effectiveness of 3D augmentations quantified using different benchmarks.** ℓ_1 errors are multiplied by 100 for readability. The O+DPT+2DCC+3D model is denoted by Ours. We also trained this model using cross-task consistency (X-TC) constraints from [72] (Ours+X-TC). Our models yield lower errors across the benchmarks. 2DCC and 3DCC are applied on the same Taskonomy test images. More results are given in the appendix. Evaluations on OASIS dataset sometimes show a large variance due to its sparse ground truth.

4.6 Conclusion and Limitations

We introduce a framework to test and improve model robustness against real-world distribution shifts, particularly those centered around 3D. Experiments demonstrate that the proposed 3D Common Corruptions is a challenging benchmark that exposes model vulnerabilities under real-world plausible corruptions. Furthermore, the proposed data augmentation leads to more robust predictions compared to baselines. We believe this direction opens up a promising direction in robustness research by showing the usefulness of 3D corruptions in benchmarking and training. Below we briefly discuss some of the limitations:

3D quality: 3DCC is upper-bounded by the quality of 3D data. The current 3DCC is an imperfect but useful *approximation* of real-world 3D corruptions, as we showed. The fidelity is expected to improve with higher resolution sensory data and better depth prediction models.

Non-exhaustive set: Our set of 3D corruptions and augmentations are *not exhaustive*. They instead serve as a starter set for researchers to experiment with. The framework can be employed to generate more domain-specific distribution shifts with minimal manual effort.

Large-scale evaluation: While we evaluate some recent robustness approaches in our analyses, our main goal was to show that 3DCC successfully exposes vulnerabilities. Thus, performing a comprehensive robustness analysis is beyond the scope of this work. We encourage researchers to test their models against our corruptions.

Balancing the benchmark: We did not explicitly balance the corruption types in our benchmark, e.g. having the same number of noise and blur distortions. Our work can further benefit from weighting strategies trying to calibrate average performance on corruption benchmarks, such as [272].

Use cases of augmentations: While we focus on robustness, investigating their usefulness

on other applications, e.g. self-supervised learning, could be worthwhile.

Evaluation tasks: We experiment with dense regression tasks. However, 3DCC can be applied to different tasks, including classification and other semantic ones. Investigating failure cases of semantic models against, e.g. on smoothly changing occlusion rates for several objects, using our framework could provide useful insights.

This chapter is based on the paper: O. F. Kar, T. Yeo, A. Atanov, A. Zamir, *3D Common Corruptions*, CVPR 2022. I contributed to generating and evaluating on the semantic corruptions, several other qualitative evaluations and quantitative evaluations on OASIS.

Conclusion

This thesis touched upon three directions for making models useful in the real world with a focus on improving robustness. For the first direction, we discussed robustness mechanisms. There, we demonstrated how ensembling predictions from diverse cues can be used to improve the final prediction. We also showed how generative models can be used to generate targeted training data. For the second direction on adaptation mechanisms, we presented an approach for efficient adaptation at test-time. Finally, for the last direction, we proposed realistic corruptions that can be used as an evaluation benchmark or as augmentations. However, this thesis has by no means fully addressed its goal of making models more robust and adaptive. We will now discuss its limitations and future work.

Limitations and Future Work

Focus on low level shifts like common corruptions. In all chapters, the bulk of our evaluations were on (3D) Common Corruptions, followed by several cross-dataset evaluations and unlabelled random images or videos from the internet. While many works focus on robustness to corruptions, distribution shifts that can be encountered in the real world are rich, complex, and not limited to these shifts. At the high level, there can be unseen shifts, e.g., new geographic locations, spurious correlations due to e.g., objects co-occurring with certain backgrounds in the training data, or underrepresented subpopulations. Furthermore, there have been massive efforts in the last few years in putting together datasets with different kinds of shifts into an easy-to-use evaluation framework e.g., WILDS [18, 273], Shift happens benchmark[1]. The methods proposed in this thesis may be able to handle a wider range of shifts than what we have considered, performing such evaluations would be interesting future work.

Benefits of scale. In Chapter 4, we saw that training a transformer model on a mixture of datasets with heavy data augmentations resulted in significantly improved predictions, and at times outperforming the predictions of the proposed methods in the other chapters. This is in line with other recent works on classification that show that large pre-training datasets and models improve robustness [102, 103]. This begs the question, how far

[1] https://github.com/shift-happens-benchmark/icml-2022

can scaling up take us and when do we need carefully curated inductive biases e.g., ensembling from diverse cues with explicit merging mechanisms? In this thesis, we have implicitly assumed that the benefits of the latter, i.e., a more interpretable framework that is less likely to fail silently, outweigh the benefits of a potentially more accurate system that is less interpretable and may fail silently. However, which to choose likely depends on context and application.

Efficient and continuous adaptation via universal representations. In Chapter 3, we introduced a method for efficient adaptation. This is a promising direction, as it is based on the premise that we are not able to anticipate all possible shifts, thus, one should learn to adapt at test-time. Furthermore, there can also be shifts in the target task. Our current models require substantial compute and data to adapt to new domains and tasks. To do well on a single task/domain, the system learns to be invariant to attributes that are irrelevant to the task/domain. However, on unseen tasks/domains, the seen task's/domain's irrelevant attribute may be necessary for the new unseen task.

Thus, decomposing the input into different attributes can allow for more efficient adaptation to new tasks/domains. Having decomposed the input, a learnable function is used to select relevant attributes, depending on their relevance to the target task. Independent modules can be used to process the different attributes and later aggregated to make a final prediction. Thus, this results in a system that is potentially more robust as the failure of one module is independent of the others. It can also be more efficient as adapting to new tasks/domains only requires learning which attributes to use as opposed to fine-tuning the entire network.

Language as an inductive bias. Language encodes some information about how we see and understand the world, it may provide useful inductive biases that can make our models more aligned with how we make decisions or perceive the world. In Chapter 2, we made use of text-to-image models to generate training data. Controlling the generation in the text space can allow us to generate data that is potentially more interpretable. Another example is if performing the decomposition described above using a visual-language model to attain more interpretable attributes would result in a model that is more robust.

Active perception involves interaction and exploration to develop a better understanding of the world. It allows for active data collection to e.g., minimize the uncertainty of our predictions. More concretely, the input to the network can be heavily occluded or corrupted that none of our advances with robustness methods on static images can work. However, being able to change the camera's parameters e.g., viewpoint, or zoom can allow us to get a better view of the object and, thus, make a better prediction. Furthermore, being able to interact with the environment allows for the learning of causal representations. Such representations encode the data generation process or the relationship between events. Thus, predictions based on these representations can be

potentially more robust.

Learning with richer inputs. This thesis mostly focused on learning from static I.I.D. images, however, this is far removed from how we learn. We benefit from a stream of rich, continuous inputs of different modalities [274]. Existing works have shown that this can also be beneficial for neural networks [275, 276]. We are interested in enriching our data by combining different modalities, including non-visual ones e.g., audio, and haptics feedback. Each modality offers different benefits, e.g., audio feedback gives us information about objects or events that are not in our line of sight, and haptic feedback gives us a description of the object in terms of its shape, weight, or material. Thus, the network can learn to integrate these modalities, and in the event that any input modality is corrupted, it is able to use the others to return an accurate prediction.

The distribution shifts that neural networks will encounter in the real world can be rich and complex. Despite the advancements in the community, neural networks remain brittle or only work well under certain shifts. This thesis presented several directions toward the goal of having models that work well in the real world. We hope that it can inspire people to think about the many possible directions for tackling this problem.

A Appendix

A.1 Ensembling diverse predictions

A.1.1 Quantitative Results

Performance on Distorted Data

Performance under common corruptions and dataset shift: Figure A.7 shows the ℓ_1 errors of our method and several baselines for each distortion, for normal, reshading and depth targets. The proposed inverse variance and network merging approaches consistently outperform baselines. In Table A.1 we report results on Replica with common corruptions (Replica-C) and Habitat. They show supportive trends for our method, similar to Taskonomy results in Fig. 6 of main paper.

Method	Replica-C			Habitat
	Normals	**Reshading**	**Depth**	**Depth**
Blind guess	15.0	17.0	5.3	8.1
Baseline	6.2	12.1	3.7	5.2
Deep ensembles	6.0	11.6	3.3	4.6
Ours	**5.1**	**10.1**	**2.4**	**4.3**

Table A.1: **Robustness on common corruptions and dataset shift.** The table provides quantitative evaluation on Replica-C and Habitat. ℓ_1 losses are reported, multiplied by 100 for readability. Our proposed method outperforms the baselines.

Perceptual losses: Similar to the evaluation setup by [72], besides the direct metric, we also report perceptual errors in Figure A.5 for a more complete evaluation. Such metrics evaluate the same prediction in a different representation space (e.g. depth $\rightarrow$ normal) to give a non-uniform weighting to pixels depending on their properties. The proposed method yields notably lower perceptual errors for all target domains compared to baselines.

Additional ablation results: Figure A.6 demonstrates the performance when the proposed method does not use sigma training and training with consistency constraints. This evaluation is performed for both target and perceptual domains. We also provide the results when deep ensembles are trained with consistency constraints. Our method significantly outperforms the baselines in all evaluations. In Figure A.8 we further demonstrate the performance for each distortion.

Performance on Clean Data

In Table A.4, we report quantitative evaluations on the *undistorted* Taskonomy [88] and Replica [87] datasets for the target domains depth, reshading, and surface normals. Our method variants yield similar performance on in-distribution data, while also being more robust against distribution shifts as shown in the main paper. This demonstrates *the robustness on out-of-distribution data did not come at the cost of degraded performance for in-distribution data.* We do not expect robust methods (ours and deep ensembles) to

yield a better performance on in-distribution data as well, since they are not designed to do so and do not have a reason for that. However, we observe that they occasionally fit better to even the in-distribution data, as shown in Table A.4.

				Noise			Blur				Weather				Digital			
	Method	Clean error	Avg.	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
ImageNet	Baseline ResNet-50	24.4	76.2	73.0	74.7	78.8	79.9	92.1	81.5	82.5	75.2	75.6	64.0	59.2	65.3	90.6	74.8	75.9
	Deep ensembles	**21.5**	70.4	67.4	69.7	72.5	73.4	87.4	76.1	**76.9**	70.3	70.3	60.1	52.4	61.7	83.8	66.4	68.3
	Ours	21.6	**67.9**	**66.6**	**68.6**	**71.2**	**71.7**	**82.1**	**75.6**	77.3	**69.1**	**67.2**	**59.1**	**51.3**	**55.8**	**82.1**	**54.4**	**65.9**
CIFAR100	Baseline ResNet-18	24.9	48.6	76.7	68.9	72.4	40.6	50.0	45.8	44.3	41.8	47.2	40.1	30	51.0	37.3	47.1	46.1
	Deep ensembles	21.0	44.1	74.1	65.2	70.1	35.9	**45.2**	**40.8**	**39.4**	37.1	42.3	**35.7**	25.5	47.0	**32.4**	42.2	40.4
	Ours	**20.3**	**41.8**	**69.8**	**60.5**	**59.5**	**35.8**	**45.2**	41.2	40.0	**36.9**	**40.7**	37.1	**23.9**	**46.6**	32.7	**39.0**	**36.4**
CIFAR10	Baseline ResNet-26	7.9	26.3	55.5	44.8	20.5	19.9	26.6	26.7	24.7	18.4	21.5	18.2	10.3	30.8	18.3	32.8	20.2
	Deep ensembles	**6.5**	23.3	46.5	36.1	33.9	18.4	22.7	23.7	22.3	16.3	18.0	15.0	**7.9**	28.0	15.1	29.1	16.8
	Ours	6.7	**20.7**	40.5	31.6	**27.9**	15.4	**19.9**	21.5	18.4	**15.8**	**16.3**	16.0	**8.0**	25.8	**13.8**	26.2	**13.9**

Table A.2: **Robustness on ImageNet-C, CIFAR-100-C, and CIFAR-10-C.** Error on clean and distorted data. For all metrics, lower is better. All methods are trained only on clean ImageNet, CIFAR-100, and CIFAR-10 training data, respectively. Our method performs noticeably better compared to deep ensembles and a single model baseline ResNet.

Additional ablations with deep ensembles

The key role of middle domains can be further seen from Figure A.2. It shows the error against distribution shift when deep ensembles is trained with consistency constraints (*cons*) and sigma training (*ST*) for Reshade task. Merging is done with both the proposed uncertainty based weighting (*inv. var. merging*) and the average of the predictions (*uniform merging*). Thus, the only remaining difference between deep ensembles and ours is in utilizing middle domains. In both cases, our method clearly outperforms.

Path Importance

Figure A.9 shows the performance for individual distortions as a function of number of paths. The proposed method consistently outperforms deep ensembles and the performance improves with more paths. In Table A.3 and Figure A.4, we further investigate the importance of each employed path and best performing paths for the depth, reshading, and normals prediction.

Effect of Employed Distortions for Sigma Training

Here we investigate the effect of using different distortions for the sigma training (ST) step. For this we consider four paths: *rgb, greyscale, sharpened, 2D edges*. We combine the predictions from these paths using the inverse variance merging with three different distortion sets for the ST stage: *ST #1* uses Gaussian blur and Gaussian noise similar to the results in the main paper while *ST #2* uses speckle noise and pixelate distortions and *ST #3* uses pixelate and Gaussian blur distortions. Figure A.1 shows the average ℓ_1 error evaluated on nine *unseen* distortions for these variants. As can be seen, all variants perform similarly, suggesting that the ST step does not require handpicking the training distortions, and can be performed successfully with *any* distortion with sufficiently high intensity.

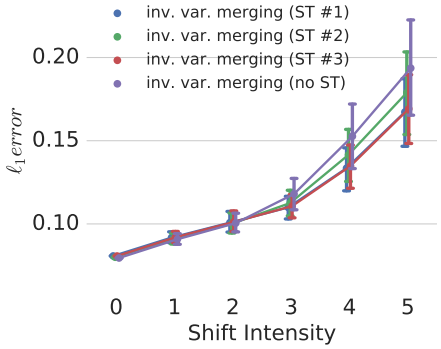

Figure A.1: **Sigma training does not require handpicking the training distortions.** We consider inverse variance merging with four paths (direct, grey, sharpened, edge) for reshading prediction. ST #1 is trained with Gaussian blur and Gaussian noise, while ST #2 is trained with speckle noise and pixelate distortions, and ST #3 is trained with pixelate and Gaussian blur distortions (no ST shows the results when there is no sigma training). The evaluations are performed over nine *unseen* distortions (shot noise, impulse noise, defocus blur, glass blur, contrast, brightness, saturate, jpeg compression, spatter) for these variants. As can be seen, sigma training performs similarly, suggesting that any distortion with sufficiently high intensity works.

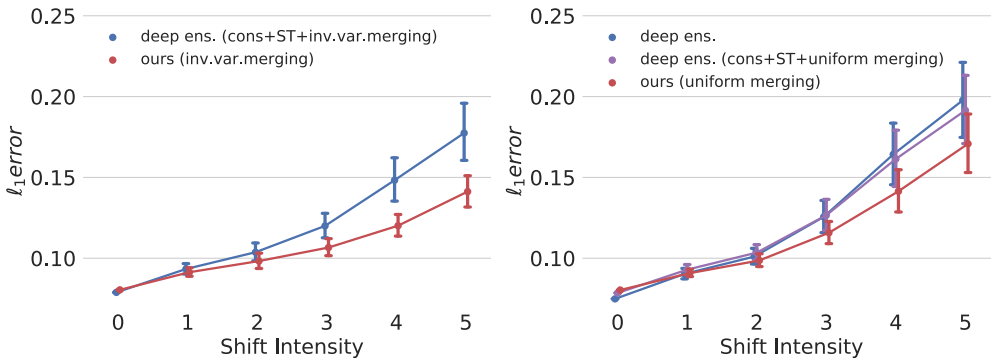

Figure A.2: **Deep ensembles trained with consistency constraints and sigma training (ST).** The only remaining difference between deep ensembles and ours is in utilizing middle domains. Our method outperforms for both *inv. var. merging* (**left**) and *uniform merging* (**right**).

Performance on Imagenet-C, CIFAR-100-C, and CIFAR-10-C

The absolute errors on ImageNet-C, CIFAR-100-C, and CIFAR-10-C are given in Table A.2. For deep ensembles and our method, we use the same backbone as the baseline ResNet model. Our method significantly outperforms the baselines for all three classification tasks.

DomainBest # paths	1	2	3	4	5	6	7	8
Direct	×	D,R	D,R,N	D,R,N	D,R,N	D,R,N	D,R,N	D,R,N
2D Edge	×	×	D,N	D,R,N	D,R,N	D,R,N	D,R,N	D,R,N
Lap. Edge	×	×	×	×	×	D,N	D,N	D,R,N
Emboss	×	N	×	N	D,N	R,N	D,R,N	D,R,N
Greyscale	×	×	R	D	D,R	D,R	D,R,N	D,R,N
Low pass	D,R,N	D,R,N	D,R,N	D,R,N	D,R,N	D,R,N	D,R,N	D,R,N
Sharpened	×	×	×	R	R	R	R	D,R,N
Wavelet	×	×	×	×	N	D,N	D,R,N	D,R,N

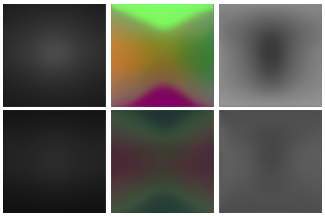

Figure A.3: **Statistically informed guesses ("Blind Guess")** on the Taskonomy dataset for depth, normal, and reshading tasks. Top row shows the predicted mean and the bottom row shows the predicted standard deviation. The predictions minimize the NLL loss on the training dataset.

Method	Taskonomy Dataset									Replica Dataset								
	Depth			Reshade			Normal			Depth			Reshade			Normal		
	Perceptual error		Direct	Perceptual error		Direct	Perceptual error		Direct	Perceptual error		Direct	Perceptual error		Direct	Perceptual error		Direct
	Reshade	Normal	error	Depth	Normal	error	Reshade	Depth	error	Reshade	Normal	error	Depth	Normal	error	Reshade	Depth	error
Blind guess	25.517	21.232	7.951	9.068	19.939	21.639	31.529	5.123	16.169	21.001	22.397	5.307	4.778	16.460	16.986	28.424	3.390	15.012
Baseline UNet	9.227	8.933	2.187	2.829	6.379	6.807	19.376	4.587	4.590	11.133	7.608	2.517	2.252	5.731	8.867	18.211	3.298	4.413
Baseline UNet (cons.)	8.121	7.409	2.198	2.360	5.494	7.252	8.121	1.784	4.957	9.172	6.192	2.090	2.535	5.288	9.051	8.915	1.377	4.405
Multi-domain	10.732	10.548	2.435	2.911	6.483	6.963	18.978	4.633	4.700	12.461	9.606	2.215	2.270	5.954	8.946	17.664	3.489	4.458
Multi-task	11.080	10.542	2.333	3.045	6.742	6.958	22.117	5.194	5.199	11.840	9.270	1.814	1.984	5.778	9.423	22.487	4.516	5.300
Deep ensembles	8.934	8.192	2.207	2.779	5.983	6.470	19.914	4.810	4.448	9.755	6.965	1.925	2.018	5.497	8.513	19.557	3.788	4.305
Deep ensembles (cons.)	8.348	7.338	2.289	2.279	5.184	6.730	8.110	1.960	4.644	9.068	5.871	2.014	2.176	4.966	8.614	9.917	1.874	4.271
Ours (Uniform merging)	8.522	7.510	2.359	2.433	5.435	6.929	8.523	2.338	4.770	8.884	5.960	1.989	2.080	5.029	8.484	10.258	2.264	4.392
Ours (Inv. var. merging)	8.542	7.877	2.301	2.339	5.389	6.787	8.342	2.044	4.685	9.443	6.798	1.980	2.194	5.051	8.451	9.539	1.734	4.220
Ours (Network merging)	8.791	8.123	2.337	2.404	5.456	6.918	8.290	1.946	4.678	9.886	6.986	2.167	2.127	5.036	8.469	9.279	1.529	4.194

Table A.4: **Robustness on out-of-distribution data did not come at the cost of degrading performance on in-distribution data.** The table provides quantitative evaluation on *undistorted* Taskonomy and Replica test sets. Results are reported for depth, reshading, and normal using direct and perceptual error metrics. The perceptual metrics evaluate the target prediction in another domain, similar to [72]. ℓ_1 losses are reported, multiplied by 100 for readability. Our proposed method outperforms on the perceptual metrics while being comparable in the direct metrics, showing that *the performance does not decrease on undistorted data while being robust against distribution shifts.*

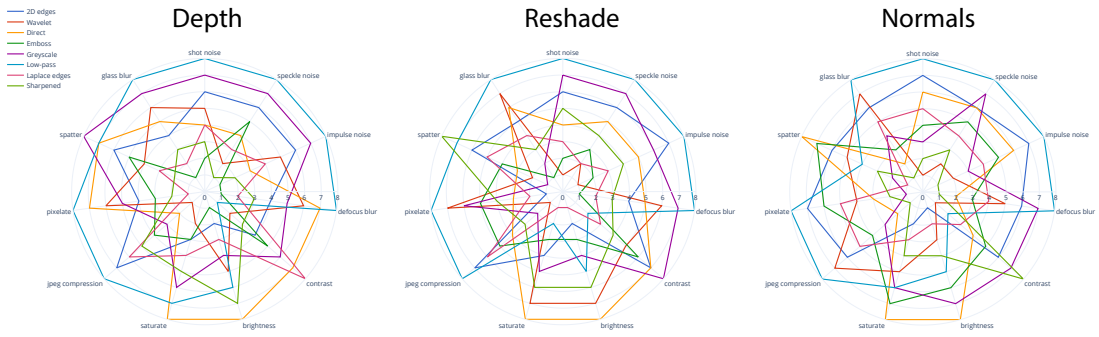

Figure A.4: **Importance of each middle domain for different distortion.** The order of the best performing paths for depth, reshade, and surface normals predictions for each of the distortions, with 8 denoting the most important and 1 the least important one, is shown. This is an extension of Fig. 7b in the main paper with reshade and depth target domains.

A.1.2 Qualitative Results

Video Evaluation

We perform video evaluations on distorted and undistorted video clips to further evaluate the performance of our method against several baselines. We strongly recommend watching them.

Distorted Images and External Queries

We provide qualitative results in Figure A.10 on out of distribution data that is markedly different from those seen during training to demonstrate the generalizability of our method. Furthermore, in Figures A.11,A.12,A.13,A.14 we provide additional prediction results for distorted Replica test images for normals, reshading, and depth prediction.

A.1.3 Further Method Details

Formulation for Learning Based Merging

Multi-modal predictions: For ill-posed problems such as ours, adopting mixture models allows us to capture inherent ambiguity in the data by assuming that there are several possible distributions that could have generated the observed data (e.g. there can be several depth estimates that corresponds to a given RGB image). Given K random variables, their mixing weights $\{\hat{w}_i\}_{i=1}^{K}$ reflect the uncertainty over which of the K variables generated the observed data. From Sec. 3.1 in the main paper, we estimate the parameters of these K distributions, $\{\hat{z}_i, \hat{s}_i\}_{i=1}^{K}$, then function m learns to output the mixing weights given these set of parameters. Our final distribution is a mixture of Laplacians,

$$h_{Mix}(z) \;=\; \sum_i \hat{w}_i h(z|\hat{z}_i, \hat{s}_i) \tag{A.1}$$

where $h(z|\hat{z}_i, \hat{s}_i)$ is the probability density function of a Laplace distribution with mean $\hat{z}_i$ and scale $\exp(\hat{s}_i)$. The final loss is the NLL of the multi-modal prediction $\mathcal{L}_m \;=\; \frac{1}{N}\sum_{n=1}^{N} - \log h_{Mix}(z_n)$.

Mixture-weights: The final prediction is computed as the weighted average of mixture components' means using the weights $\hat{w}_i$ as in Eq. A.1. We approximate the weights for a path to be proportional to its mixture probability distribution function (p.d.f.) evaluated at its estimate $\hat{z}_i$, i.e. $\hat{w}_i \propto h_{Mix}(\hat{z}_i)$. To enforce that h_{Mix} is a p.d.f, we require $\hat{w}_i \geq 0$ and $\sum_i \hat{w}_i = 1$.

Learning based merging: We consider a stacking model [98] that obtains the final predictions given the outputs from each path using the predicted mixture weights $\hat{w}_i$. It has the advantage that the loss is over the entire image, thus, taking into account its spatial structure. This option is denoted as *Network merging*.

Training with Cross-Task Consistency Loss

The proposed ensembling method can be further augmented with "cross-task consistency constraints" [72], to ensure that predictions from the different paths are in cross-task agreement. While this is not a fundamental step for the method, it yields better accuracy especially in fine-grained regions. Following [72], we consider a set of *perceptual* loss networks on the outputs of g. This corresponds to minimizing an ℓ_1 error between the predictions obtained by the model and those from the ground truth in the perceptual domain.

Adversarial Attack/Defence Setup

To generate the I-FGSM attack [10], we apply the following:

$$X_0 = X, \tag{A.2}$$

$$X_{n+1} = Clip_{X,\epsilon}\{X_n + \alpha \text{sign}(\nabla J(X_n, y))\} \tag{A.3}$$

Similar to [10], we set $\alpha = 1$ in our experiments and the number of iterations given by $N = \min(4 + \epsilon, 1.25\epsilon)$. For adversarial training, we finetune the Baseline UNets with an equal number of adversarial and clean examples in the training data. The former were generated from a I-FSGM attack with $\epsilon = (0 - 16]$.

Table A.5 shows that the proposed method outperforms the baselines against corruptions generated with I-FSGM.

Blind Guess Baseline

This baseline is computed using the following formula:

$$g^* = \min_g \mathcal{L}_{NLL} \tag{A.4}$$

The resulting "blind guess" minimizes expected NLL loss on the training dataset. Hence it is a statistically informed guess which does not look at the input for predicting the label. A visualization of these guesses for depth, normal, and reshading are provided in Figure A.3.

A.1.4 Middle domain definitions

Here we define the middle domains. They can be extracted with standard libraries such as OpenCV [93], Scikit-Image [277], or implemented in Pytorch [217]. Our Pytorch implementation for them can be found here.

2D edges are from the output of a Canny edge detector without non-maximum suppression.

	Normal				Reshade				Depth			
Methode	2	4	8	16	2	4	8	16	2	4	8	16
Baseline UNet	8.23	11.53	13.03	14.37	17.92	22.78	27.26	34.40	5.50	6.76	8.36	9.80
Deep ensembles	**7.49**	11.13	13.36	15.65	15.66	21.95	27.75	34.98	5.45	6.68	8.27	10.52
Uniform merging	7.77	9.48	11.00	13.21	**15.50**	18.61	21.32	27.20	**4.85**	5.45	5.97	7.36
Inv. var. merging	7.60	**8.89**	**10.40**	**12.77**	**15.56**	**16.55**	**18.93**	**22.01**	**4.94**	**4.99**	**5.93**	**6.75**
Adv. T. (lower bound error)	5.78	5.74	5.45	5.53	9.39	8.98	8.07	8.20	2.23	2.27	2.39	2.74

Table A.5: **Robustness against adversarial corruptions.** ℓ_1 errors for surface normals, reshade, and depth under adversarial attacks are reported. (Lower is better. Errors are multiplied by 100 for readability.) The proposed method significantly improves robustness against I-FGSM [10] based attacks *without adversarial training*, compared to the baselines. The last row shows the error for a model that has undergone adversarial training [11] with the *same attacks as those evaluated at test time*, hence it gives a lower bound on the error.

Greyscale images are computed by taking the average over the R, G, and B channels of the input image.

Embossed: We perform the filtering using four emboss kernels with different directions and concatenate the embossed outputs.

Laplace edges: Similar to 2D edges, we perform the filtering using a Laplacian filter.

Low-pass filtered: We perform a low pass filtering with a Gaussian filter.

Sharpened: We sharpen the image by subtracting its low-pass filtered version and then adding the original image.

Wavelet: We use the Daubechies-1 with 3 levels of decomposition to obtain wavelet coefficients. We then perform bilinear upsampling on the low resolution coefficients to have the same size as the original image.

A.1.5 Visualizations of Common Corruption on Taskonomy data

Common Corruptions [2] used during evaluation are shown for a sample Taskonomy image in Figure A.15.

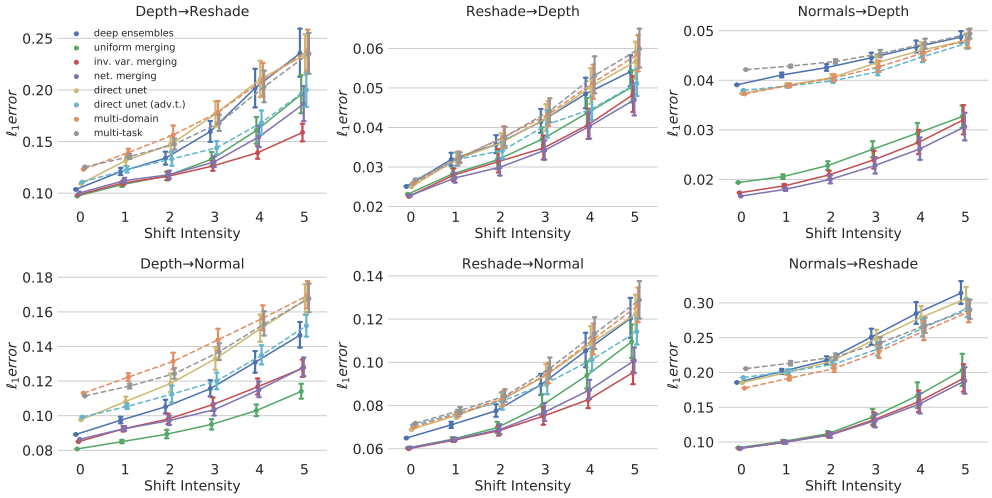

Figure A.5: **Average ℓ_1 loss over 11 unseen distortions in perceptual domains.** Similar to Fig. 6 in the main paper, this shows the performance for unseen distortions where ℓ_1 losses are computed in *perceptual* domains instead, e.g. for the top left plot, loss in the reshade inferred out of the predicted depth is computed. The proposed ensembling approaches are more robust against shift with increasing intensities in perceptual domains as well.

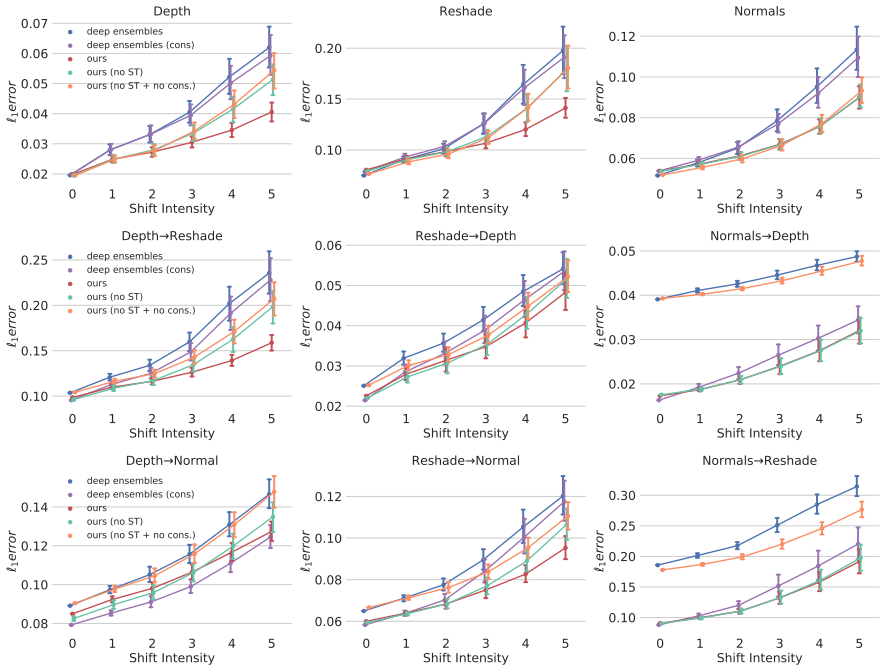

Figure A.6: **Average ℓ_1 loss over 11 unseen distortions in target and perceptual domains.** This shows the performance in target and perceptual domains when the proposed method does not use sigma training or training with consistency constraints.

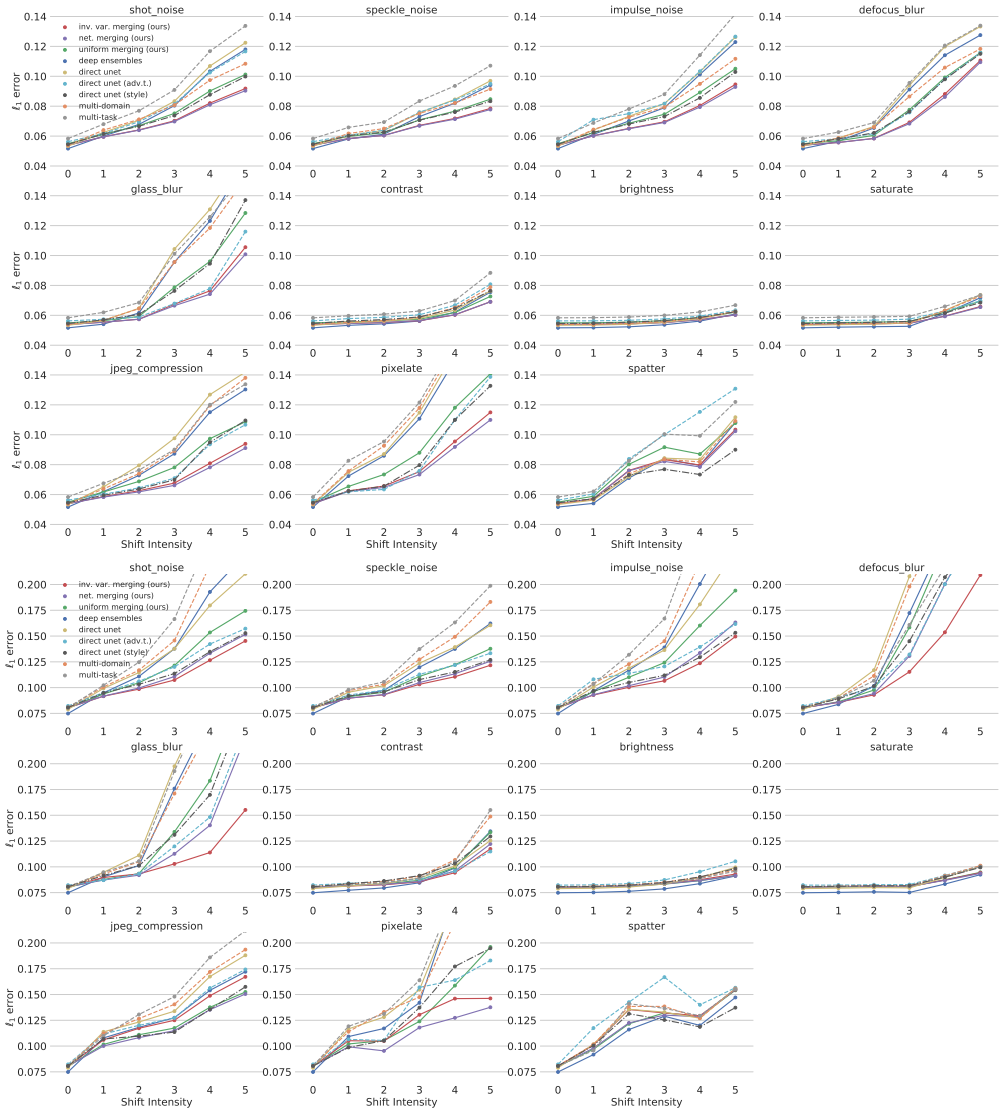

Figure A.7: ℓ_1 **error for each distortion for our method against several baselines for normal and reshade task.** This figure is an extension of Fig. 6 in the main paper for individual distortions.

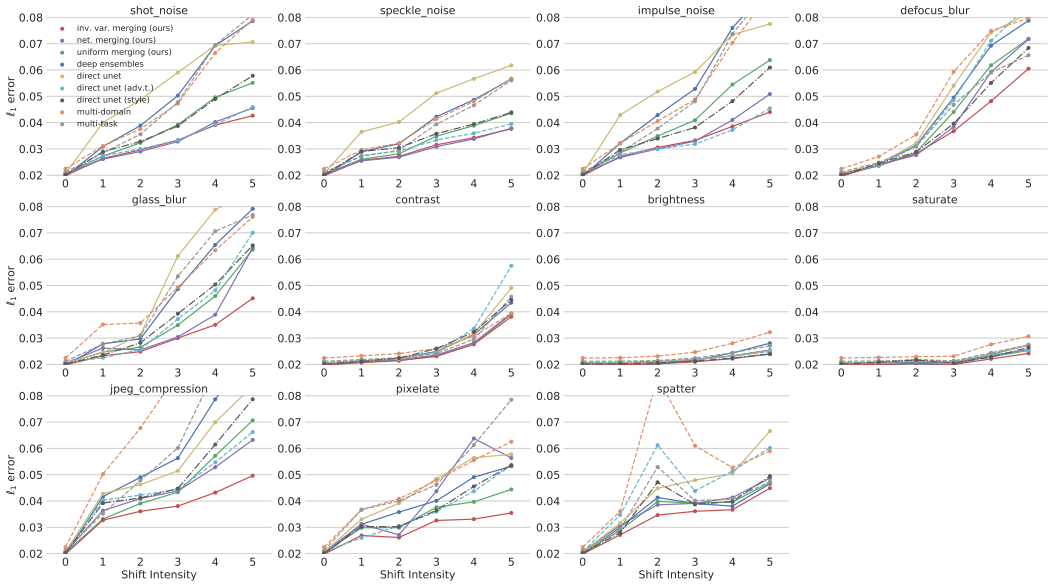

Figure A.7: ℓ_1 **error for each distortion for our method against several baselines for depth task.** This figure is an extension of Fig. 6 in the main paper for individual distortions.

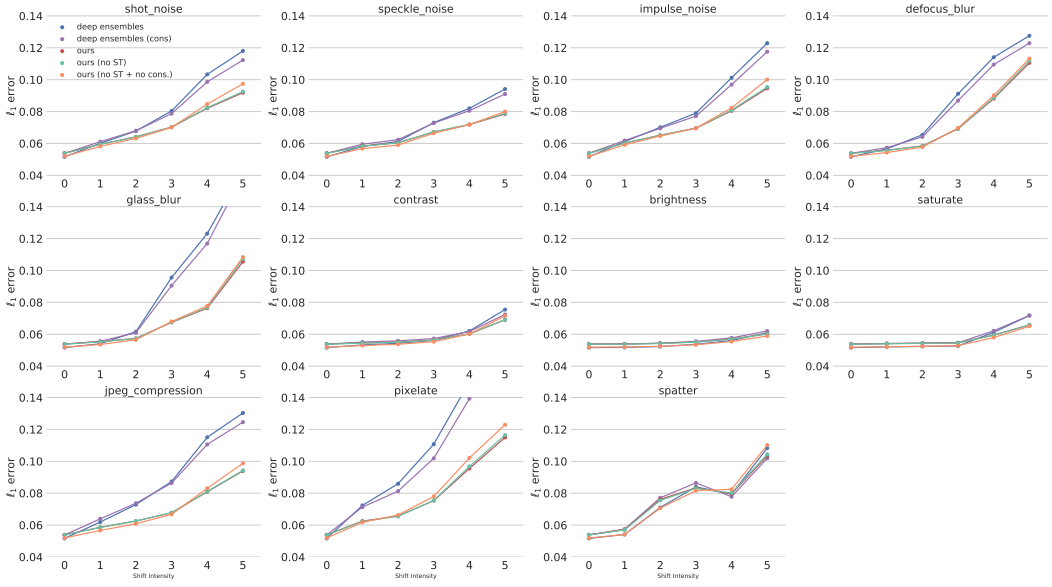

Figure A.8: ℓ_1 **error for each distortion for our method without sigma training and consistency training stages.** This figure is an extension of Fig. 7a (top) in the main paper for individual distortions.

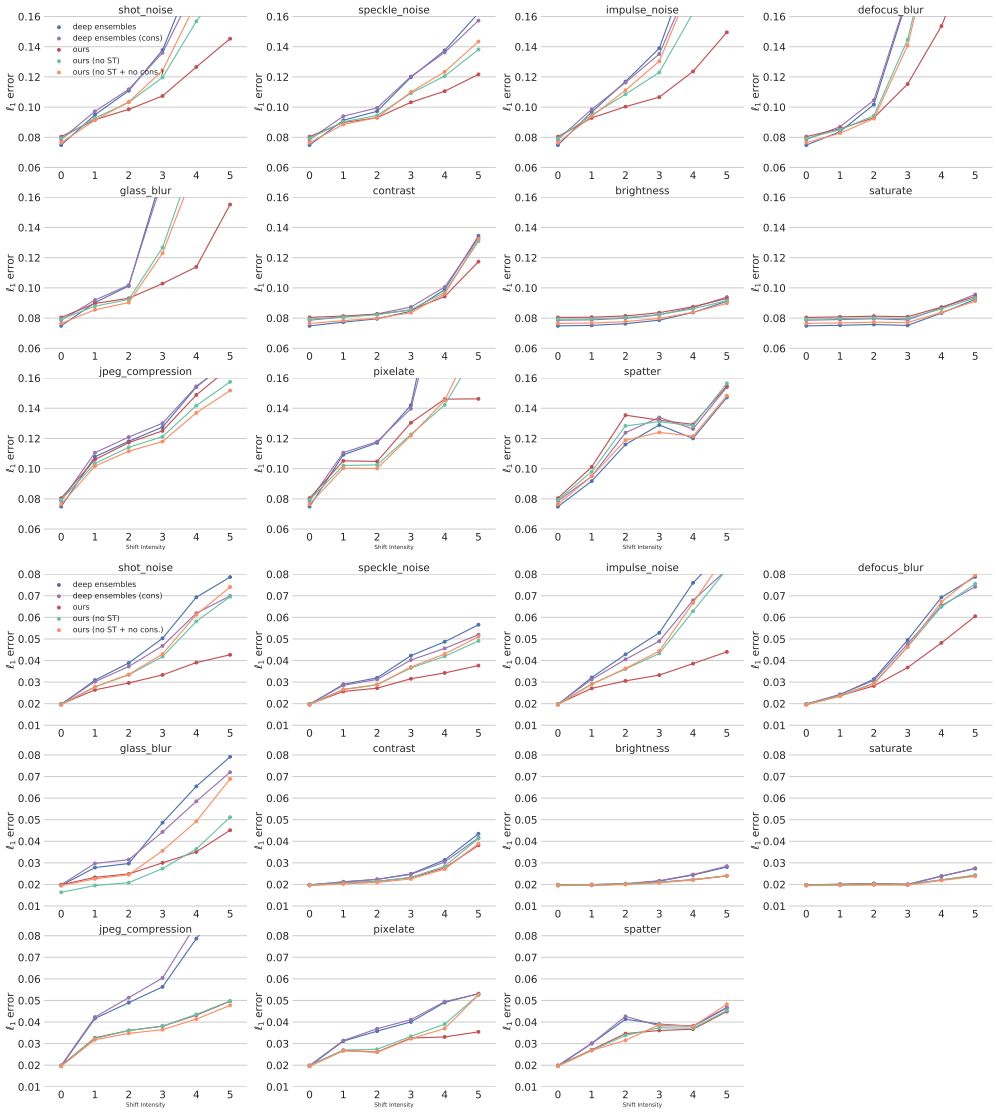

Figure A.8: ℓ_1 **error for each distortion for our method without sigma training and consistency training stages for reshade and depth task.** This figure is an extension of Fig. 7a (top) in the main paper for individual distortions.

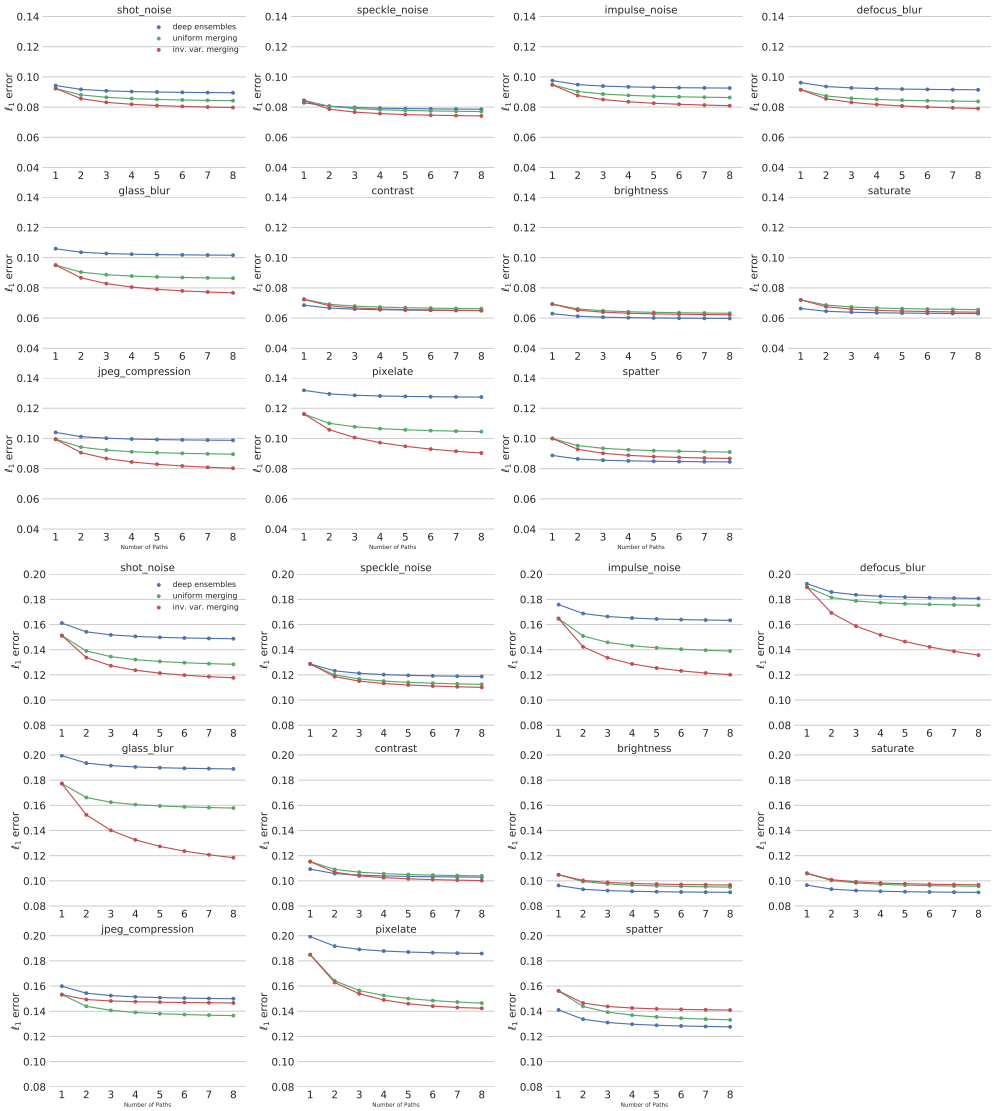

Figure A.9: **Robustness as a function of number of paths** for normal and reshade tasks. This is an extension of Fig. 7a (bottom) in the main paper and shows the ℓ_1 error for each *unseen* distortion as we increase the number of paths. For deep ensembles, this corresponds to increasing the number of ensemble components. The proposed method (*inv. var. merging*) and its simpler variants (*uniform merging*) consistently outperforms deep ensembles.

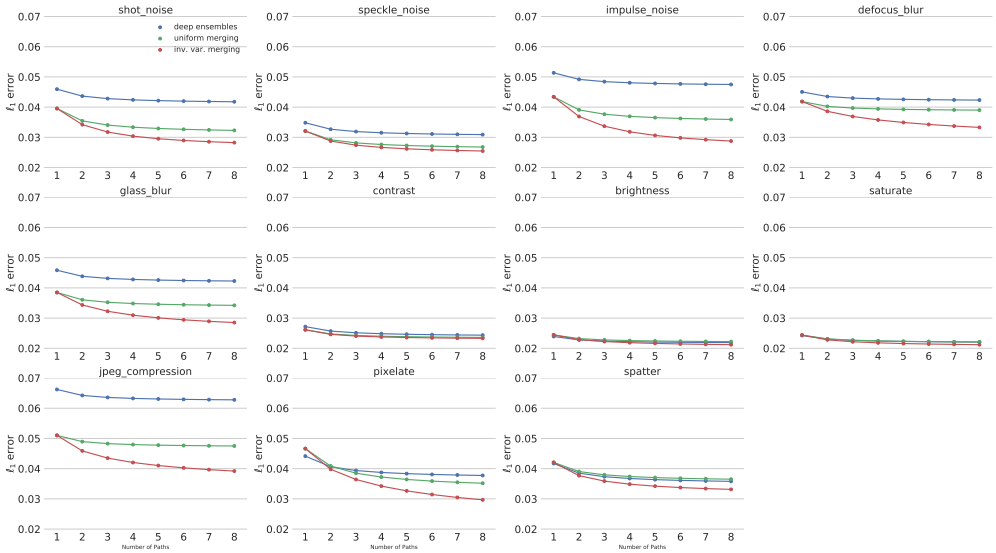

Figure A.9: **Robustness as a function of number of paths** for depth task. This is an extension of Fig. 7a (bottom) in the main paper and shows the ℓ_1 error for each *unseen* distortion as we increase the number of paths. For deep ensembles, this corresponds to increasing the number of ensemble components. The proposed method (*inv. var. merging*) and its simpler variants (*uniform merging*) consistently outperforms deep ensembles.

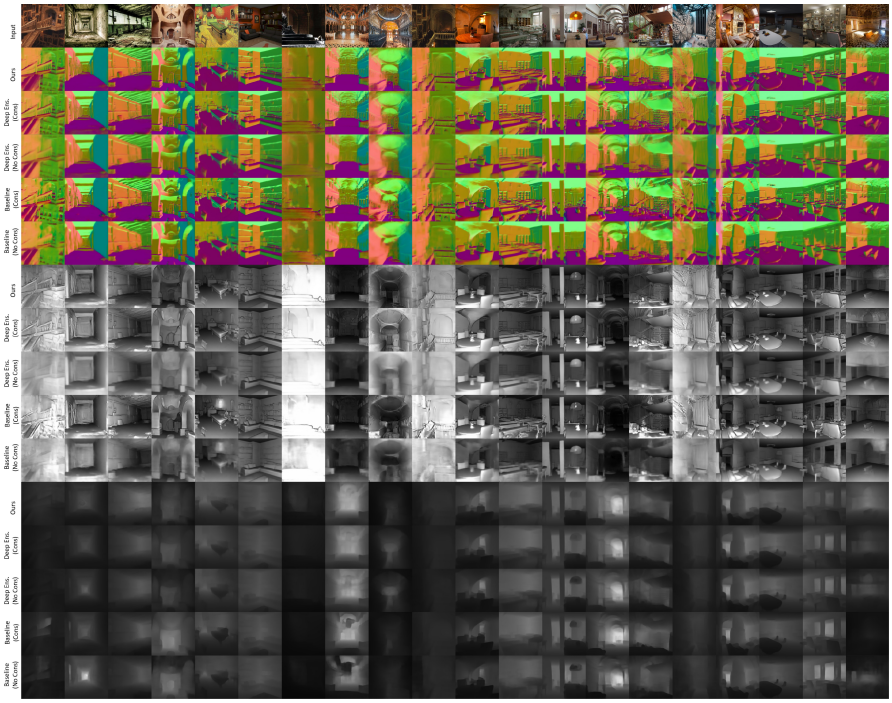

Figure A.10: Qualitative results on *undistorted* external query images for the proposed method and baselines trained with (Cons) and without (No Cons) consistency constraints.

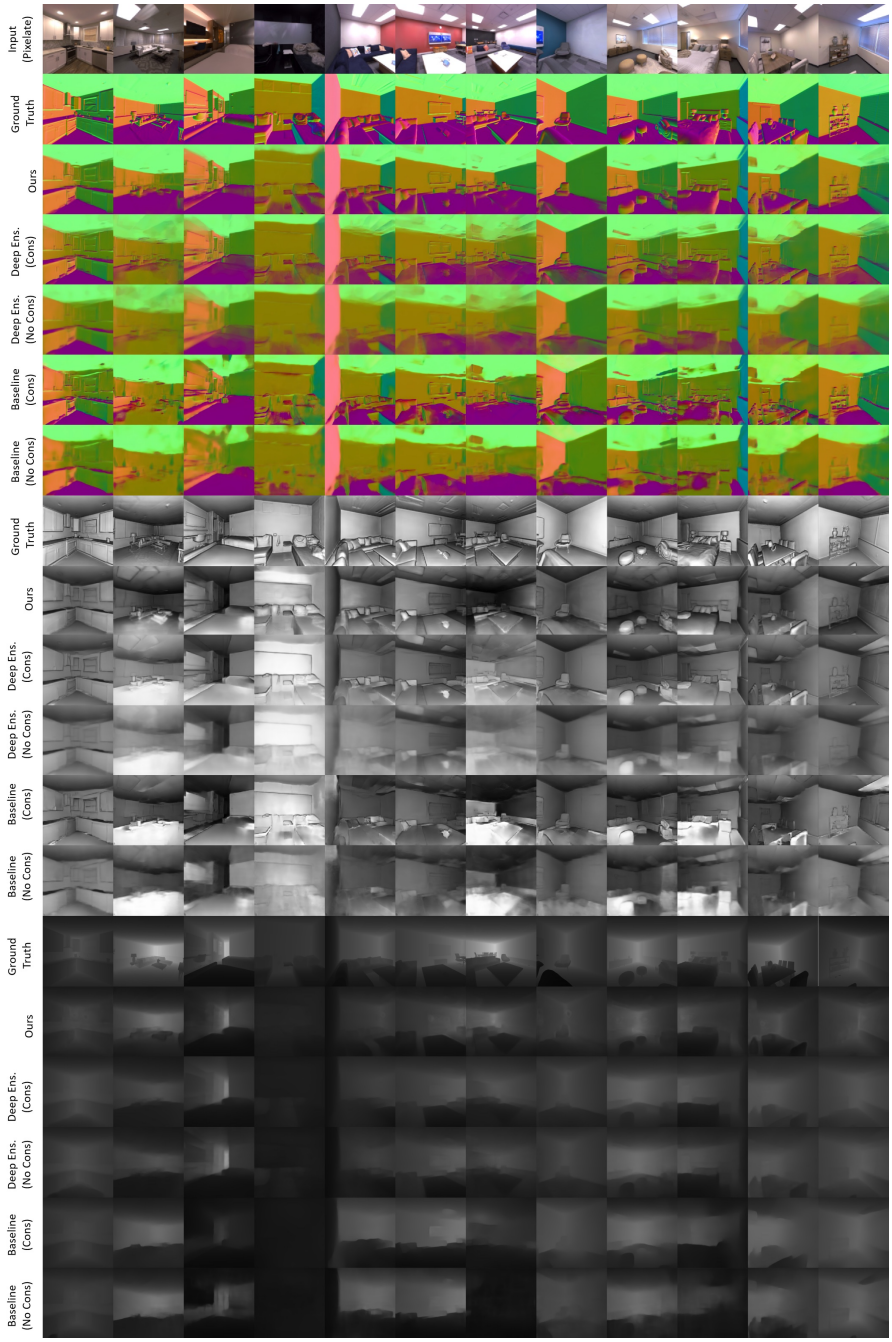

Figure A.11: Qualitative results on Replica images with pixelate distortion (shift intensity 3) for the proposed method and baselines trained with (Cons) and without (No Cons) consistency constraints.

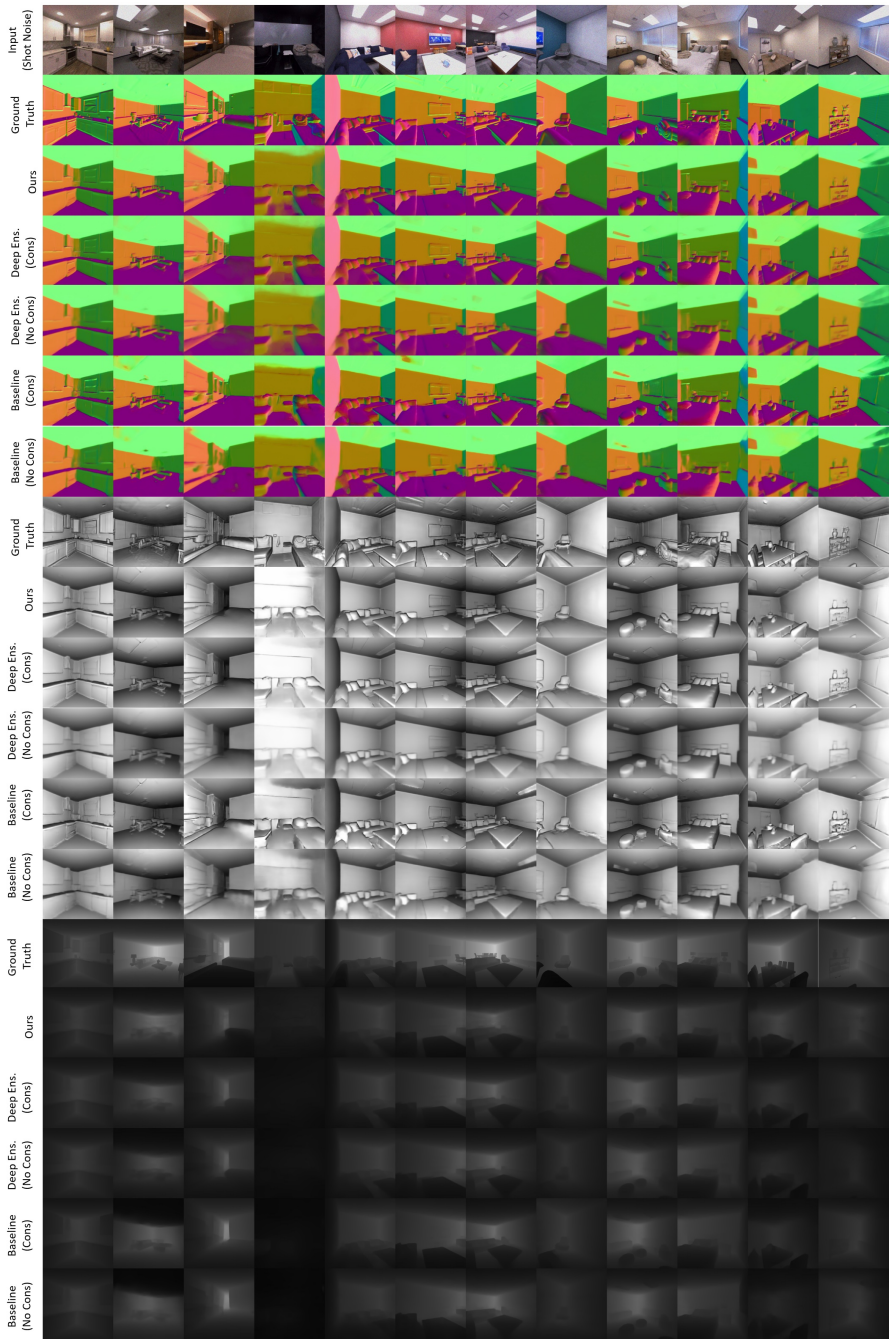

Figure A.12: Qualitative results on Replica images with shot noise distortion (shift intensity 3) for the proposed method and baselines trained with (Cons) and without (No Cons) consistency constraints.

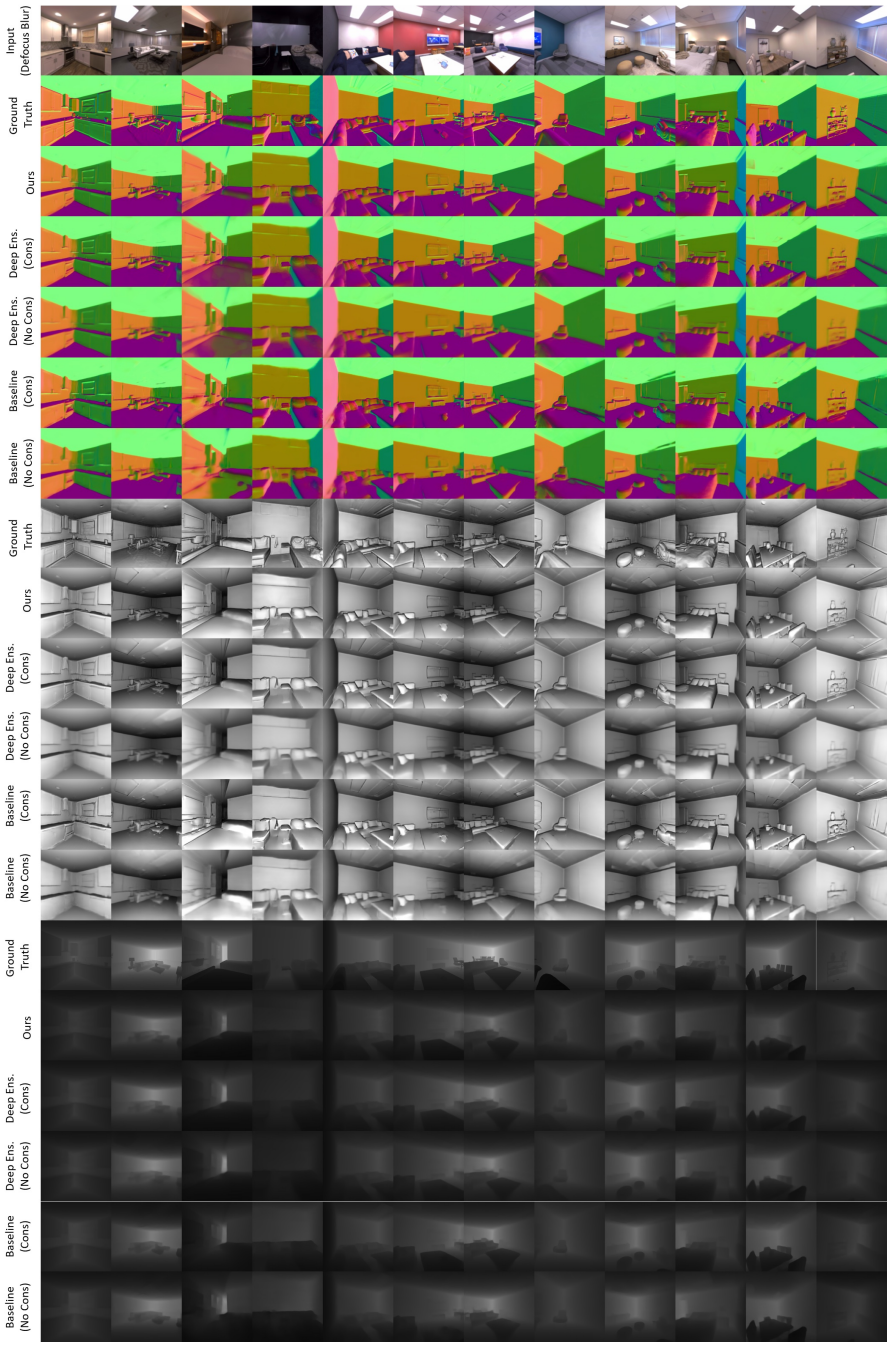

Figure A.13: Qualitative results on Replica images with defocus blur distortion (shift intensity 3) for the proposed method and baselines trained with (Cons) and without (No Cons) consistency constraints.

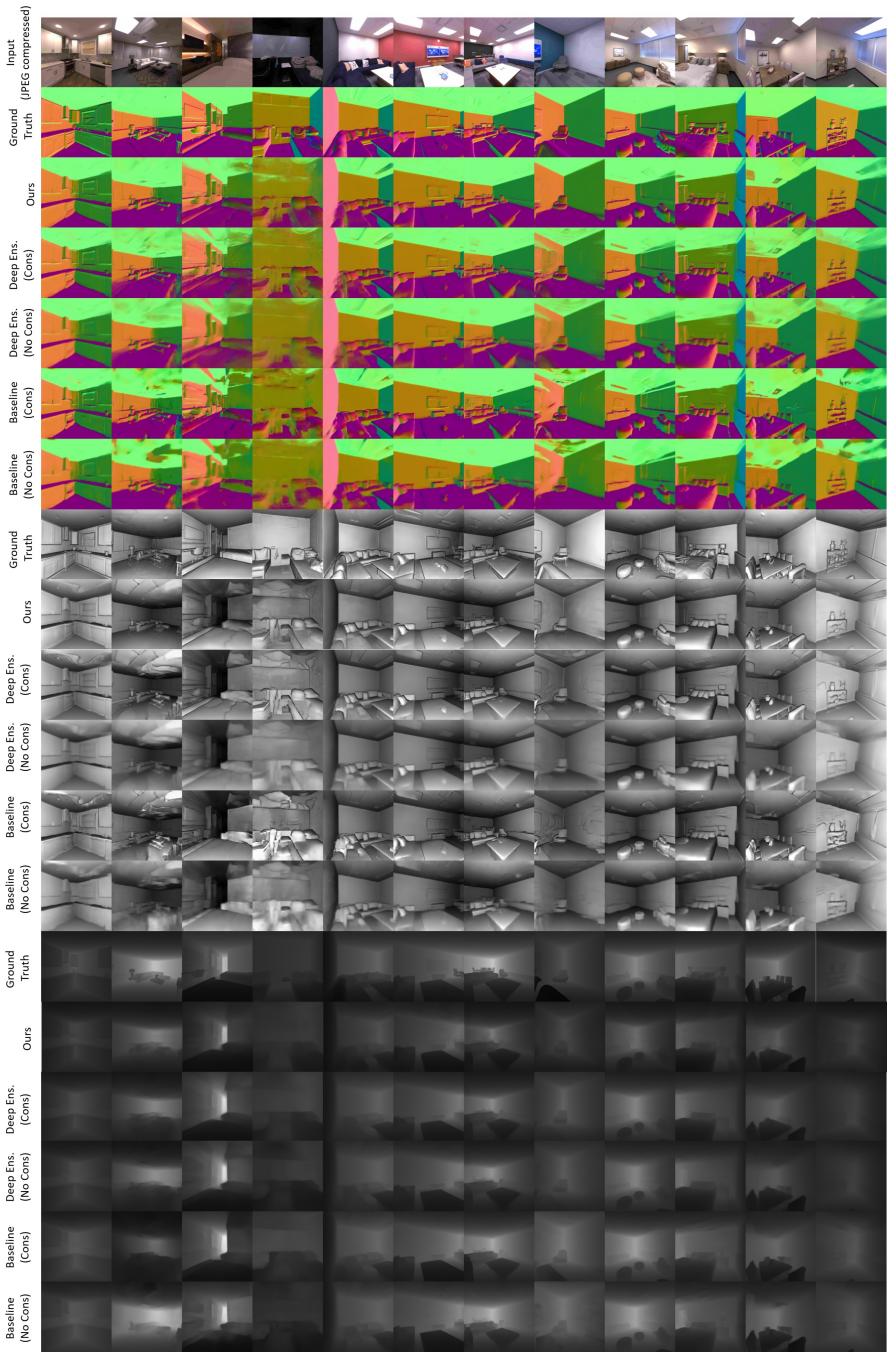

Figure A.14: Qualitative results on Replica images with JPEG distortion (shift intensity 3) for the proposed method and baselines trained with (Cons) and without (No Cons) consistency constraints.

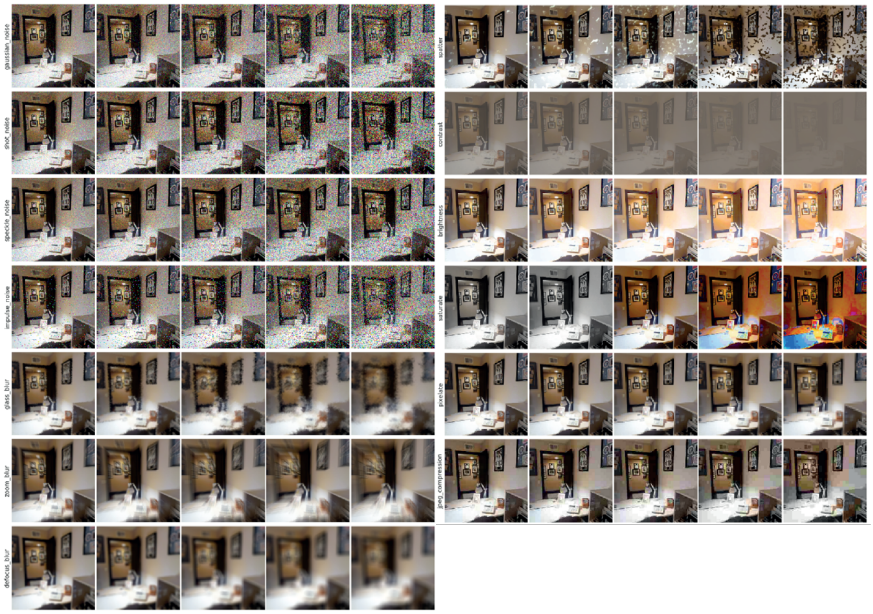

Figure A.15: **Visuals of common corruptions** used for a single image sample with increasing severities from left to right.

A.2 Fast adaptation using test-time feedback

A.2.1 Monocular Depth

Training Details

All networks for our method and baselines use the same UNet architecture [94] and were trained with AMSGrad [95], with a learning rate of 5×10^{-4}, weight decay 2×10^{-6} and batch size of 64.

Rapid Network Adaptation (RNA). We aim to fuse the additional information provided by g at test-time into f_θ via a small network, h_ϕ. We insert k Feature-wise Linear Modulation (FiLM) layers into the network f_θ, we denote this network as $f_{\hat{\theta}}$. Each FiLM layer performs the following operation $\text{FiLM}(x_i; \gamma_i, \beta_i) = \gamma_i \odot \mathbf{x}_i + \beta_i$, where $\mathbf{x_i}$ is the activation of layer i. For depth estimation, h_ϕ only predicts $\{\beta_i\}_{i=1}^k$, thus, $\gamma_i = 1, i = 1 \dots k$ as it gave similar results with fewer parameters. h_ϕ consists of an encoder with 3 downsampling blocks and k convolution layers (one for each FiLM layer) to predict the β_i parameters. h_ϕ was trained by randomly masking the GT, with sparsity ratio ranging from 0% to 0.25%. h_ϕ is trained using the following loss function:

$$\min_{\phi} \mathcal{L} = \sum_{n=1}^{N} \| f_\theta(x_n; h_\phi(y_n \odot \bar{m}_n)) - y_n \|_1 \tag{A.5}$$

where $\bar{m}_n$ is the mask that simulates the sparsity pattern that will be encountered at test-time. If g extracts sparse depth via SFM, we use the model from [278] to extract keypoints locations to generate $\bar{m}$. This creates a sparse binary mask that we apply on the GT. We then add Gaussian noise to $y_n \odot \bar{m}_n$ to account for potential outliers. At test-time, RNA only requires a forward pass through h_ϕ and $f_{\hat{\theta}}$.

Test-Time Optimization (TTO) details. We use the same optimization parameters as in training time. Optimization is done for 10 iterations for each batch. With the exception of the TENT baseline, all parameters of the model were updated. For TENT, only the GroupNorm parameters were updates as it results in more stable optimization. Test-Time Optimization minimizes the following loss function:

$$\min_{\hat{\theta}} \mathcal{L} = \sum_{n=1}^{N} \| f_\theta(x_n) \odot m_n - z_n \|_1. \tag{A.6}$$

where N is the number of datapoints, $\odot$ is the element-wise product and $\hat{\theta}$ is the subset of parameters of f_θ that is updated i.e., $\hat{\theta} \subseteq \theta$.

Additional Results

RNA with existing supervision signals. We show that one can train a controller network to use existing adaptation signals, e.g. prediction entropy and Sobel edges to get better performance at test time. As described in Sec. 4.1 in the main paper, to use supervision from entropy, we train a Baseline UNet model with NLL loss. The model outputs the prediction and an uncertainty estimate of the predictions. We also show results with the model from [142] with calibrated uncertainty estimates as it was shown to predict uncertainties that are better correlated with error. The results are shown in Table A.6. For all cases, RNA improves on the performance over the baseline, even if the adaptation signal is poor. This is in contrast to TTO, where the performance can be worse after adaptation, e.g. after optimizing entropy.

Method\Supervision	Sobel edge	Entropy	Entropy (calibrated)	Sparse GT
RNA	**0.29**	**0.07**	**0.12**	**4.06**
TTO (GN)	-0.02	-0.11	0.04	0.55
TTO (F)	0.06	-1.21	0.06	2.76

Table A.6: **RNA can be used with existing supervision signals.** ℓ_1 errors on the depth prediction task. The numbers are relative to the baseline error (i.e. the difference between that method's ℓ_1 error and that of the pre-adaptation baseline's). F denotes TTO by optimizing all parameters and GN denotes TTO by optimizing only group normalization parameters.

Controlling for different number of parameters. Table A.7 shows the results on Common Corruptions applied to Taskonomy test set. All methods have the same architecture, thus, same number of parameters. RNA still outperforms, thus, its performance is not due to extra parameters or architecture.

Shift	Pre-adaptation Baseline	Densification	TTO	RNA
CC	0.045	0.023	0.019	**0.018**

Table A.7: **Controlling for number of parameters.** ℓ_1 errors (multiplied by 100 for readability) on the depth estimation task, evaluated on the Taskonomy test set under a subset of common corruptions. Each method is using the same architecture and number of parameters. The adaptation signal here is masked GT, fixed at 0.05% of valid pixels.

Implementations of RNA with different architectures. We experimented with several versions of RNA. Instead of adding FiLM layers to adapt f_θ (denoted as FiLM-f), we also added FiLM layers to a UNet model that is trained to update the input image x (denoted as FiLM-x). For FiLM-x, only x is updated and there is no adaptation on f_θ. Lastly, as Hypernetworks [145] have been shown to be expressive and suitable for adaptation, we trained a HyperNetwork, in this case an MLP, to predict the weights of a 3-layer convolutional network that updates x (denoted as HyperNetwork-x). In all cases, RNA takes in the sparse error, i.e., the difference between the predictions and the

adaption signal, as opposed to separate inputs as done in the main paper. The results of adaptation with these variants of RNA are shown in Table A.8. The FiLM-f variant performed best, thus, we adopted it as our main architecture.

Method\Shift	None	Taskonomy-CC	Taskonomy-3DCC	Hypersim	BlendedMVG
Pre-adaptation	0.027	0.057	0.048	0.336	3.450
RNA (HyperNetwork-x)	0.019	0.041	0.033	0.257	2.587
RNA (FiLM-x)	0.019	0.039	0.033	0.279	2.636
RNA (FiLM-f)	0.013	0.024	0.020	0.198	2.310

Table A.8: **RNA with different architectures.** ℓ_1 errors on the depth estimation task under distribution shifts are reported. The adaptation signal here is masked GT, fixed at 0.05% of valid pixels.

RNA performs better than training a single model that takes as input the RGB image and sparse supervision concatenated. We call this model Multi-domain. For 0.05% GT supervision, RNA has a much better performance, with an average loss over all distortions and severities of 0.0179 while the multi-domain model has an average loss of 0.0255 (see Fig. A.16).

Results for different levels of GT supervision. In Fig. A.16, we show how the error changes with increasing GT supervision for our proposed methods and the baselines. Note that for the two RNA variants (frozen f and jointly trained f), and multi-domain model, we also included the case where only the supervision signal z in passed as input to h_ϕ. These methods have been post-fixed with 'z input only' in the legend.

Model-based RNA. During training of RNA, we applied several augmentations on both x (speckle noise, Gaussian noise, spatter) and y (affine transformation). The augmentations on x are those from the validation set of Common Corruptions [2]. Training with validation corruptions has also been done in [41]. The motivation for y augmentation is that there is a scale ambiguity for monocular depth estimation. Thus, we can change the scale of the ground truth depth label, while keeping x the same, this trains RNA to predict depth values in different ranges. We call the parameters of these augmentations e.g., the kernel size for Gaussian blur, environmental parameters.

We trained RNA to predict environmental parameters, and to use these parameters for adaptation. Thus, compared to the variants of RNA mentioned before, we are now forcing RNA to learn a model of the environment. Training is done in several stages. First, we trained h^1_ϕ to take in environmental parameters and update x. This is akin to a controller. Next, we train h^2_ϕ to predict these environment parameters from the adaptation signal and predictions, akin to a sensor. Finally, we finetune both h^1_ϕ and h^2_ϕ. Thus, $h_\phi = h^2_\phi \circ h^1_\phi$. This model is denoted as (model-based). We also trained a version of this with a single training stage, i.e., h_ϕ is not forced to learn to predict the environment parameters (denoted as model-free). The results are shown in Tab. A.9. Although the performance of both version of RNA are similar, we believe this is an interesting future direction.

Method\Shift	None	Taskonomy-CC	Taskonomy-3DCC	Hypersim	BlendedMVG
Pre-adaptation	0.025	0.053	0.045	0.336	3.450
RNA (model-based)	0.020	0.042	0.035	0.212	2.281
RNA (model-free)	0.020	0.041	0.033	0.215	2.326

Table A.9: **Comparison of model-based and model-free versions of RNA.** ℓ_1 errors on the depth estimation task, under distribution shifts. The adaptation signal here is masked GT, fixed at 0.05% of valid pixels.

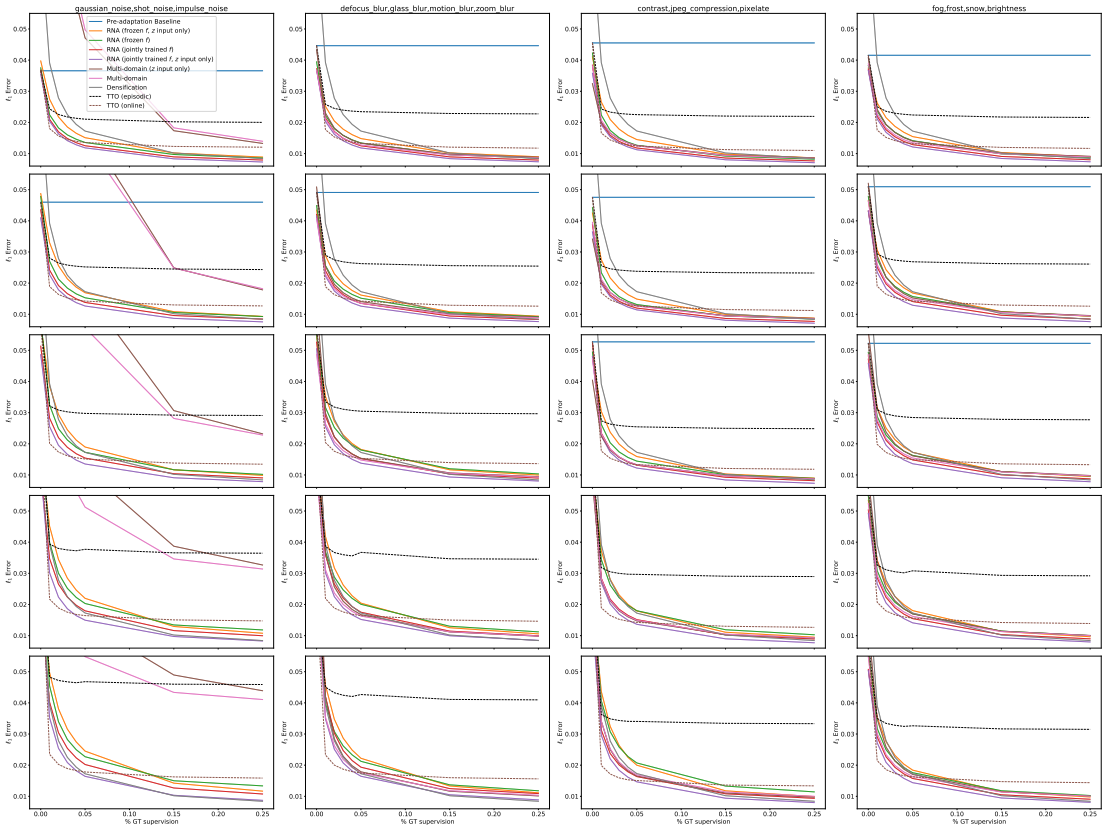

Figure A.16: ℓ_1 **error vs different levels of GT supervision.** These results are on Taskonomy-CC data, as described in Sec. 4.3 in the main paper. Each curve shows an average over all the distortions listed in the title.

Qualitative results. We provide more qualitative results in Fig. A.23 where our method outperforms the baselines.

A.2.2 Optical Flow Experiments

To predict optical flow, we use a pre-trained RAFT model from [215]. We use sparse optical flow as our supervision signal, attained from keypoint matching between images. We perform TTO to adapt the model. We use the same episodes from Replica+CC as described in Sec. 4.3 in the main paper. TTO was done for 10 iterations for each episode, all parameters of the RAFT model were updated. Figure A.17 shows the results.

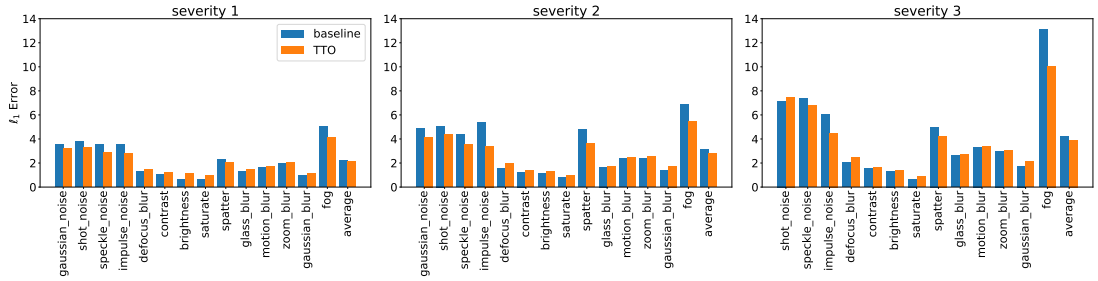

Figure A.17: **Quantiative results for optical flow.** These results were attained from Replica+CC data, as described in Sec 4.3 in the main paper.

Adaptation with TTO results in an 8.5% improvement over the baseline.

A.2.3 Dense 3D Reconstruction

Test-Time Optimization details. As mentioned in the main paper, we achieve multi-view consistency using the same process as [179], where: **1.** every pixel is backprojected into 3D world-coordinates using the estimated depth and camera poses, **2.** optical-flow predictions are employed to establish dense correspondences across pixels, **3.** weights of the depth model are optimized to minimize the discrepancy between the estimated 3D world coordinates of corresponding pixels.

We provide additional qualitative results and the corresponding error images in Fig. A.18.

A.2.4 Semantic Segmentation

Training Details

TTO. We optimize by SGD with 0.0001 learning rate, 0.9 momentum, batch size 2 for 10 iterations per batch.

TENT. Following [53], we optimize by SGD with 0.0001 learning rate, 0.9 momentum, batch size 1. As TENT is unstable for online and multi-iteration optimization, we restart the model after each batch, i.e. episodic optimization, and run 1 iteration per batch.

TENT (all). In contrast to [53] which only updates batch normalization parameters at test-time, we also included a TENT baseline that optimizes all parameters. We optimize by SGD with 0.00001 learning rate, 0.9 momentum, batch size 1. We run the optimization for each image for 10 iterations to be comparable to the TTO model. Note that we reduced learning rate by a factor of 10 as TENT was unstable. Since TENT and TENT (all) models perform similarly, we only show the TENT results in the main paper. See A.2.4 for all results.

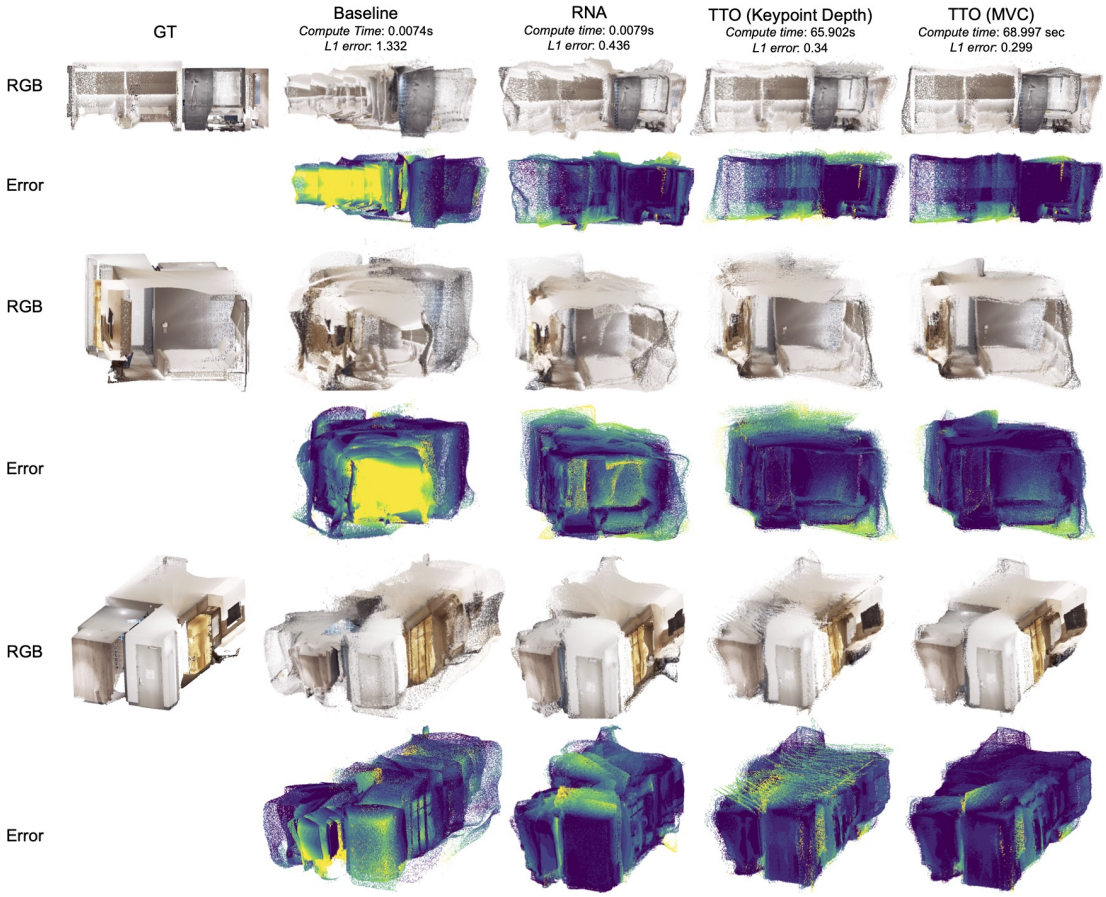

Figure A.18: Extension of Fig. 4 from main paper to show more examples and error maps.

RNA. As an encoder we used a small CNN with 3 downsampling blocks. We trained the FiLM generator with frozen segmentation model on the *clean* COCO training dataset with cross entropy losses. During the training we sparsify the target segmentation mask uniformly in [0,30] pixels to generate sparse ground truth inputs to the encoder. We optimize by Adam with 0.0001 learning rate and 0.0001 weight decay. We select the model with highest mean IOU on the clean validation set. The model is trained with two forward passes. During the first pass the input to FiLM encoder is sparse ground truth and zeros as prediction. After this, in the second pass the encoder takes the sparse ground truth and the prediction from the previous pass as input. We compute the cross-entropy loss for the prediction of second pass.

Densification. It uses the same FCN-ResNet50 model as other baselines and is trained with the same setup as RNA.

Additional Results

Quantitative Results. We provide the results for mean IOU vs number of pixels for different severities in Fig. A.24. Figures A.25, A.26, A.27, A.28, A.29, A.30, A.31 give a breakdown of the performance for our methods and baselines for each corruption, severity level, and number of pixel annotations.

Qualitative Results. We include additional qualitative comparisons between our methods and baselines in Figures A.32, A.33, A.34.

A.2.5 Image classification

Generating Coarse Label Sets

One method to generate coarse label sets using WordNet tree [208] is proposed in [194]. This method and other clustering methods create imbalanced coarse labels and too many ImageNet classes are assigned to coarse labels that are either too coarse or too fine-grained (See Figure A.19 and Table A.14 for the statistics). As we aim to use coarse supervision to adapt models at test-time, we focus on generating more balanced coarse labels, as explained below.

To generate balanced coarse label sets, instead of going from top to bottom or bottom to top for a fixed depth in WordNet tree, we follow a different approach. For each ImageNet class we go up until we get to a hypernym that has a certain number of hyponyms that are ImageNet classes. That number determines the coarsity level of the coarse label. To achieve this, we use a priority-based selection criteria where we define certain ranges to reach a given coarsity level. Using this approach, we created three different coarse sets with 26, 45, and 85 coarse labels. See Tables A.10, A.11, A.12, A.13 for the coarse labels and their IDs. The resulting sets are more balanced than the 127-label set provided in [194]. See Figures A.19, A.20, A.21, A.22 and Table A.14 for more details about the statistics of the coarse label sets.

Training Details

The baseline model used for classification is ResNet50 trained on ImageNet.

TTO. We optimize by SGD with 0.00025 learning rate, 0.9 momentum and batch size 64, following [53]. The following loss function is used for TTO, which is a linear combination of cross-entropy loss on the summation of probabilities of all of the classes in the coarse label set, and the entropy of the predictions:

$$\min_{\phi} \mathcal{L} = -\log \sum_{c\in} p_c + w_e \sum_{c} -p_c \log p_c \tag{A.7}$$

where p_c is the probability of class c and is the set of classes that are present in the coarse label.

TENT. We used the optimizer and parameters that were reported in [53] to adapt TENT. For both TTO and TENT, we optimize the transformation parameters of the normalization layers and estimate the normalization statistics from the current batch. Note that for each batch we re-evaluate after the updates (in our experiments we run 1 iteration per batch) to get the final predictions.

RNA. We optimize by Adam with 0.0001 learning rate, 0.0001 weight decay, batch size 64 and for about 50 epochs. The FiLM generator we used has an encoder-decoder structure. The encoder has one hidden layer with 128 nodes and 64 nodes for the output layer. The decoder for each FiLM layer has one hidden layer with 64 nodes and the output size is equal to the number of FiLM layer parameters. The FiLM layers are inserted between normalization layers and ReLUs. During training all of the model parameters are fixed and only the FiLM generator parameters are being trained, and cross-entropy loss is minimized.

Additional Results

In Figures A.35, A.36, A.37 we provide a breakdown of performance against individual corruptions from ImageNet-C and ImageNet-3DCC using 26-,45-, and 85-way coarse labels. We also included the results when we used 127-way coarse labels from [194] in Fig. A.38. Note that this coarse set has imbalances as explained in A.2.5, yet our methods can still benefit from it.

n01428580 soft-finned fish	n02858304 boat	n03613592 key	n06595351 magazine
n01482330 shark	n02924116 bus	n03619396 kit	n06793231 sign
n01495701 ray	n02942699 camera	n03664943 ligament	n06874019 light
n01503061 bird	n02954340 cap	n03666917 lighter-than-air craft	n07557434 dish
n01629276 salamander	n03035510 cistern	n03678362 litter	n07560652 fare
n01639765 frog	n03039947 cleaning implement	n03764276 military vehicle	n07579575 entree
n01662784 turtle	n03094503 container	n03825080 nightwear	n07582609 dip
n01674990 gecko	n03101156 cooker	n03837422 oar	n07611358 frozen dessert
n01676755 iguanid	n03122748 covering	n03880531 pan	n07612996 pudding
n01685439 teiid lizard	n03151500 cushion	n03896233 passenger train	n07680932 bun
n01687665 agamid	n03183080 device	n03906997 pen	n07681926 cracker
n01689411 anguid lizard	n03236735 dress	n03961939 platform	n07683786 loaf of bread
n01691951 venomous lizard	n03241093 drill rig	n03964744 plaything	n07707451 vegetable
n01692864 lacertid lizard	n03257586 duplicator	n03990474 pot	n07800740 fodder
n01693783 chameleon	n03278248 electronic equipment	n04015204 protective garment	n07829412 sauce
n01694709 monitor	n03294833 eraser	n04077734 rescue equipment	n07882497 concoction
n01697178 crocodile	n03309808 fabric	n04099429 rocket	n07891726 wine
n01698434 alligator	n03405725 furniture	n04100174 rod	n07929519 coffee
n01703569 ceratopsian	n03414162 game equipment	n04125853 safety belt	n07930554 punch
n01726692 snake	n03419014 garment	n04128837 sailing vessel	n09214060 bar
n01767661 arthropod	n03441112 glove	n04185071 sharpener	n09287968 geological formation
n01861778 mammal	n03446832 golf equipment	n04194289 ship	n09289709 globule
n01909422 coelenterate	n03450516 gown	n04235291 sled	n09820263 athlete
n01922303 worm	n03472232 gymnastic apparatus	n04264914 spacecraft	n10019552 diver
n01940736 mollusk	n03476083 hairpiece	n04285622 sports implement	n10401829 participant
n02316707 echinoderm	n03497657 hat	n04317420 stick	n11669921 flower
n02512938 food fish	n03510583 heavier-than-air craft	n04341686 structure	n12992868 fungus
n02605316 butterfly fish	n03513137 helmet	n04377057 system	n13134947 fruit
n02606384 damselfish	n03528263 home appliance	n04447443 toiletry	
n02638596 ganoid	n03540267 hosiery	n04451818 tool	n15074962 tissue
n02642644 scorpaenid	n03597469 jewelry	n04500060 turner	
n02652668 plectognath		n04509592 uniform	
n02807260 bath linen		n04571292 weight	
n02856463 board			

n00002137 abstraction	n02087122 hunting dog	n03125870 craft	n04014297 protective covering
n00004475 organism	n02087551 hound	n03183080 device	n04081844 restraint
n00007347 causal agent	n02092468 terrier	n03257877 durables	n04170037 self-propelled vehicle
n00020090 substance	n02098550 sporting dog	n03278248 electronic equipment	n04285146 sports equipment
n00020827 matter	n02103406 working dog	n03294048 equipment	n04341686 structure
n00021265 food	n02104523 shepherd dog	n03297735 establishment	n04447443 toiletry
n00021939 artifact	n02120997 feline	n03309808 fabric	n04451818 tool
n01503061 bird	n02159955 insect	n03405265 furnishing	n04524313 vehicle
n01525720 oscine	n02329401 rodent	n03405725 furniture	n04530566 vessel
n01627424 amphibian	n02370806 ungulate	n03414162 game equipment	n04531098 vessel
n01661091 reptile	n02394477 even-toed ungulate	n03419014 garment	n04576211 wheeled vehicle
n01661818 diapsid	n02469914 primate	n03528263 home appliance	n04586932 wind instrument
n01674464 lizard	n02484322 monkey	n03563967 implement	n07570720 nutriment
n01726692 snake	n02512053 fish	n03574816 instrument	n07705931 edible fruit
n01767661 arthropod	n02528163 teleost fish	n03575240 instrumentality	n07707451 vegetable
n01844917 aquatic bird	n02778669 ball	n03699975 machine	n09287968 geological formation
n01861778 mammal	n02913152 building	n03733925 measuring instrument	n12992868 fungus
n01886756 placental	n02958343 car	n03738472 mechanism	n13134947 fruit
n01905661 invertebrate	n03051540 clothing	n03791235 motor vehicle	
n02000954 wading bird	n03076708 commodity	n03800933 musical instrument	
n02075296 carnivore	n03094503 container	n03839993 obstruction	
n02083346 canine	n03100490 conveyance		
n02084071 dog	n03122748 covering		

Table A.11: **The classes used in the 85-coarse label set.** See Figure A.20 for more details.

n00002137 abstraction	n02075296 carnivore	n03093574 consumer goods	n03738472 mechanism
n00004475 organism	n02083346 canine	n03094503 container	n03800933 musical instrument
n00007347 causal agent	n02084071 dog	n03100490 conveyance	n04014297 protective covering
n00019128 natural object	n02087551 hound	n03122748 covering	n04081844 restraint
n00020827 matter	n02092468 terrier	n03183080 device	n04341686 structure
n00021939 artifact	n02098550 sporting dog	n03294048 equipment	n04524313 vehicle
n01503061 bird	n02103406 working dog	n03309808 fabric	n04531098 vessel
n01627424 amphibian	n02120997 feline	n03405265 furnishing	n09287968 geological formation
n01661091 reptile	n02370806 ungulate	n03563967 implement	n12992868 fungus
n01861778 mammal	n02441326 musteline mammal	n03574816 instrument	
n01886756 placental	n02469914 primate	n03575240 instrumentality	
n01905661 invertebrate	n02512053 fish	n03699975 machine	

Table A.12: **The classes used in the 45-coarse label set.** See Figure A.21 for more details.

n00002137 abstraction	n01661091 reptile	n03257877 durables	n04524313 vehicle
n00004475 organism	n01861778 mammal	n03294048 equipment	n04531098 vessel
n00007347 causal agent	n01905661 invertebrate	n03309808 fabric	n09287968 geological formation
n00020827 matter	n02075296 carnivore	n03405265 furnishing	n12992868 fungus
n00021939 artifact	n02512053 fish	n03563967 implement	n13134947 fruit
n01503061 bird	n03122748 covering	n03575240 instrumentality	
n01627424 amphibian	n03183080 device	n04341686 structure	

Table A.13: **The classes used in the 26-coarse label set.** See Figure A.22 for more details.

Coarse Label Set	Min	Max	Mean	Median	Mode	Standard Deviation
127-way	1	218	88.40	59.0	218	79.96
85-way	6	522	44.05	24.0	26	59.10
45-way	7	522	59.33	50.0	67	60.27
26-way	7	522	105.86	67.0	158	84.77

Table A.14: **The statistics of coarse label sets.** See Figures A.19, A.20, A.21, A.22 for more details.

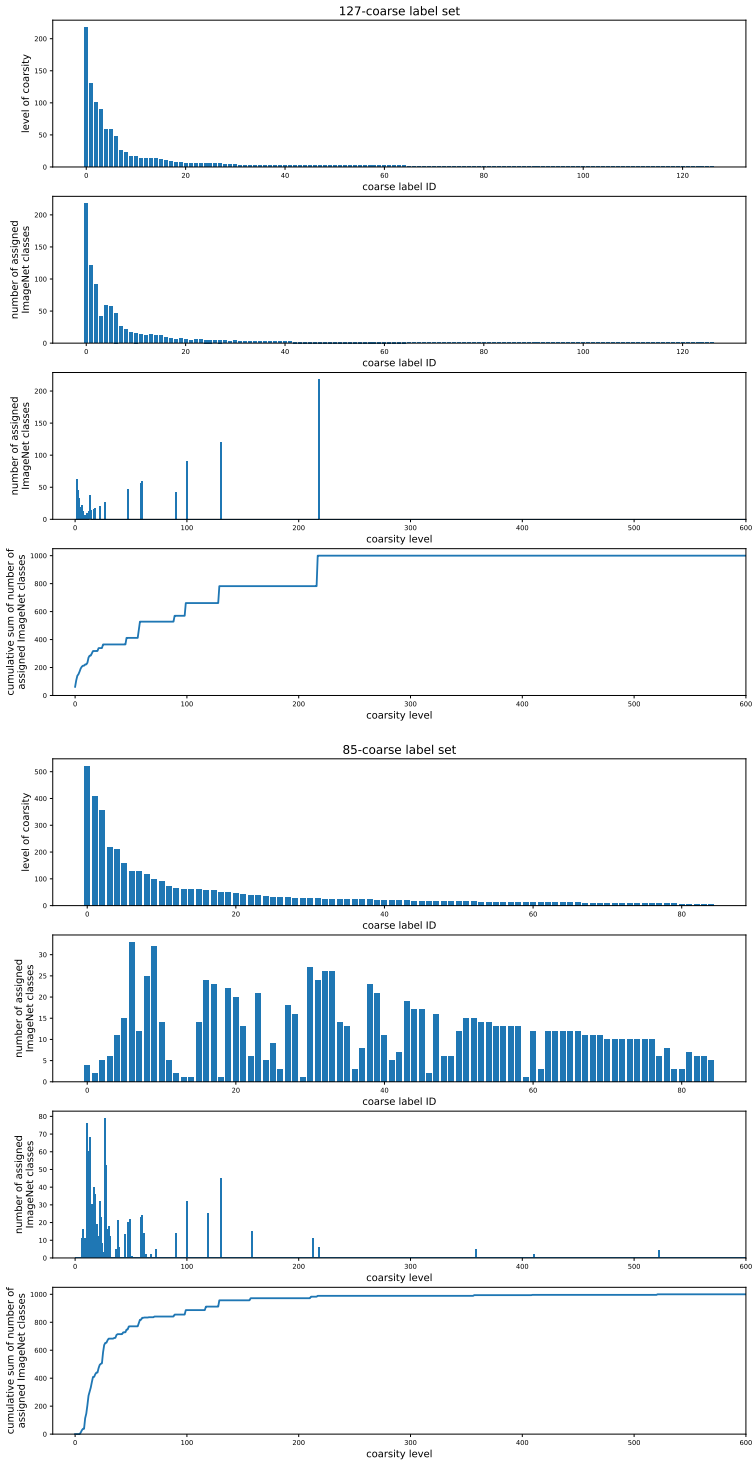

Figure A.20: **Distribution of 85-coarse label set.** None of ImageNet classes are using coarse labels with coarsity level of 5 or less, 114 classes are using coarse labels with coarsity level of 10 or less, and 28 classes are using the coarse labels with coarsity level of 200 or more.

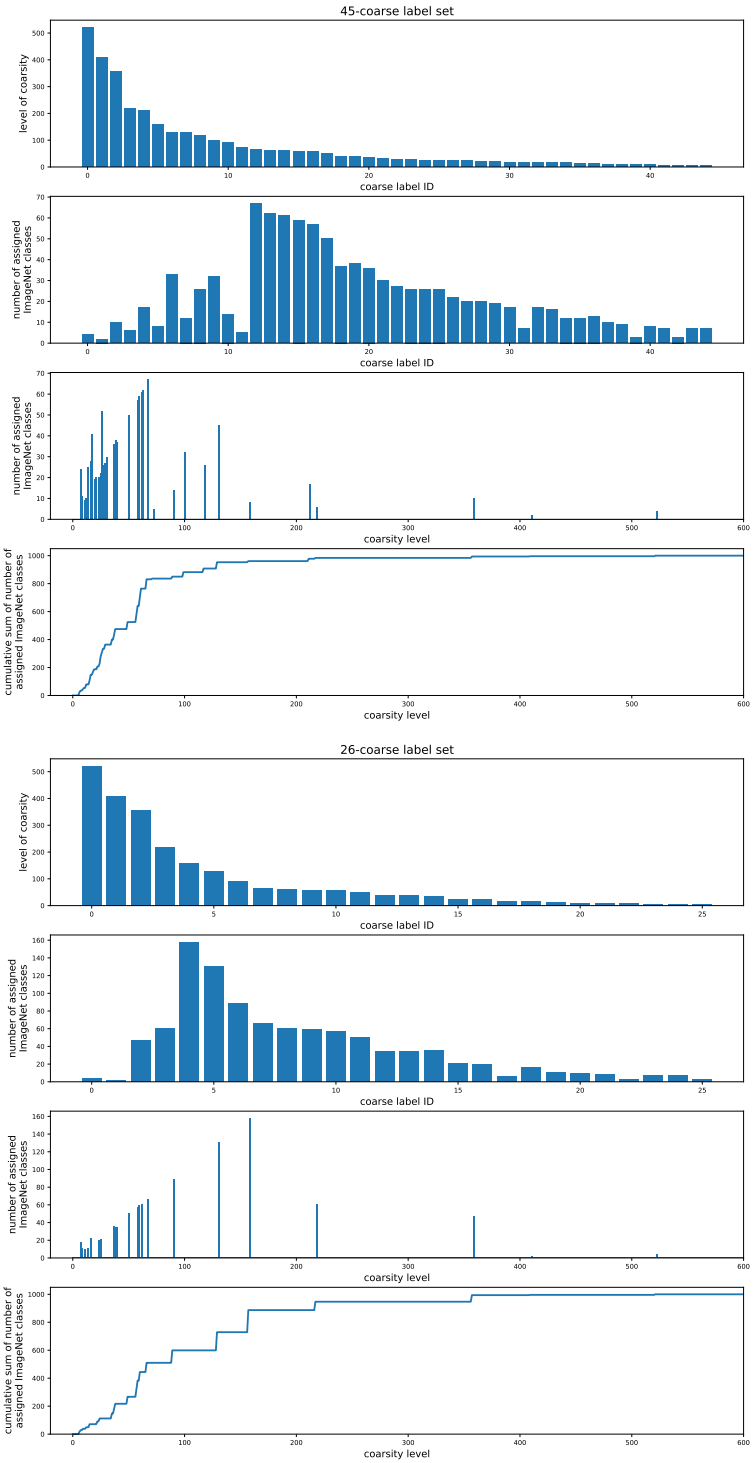

Figure A.22: **Distribution of 26-coarse label set.** None of ImageNet classes are using coarse labels with coarsity level of 5 or less, 38 classes are using coarse labels with coarsity level of 10 or less, and 113 classes are using the coarse labels with coarsity level of 200 or more.

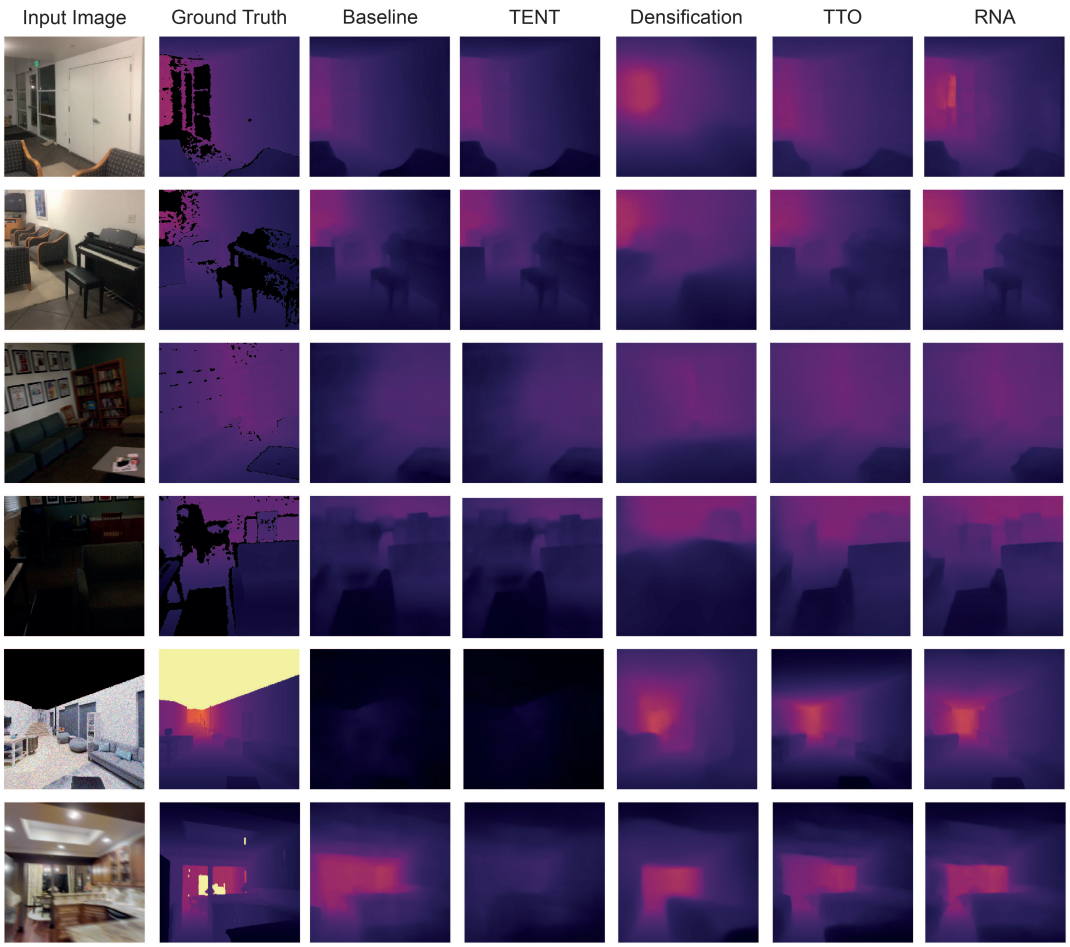

Figure A.23: **Supplementary results for monocular depth estimation.** Qualitative comparison of our method vs baselines on query images from ScanNet, Replica, and Taskonomy datasets.

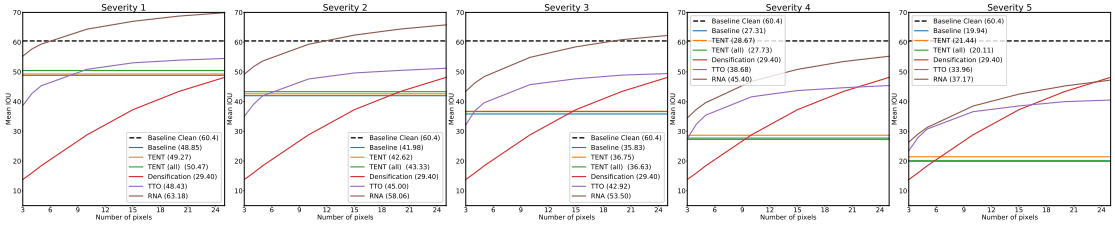

Figure A.24: **Supplementary results for semantic segmentation.** Mean IOU vs number of pixels for different severities. Numbers in the legend denote the average over all pixel levels.

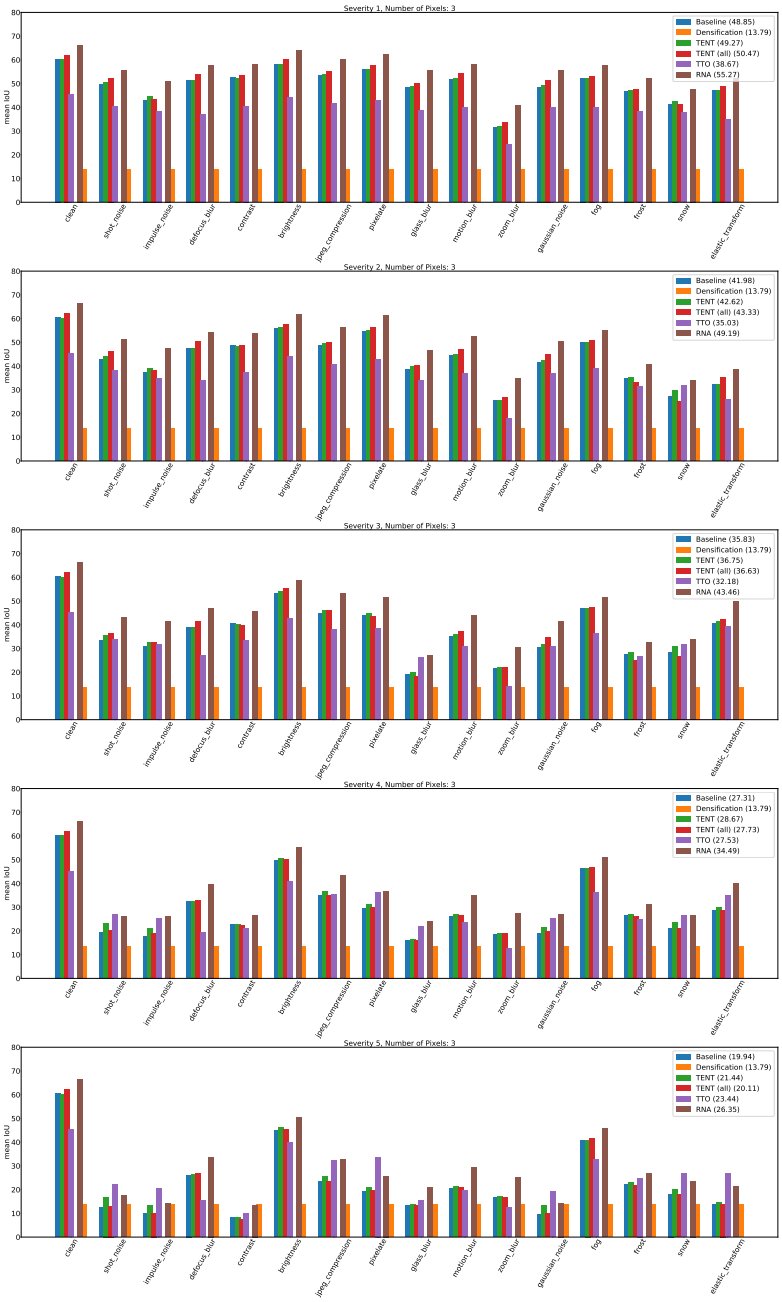

Figure A.25: **Supplementary results for semantic segmentation.** Mean IOU vs individual corruptions for different severities when the number of pixels is 3. Numbers in the legend denote the average over the corruptions.

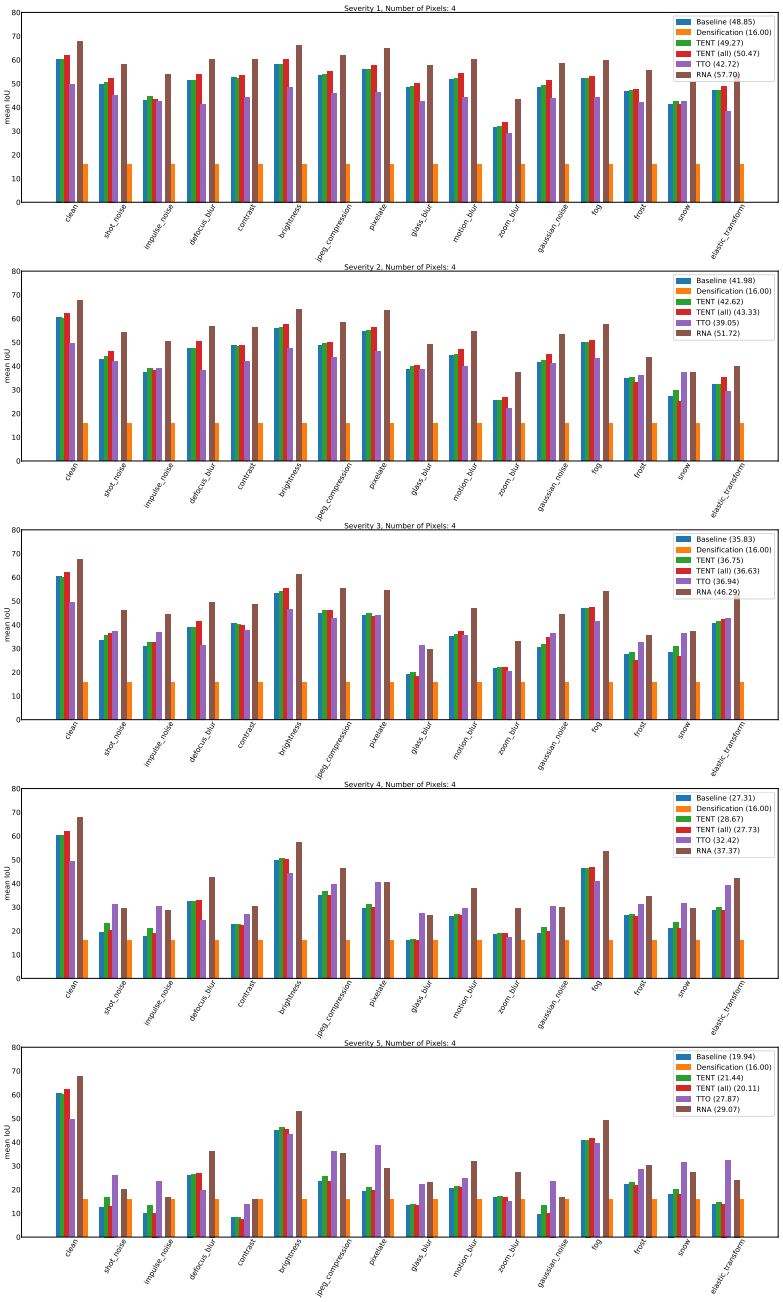

Figure A.26: **Supplementary results for semantic segmentation.** Mean IOU vs individual corruptions for different severities when the number of pixels is 4. Numbers in the legend denote the average over the corruptions.

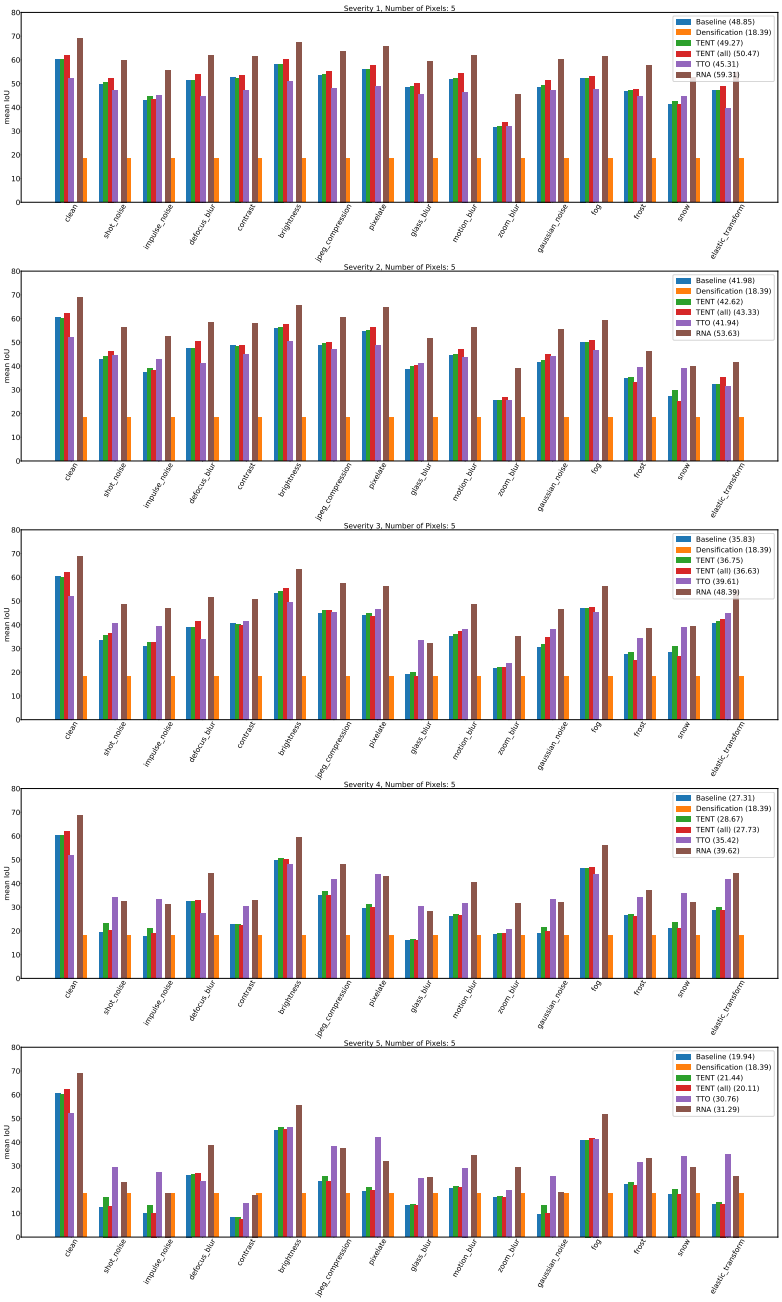

Figure A.27: **Supplementary results for semantic segmentation.** Mean IOU vs individual corruptions for different severities when the number of pixels is 5. Numbers in the legend denote the average over the corruptions.

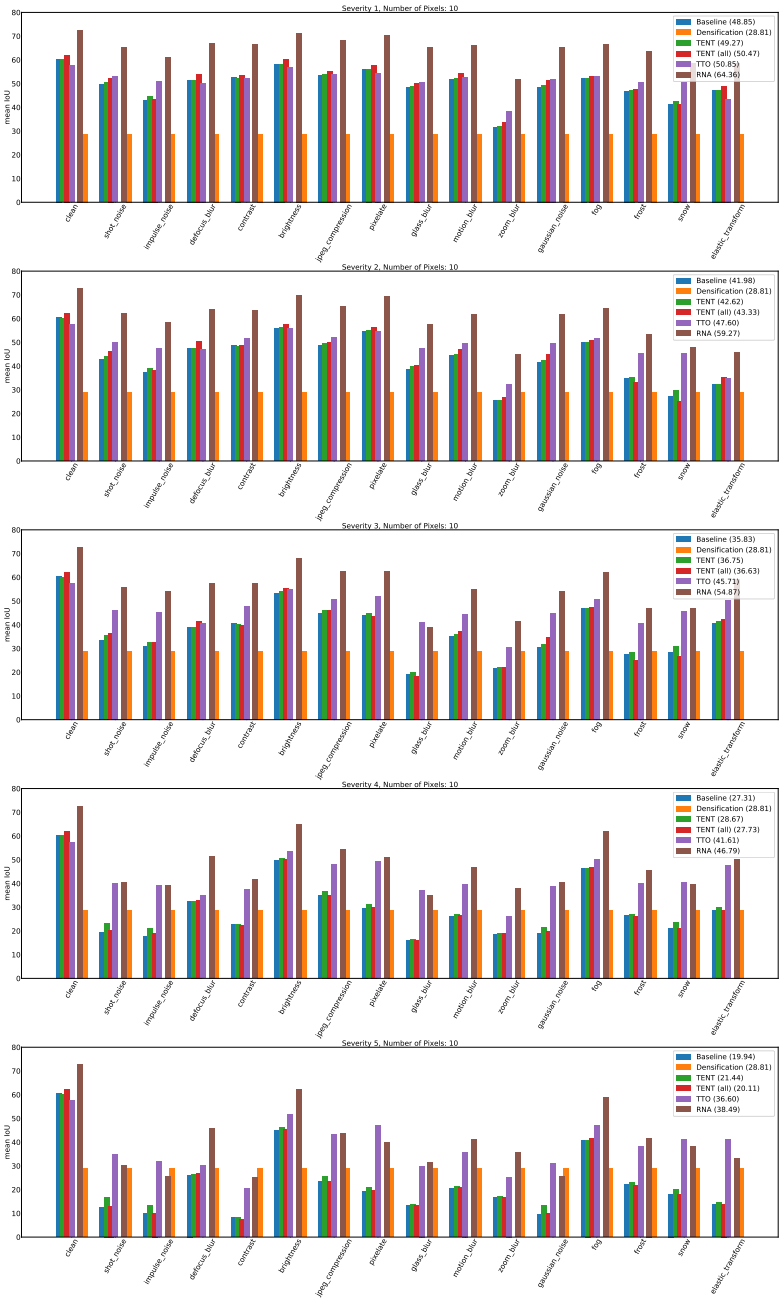

Figure A.28: **Supplementary results for semantic segmentation.** Mean IOU vs individual corruptions for different severities when the number of pixels is 10. Numbers in the legend denote the average over the corruptions.

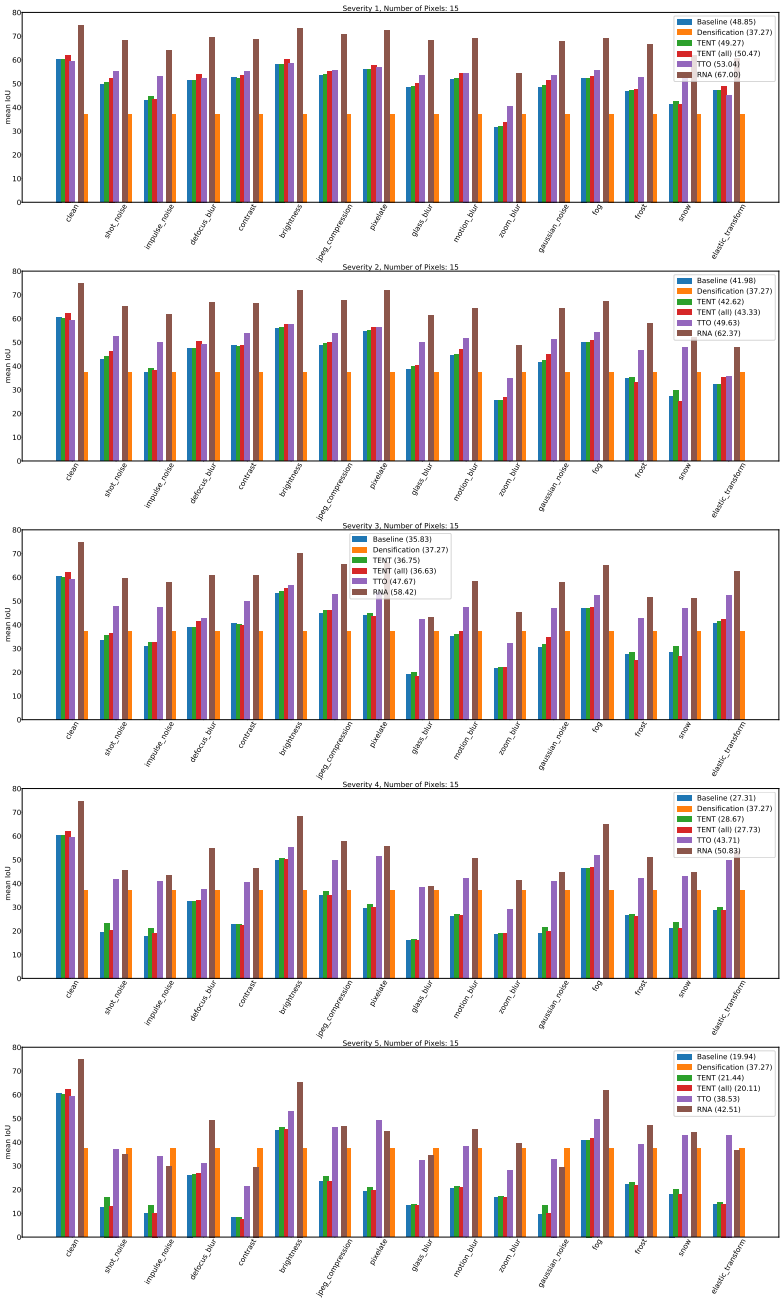

Figure A.29: **Supplementary results for semantic segmentation.** Mean IOU vs individual corruptions for different severities when the number of pixels is 15. Numbers in the legend denote the average over the corruptions.

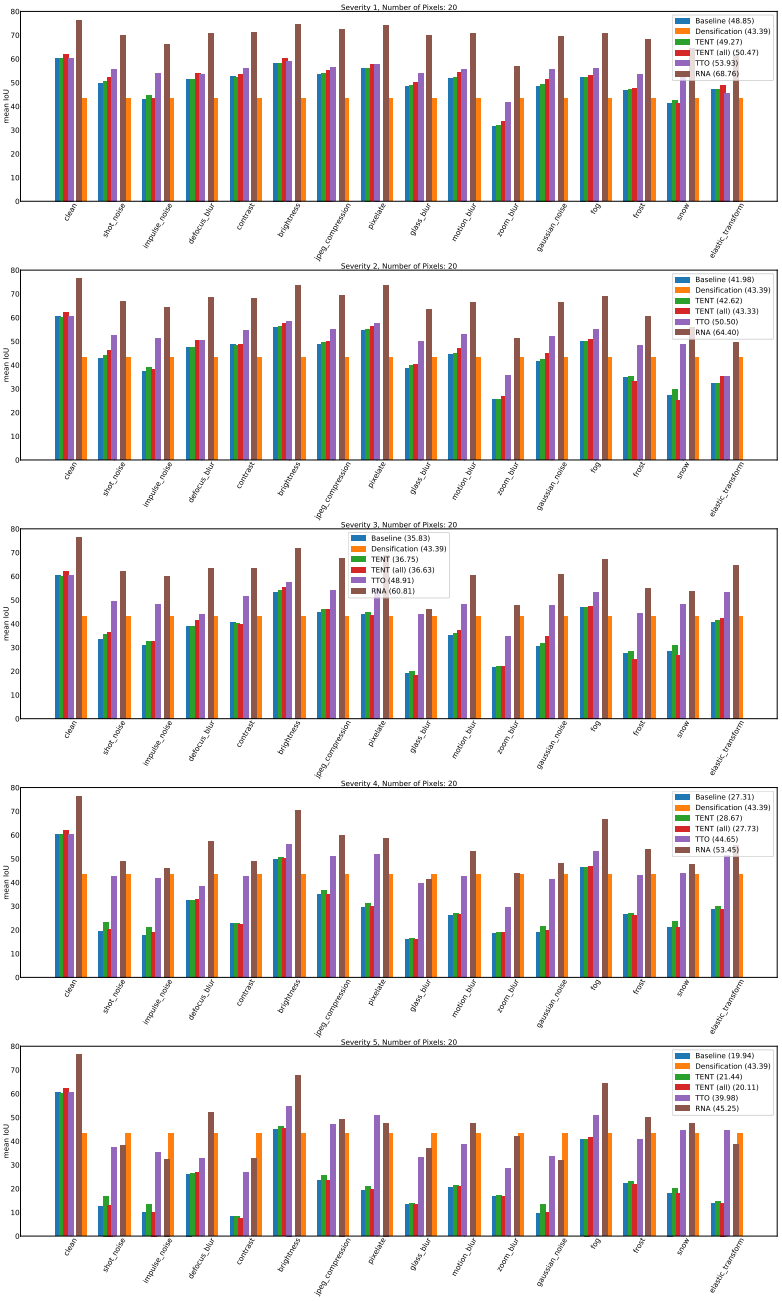

Figure A.30: **Supplementary results for semantic segmentation.** Mean IOU vs individual corruptions for different severities when the number of pixels is 20. Numbers in the legend denote the average over the corruptions.

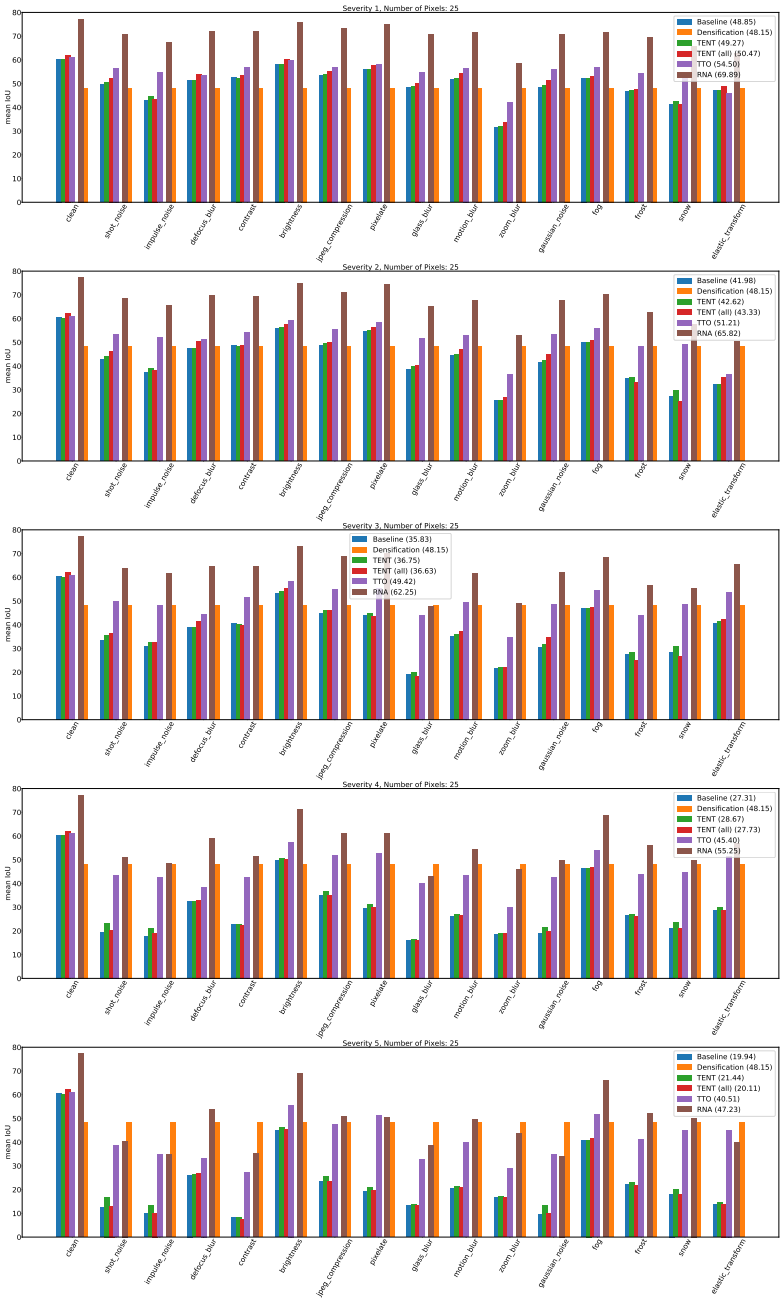

Figure A.31: **Supplementary results for semantic segmentation.** Mean IOU vs individual corruptions for different severities when the number of pixels is 25. Numbers in the legend denote the average over the corruptions.

Figure A.32: **Supplementary results for semantic segmentation.** Qualitative comparison of our method vs baselines for defocus blur and glass blur corruptions applied to COCO validation images.

Figure A.33: **Supplementary results for semantic segmentation.** Qualitative comparison of our method vs baselines for JPEG compression and motion blur corruptions applied to COCO validation images.

Figure A.34: **Supplementary results for semantic segmentation.** Qualitative comparison of our method vs baselines for shot noise corruption applied to COCO validation images.

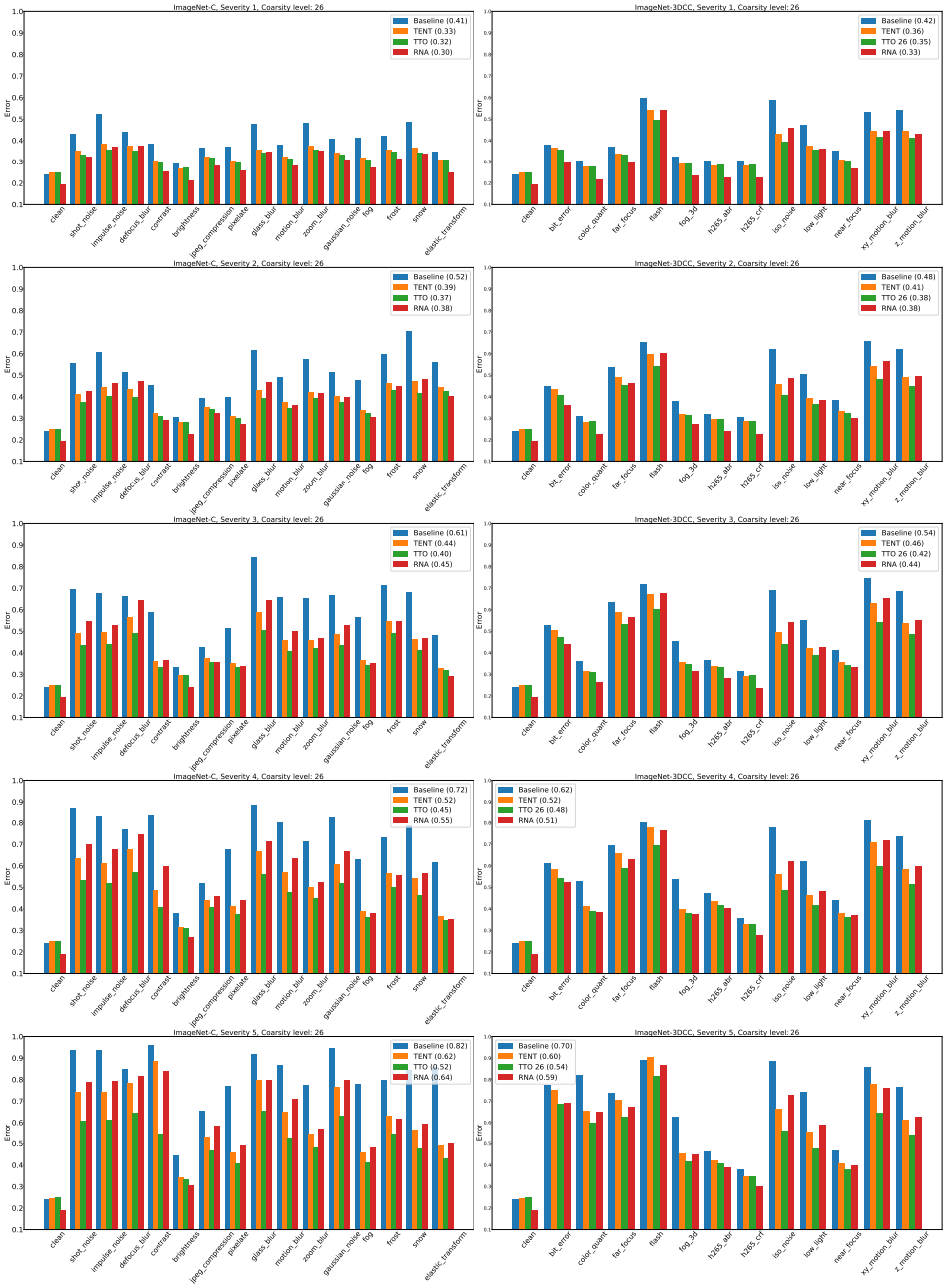

Figure A.35: **Supplementary results for ImageNet classification.** Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 26-way coarse label supervision. Numbers in the legend denote the average over the corruptions.

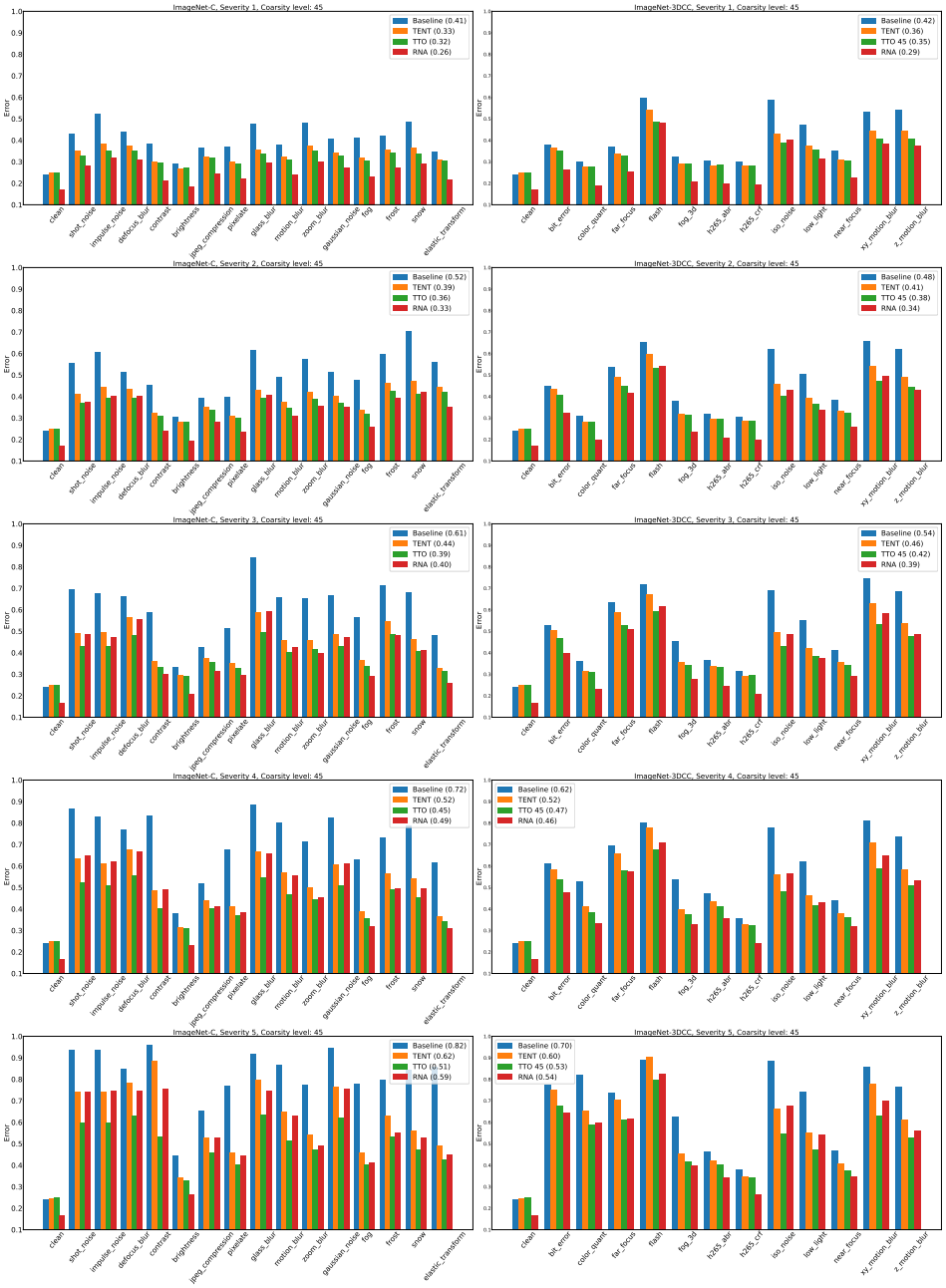

Figure A.36: **Supplementary results for ImageNet classification.** Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 45-way coarse label supervision. Numbers in the legend denote the average over the corruptions.

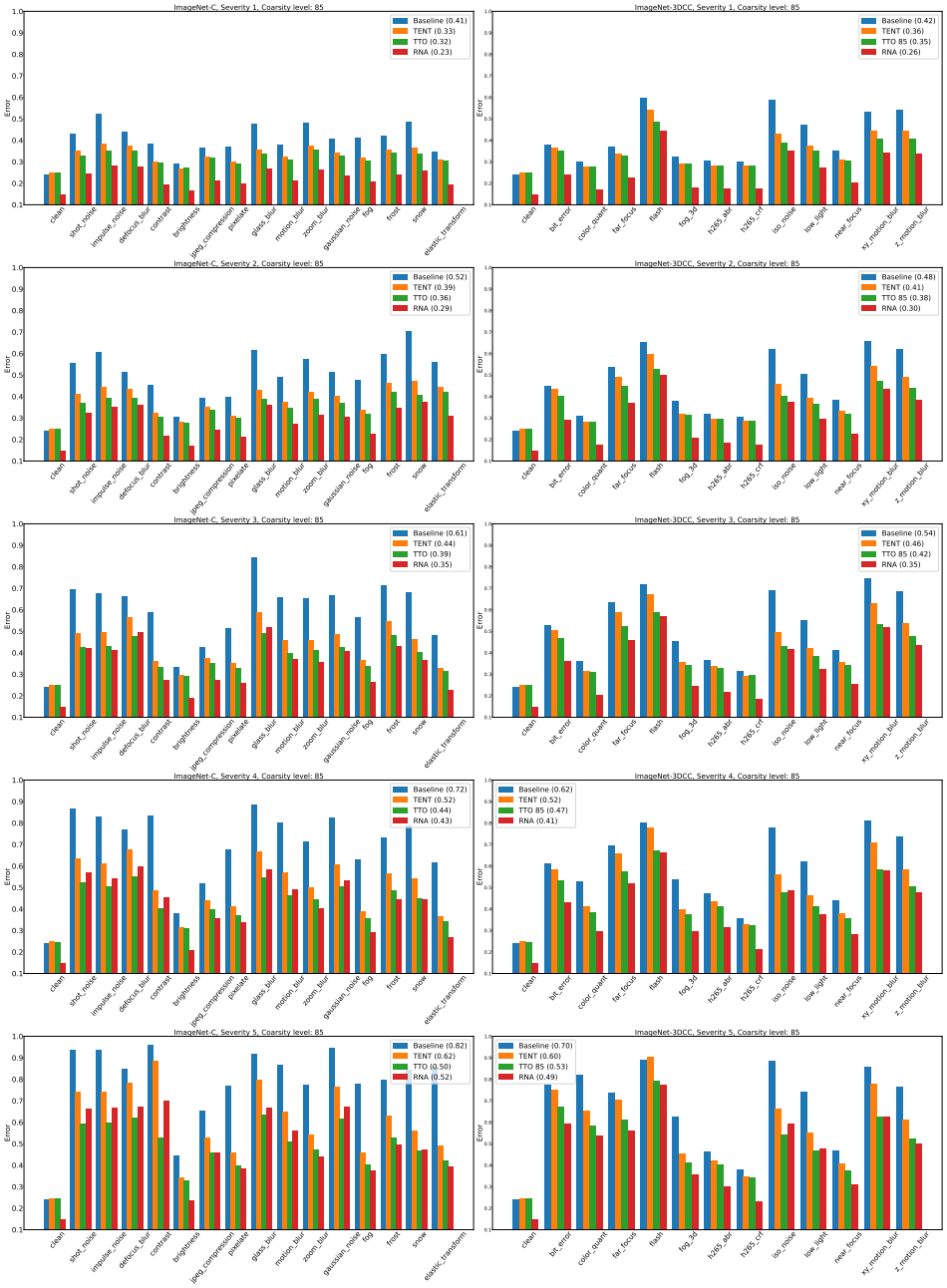

Figure A.37: **Supplementary results for ImageNet classification.** Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 85-way coarse label supervision. Numbers in the legend denote the average over the corruptions.

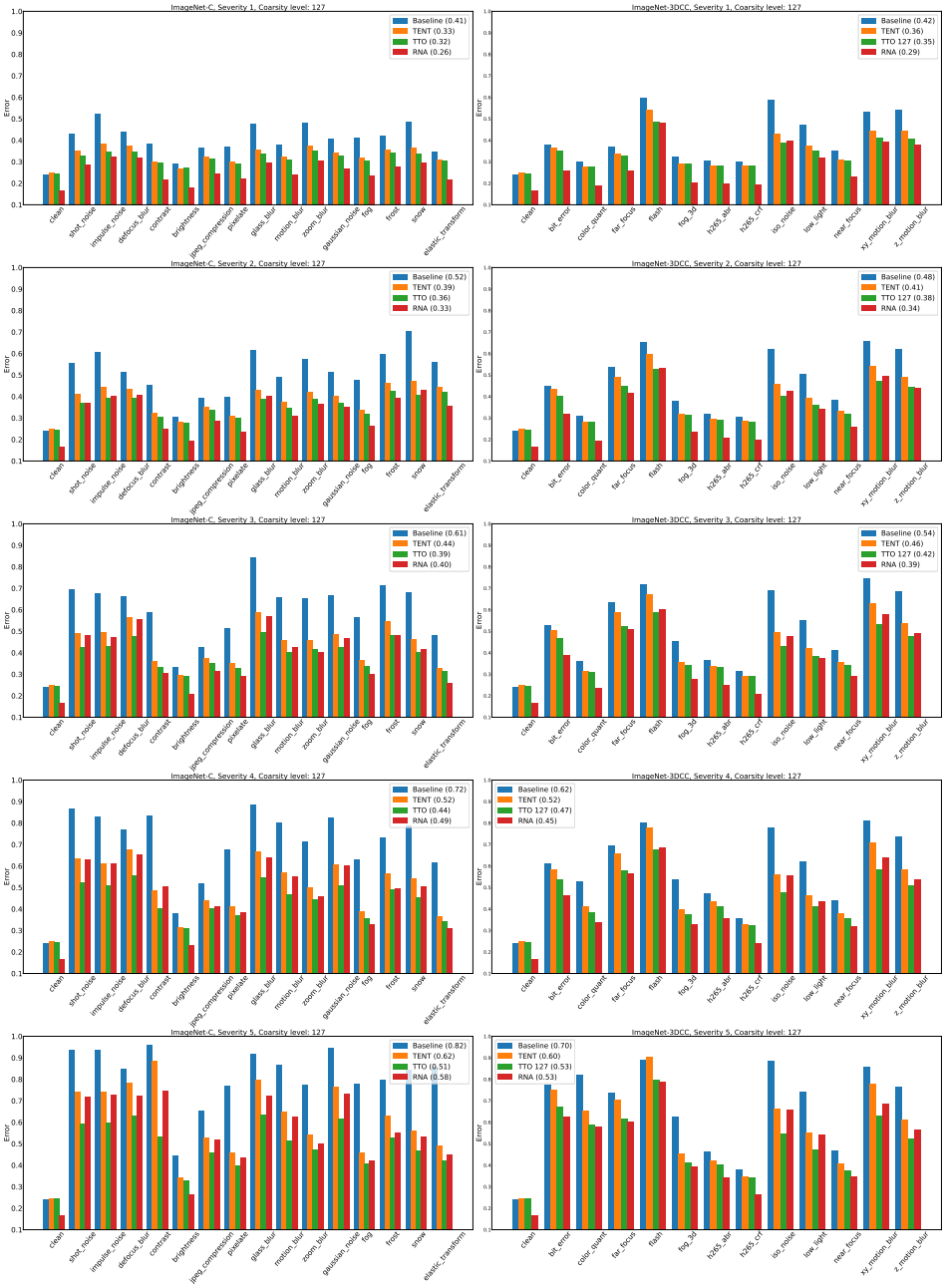

Figure A.38: **Supplementary results for ImageNet classification.** Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 127-way coarse label supervision from [194]. Numbers in the legend denote the average over the corruptions.

A.3 3D Common Corruptions

A.3.1 Quantitative Results

Robustness mechanisms against 3DCC and 2DCC

Figure A.43 shows ℓ_1 errors of robustness mechanisms against individual corruptions in 3DCC for surface normals estimation. Figure A.44 shows the same result for depth estimation. For both tasks, 3DCC leads to significantly degraded predictions for models trained with robustness mechanisms. For completeness, we also provide performances of these models against corruptions in 2DCC, in Figures A.45 and A.46.

Redundancy between 3DCC and 2DCC

We provide in Figures A.47 and A.48 the full affinity matrices between 2DCC and 3DCC by computing the correlations of ℓ_1 errors made in the surface normals and depth estimation tasks, respectively. Figure A.49 shows the same result by computing ℓ_1 errors in the RGB domain. As can be seen, 3DCC yields lower correlations both intra-benchmark and against 2DCC.

Effectiveness of predicted depth to generate 3DCC

We compare the effectiveness of using MiDaS [268] depth estimation model to generate 3DCC against two control baselines in Fig A.42. The first one is *incorrect instance depth* where we randomly swap depth predictions for a given RGB image with another depth prediction. The second one is *blind guess depth* which minimizes the expected likelihood loss in the training dataset, hence it is a statistically informed guess reflecting the dataset regularities [142, 72] (See Fig. A.41 for its visualization). As shown in Fig. A.42, the predicted depth yields higher correlation with the ground truth compared to control baselines, showing that it can be used to generate 3DCC for datasets without 3D information.

Comparing robust ImageNet models on 2DCC and 3DCC benchmarks

We compare performances of the robust ImageNet models from RobustBench and ImageNet-C leaderboards below (See the links for full model names) in Figure A.39. A quick look at the scatter plot (top) shows that the general trends between 2DCC and 3DCC are similar, an observation also made in [230] even when the corruptions are *designed to be dissimilar to 2DCC*. Hence, this is also expected for our case as 2D and 3D corruptions are not completely disjoint (expected). But, we observe notable differences between the two benchmarks in local trends, e.g. in the ellipsoid regions certain robustness mechanisms improved performance on 2DCC while being ineffective against 3DCC.

The bar plot (bottom) comparing a subset of corruptions that exists in both benchmarks

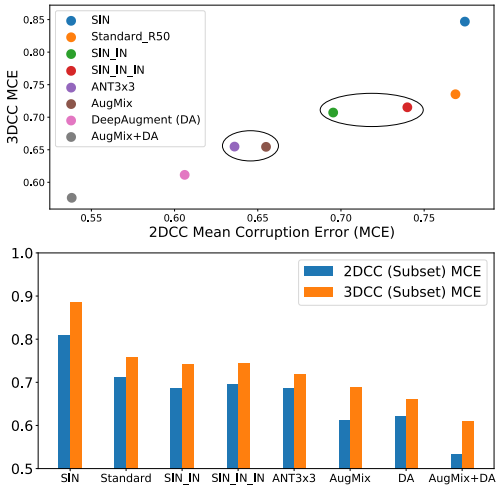

Figure A.39: Comparing ImageNet models on 2DCC and 3DCC benchmarks (i.e. ImageNet-C vs ImageNet-3DCC). **Top:** Comparison of mean corruption errors (MCEs) on 2DCC and 3DCC. **Bottom:** Comparison of MCEs for a subset of corruptions that exists in both benchmarks (e.g. 2D defocus blur vs its 3D version). See Sec. A.3.1 for details.

(e.g. 2D defocus blur vs its 3D version), further reflects the differences where **1.** all models have *consistently higher* errors on 3D corruptions compared to their 2DCC counterparts and **2.** certain models, e.g. AugMix and AugMix+DA, face a larger drop in performance on 3DCC compared to the other models, indicating that AugMix may be biased towards 2DCC. Thus, 3DCC evaluations can be informative during model development as they **expose nonlinear trends and vulnerabilities that are not captured by 2DCC** (also discussed in Sec. 5.2.2).

Performance of 3D data augmentation

We show in Fig. A.50 the performance of 3D data augmentation and baselines on individual corruptions from 3DCC for surface normals estimation task. Similarly, Fig. A.51 shows the performance on 2DCC and Table A.15 provides full performance metrics on OASIS [270] benchmark. As can be seen from the results, 3D data augmentation notably boosts robustness.

A.3.2 Qualitative Results

Video evaluations

We perform evaluations on clips from manually collected DSLR data, YouTube videos, Adobe After Affects (AE) generated corrupted data (Sec.5.2.3 in the main paper), and sample queries from OASIS [270]. They suggest the proposed 3D data augmentation yields notably sharper and more accurate predictions with less flickering, compared to

baselines. We recommend watching the clips on the project page.

Additional queries

In addition to video evaluations in Sec. A.3.2 we provided additional results on OASIS and AE datasets in Figures A.52 and A.53, again suggesting 3D data augmentation is beneficial for improving robustness.

A.3.3 Further method details

Data augmentation mechanisms

Below we provide additional details about the training procedures of data augmentation models. All models were finetuned from the Baseline UNet (T+UNet) with an equal number of clean and augmented images.

Adversarial training: The adversarial examples are generated from an I-FSGM attack with $\epsilon = (0 - 16]$. To generate the I-FGSM attack [10], we apply the following:

$$X_0 = X, \tag{A.8}$$

$$X_{n+1} = Clip_{X,\epsilon}\{X_n + \alpha \text{sign}(\nabla J(X_n, y))\} \tag{A.9}$$

where J is the loss function. Similar to [10], we set $\alpha = 1$ in our experiments and the number of iterations given by $N = \min(4 + \epsilon, 1.25\epsilon)$.

Style [96]: We applied the AdaIN style transfer [150]. The stylization coefficient is randomly selected from the range $[0.1, 0.5]$.

DeepAugment [7]: We use the same perturbations as [7], with the exception of the ones that change the scene geometry, e.g. rotation, flipping.

Implementing corruptions

We release the full open source code of our pipeline, which enables using the implemented corruptions on any dataset. Below, we provide further details about implementing corruptions.

Depth of field: We divide the scene into two regions using hyperfocal distance [279] which is the focus distance yielding the largest depth of field. We then define *near focus* region as parts of the scene closer to camera than hyperfocal distance (and vice versa for far focus). After picking the focus region, we perform the blurring (See Fig. 3 right of main paper for an illustration.)

Video: We use *ffmpeg* scripts to generate video-based corruptions.

Semantics: We use Replica[87] dataset which comes with 3D mesh annotations. To

	Angular error°		% within t°			Relative Normal	
	Mean	Median	11.25°	22.5°	30°	AUCo	AUCp
T+UNet	30.49	22.93	23.18	49.24	61.12	0.6095	0.5953
T+DPT	32.13	25.68	18.82	44.06	57.13	0.6078	0.5484
OASIS	24.63	19.06	30.10	57.34	69.91	0.5693	0.5490
O+DPT	24.42	18.46	28.82	59.53	72.39	0.6320	0.5484
O+DPT+2DCC	23.67	17.75	30.24	61.22	73.83	0.6287	0.6806
O+DPT+2DCC+3D **(Ours)**	24.65	18.53	28.89	58.97	71.89	0.6251	0.6796
Ours (+X-TC [72])	23.89	18.34	28.66	60.00	73.29	0.6264	0.6928

Table A.15: **Evaluations on OASIS.** Similar to Table 1 in the main paper, but results for more metrics are shown.

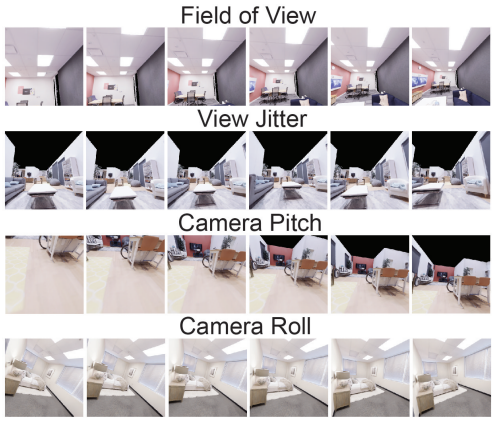

Figure A.40: Visualizations of view change corruptions from 3DCC for different sampled angles.

obtain occlusion masks, while it is possible to perform ray-tracing, it could be expensive and time-consuming. Thus, we also investigated an alternative approach and modified the mesh by removing all the objects except the target one and rendering semantic masks. Performing a second rendering when all objects are in the mesh, i.e. original state, yields the semantic masks with occlusions (e.g. blue masks in Fig 1 and Fig. 4 in the main paper). The difference between the two masks correspond to occlusion mask (e.g. red masks in Fig. 1 and Fig. 4 in the main paper).

A.3.4 Visualizing Corruptions

We show visualizations of corruptions from 3DCC and 2DCC for 5 shift intensities in Figures A.54 and A.55, respectively. Furthermore, we also show samples from *view changes* corruptions from 3DCC in Fig. A.40.

Figure A.41: Statistically informed blind guess depth prediction of Taskonomy dataset [142, 72].

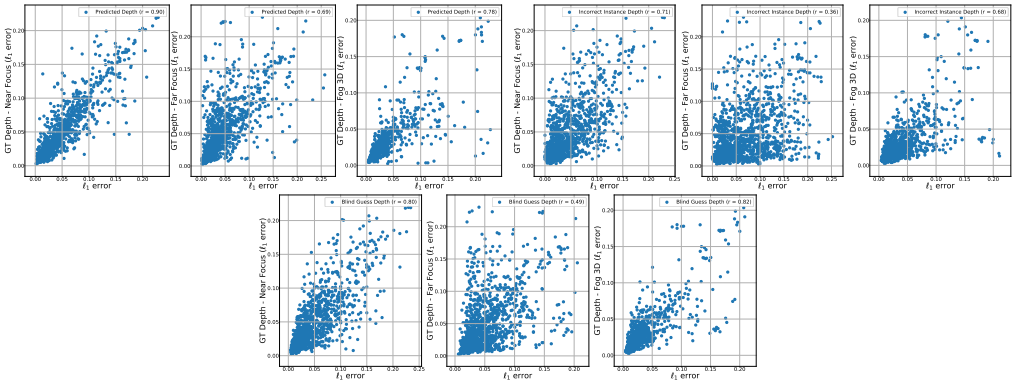

Figure A.42: **Effectiveness of applying 3DCC without ground truth depth.** Similar to Fig. 12 in the main paper, but control baselines are also provided, namely *incorrect instance depth* and *blind guess depth*. The predicted depth from MiDaS [268] model yields the strongest correlations with the ground truth depth.

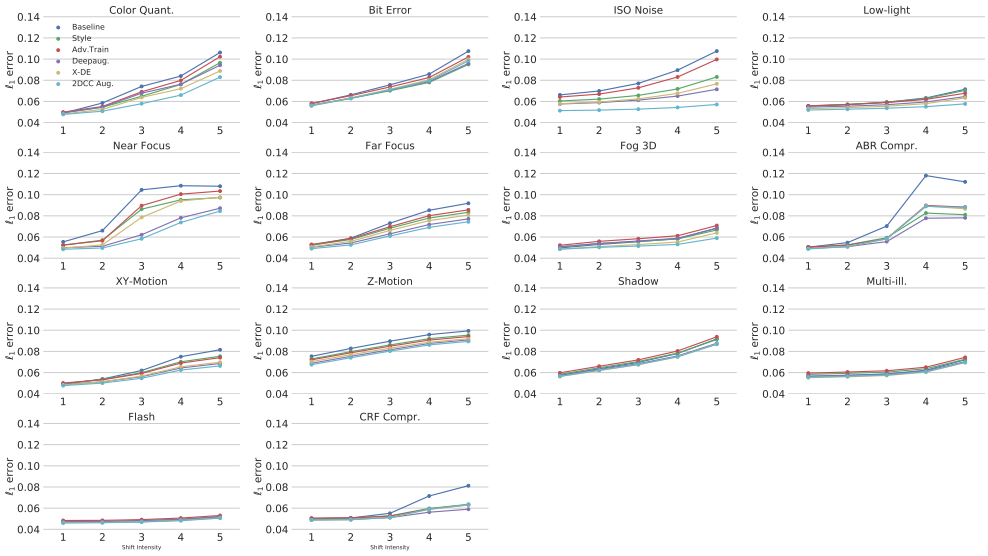

Figure A.43: **Average ℓ_1 losses of robustness mechanisms against corruptions in 3DCC for surface normals estimation.** Similar to Fig. 6 in the main paper, but demonstrates the performance for individual corruptions in 3DCC.

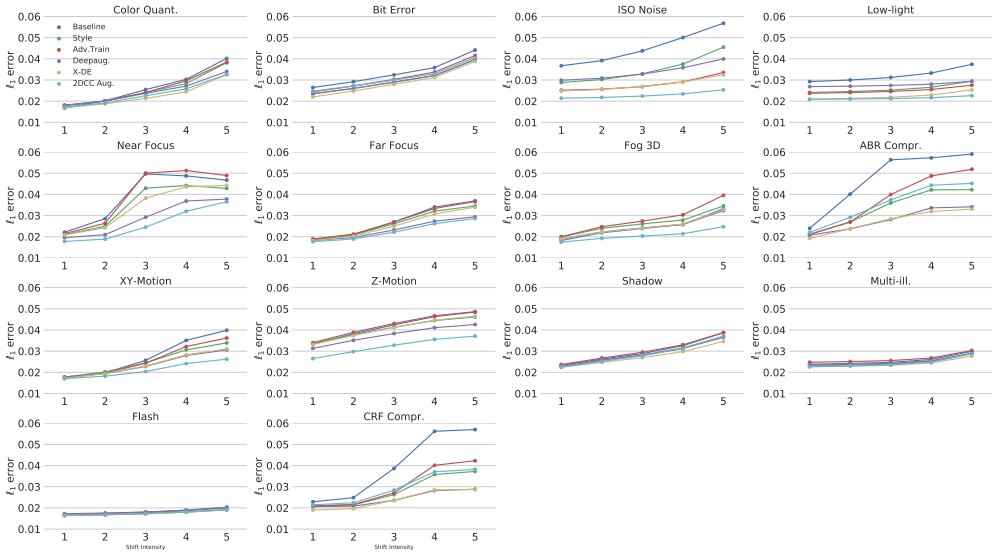

Figure A.44: **Average ℓ_1 losses of robustness mechanisms against corruptions in 3DCC for depth estimation.** Similar to Fig. 6 in the main paper, but demonstrates the performance for individual corruptions in 3DCC.

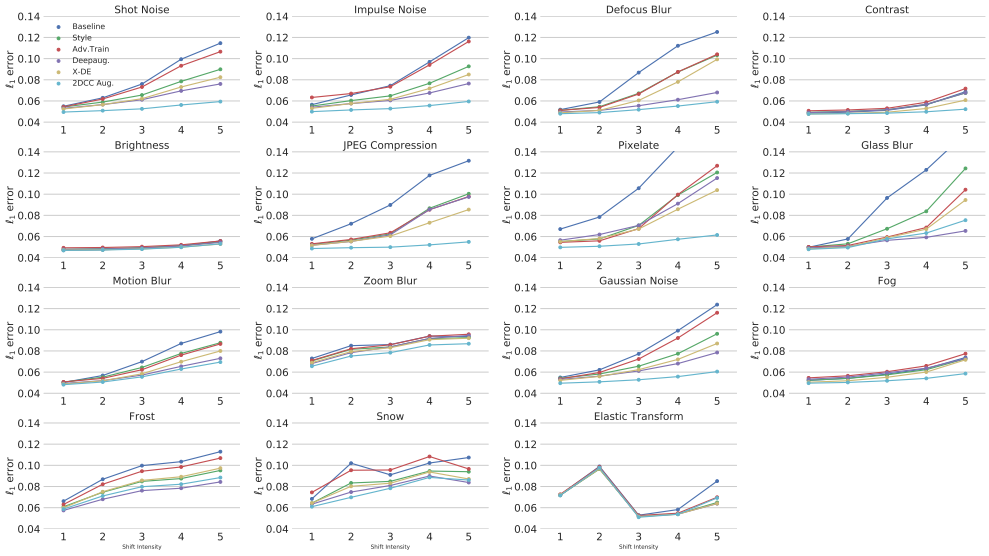

Figure A.45: **Average ℓ_1 losses of robustness mechanisms against corruptions in 2DCC for surface normals estimation.** Similar to Fig. 6 in the main paper, but demonstrates the performance for individual corruptions in 2DCC.

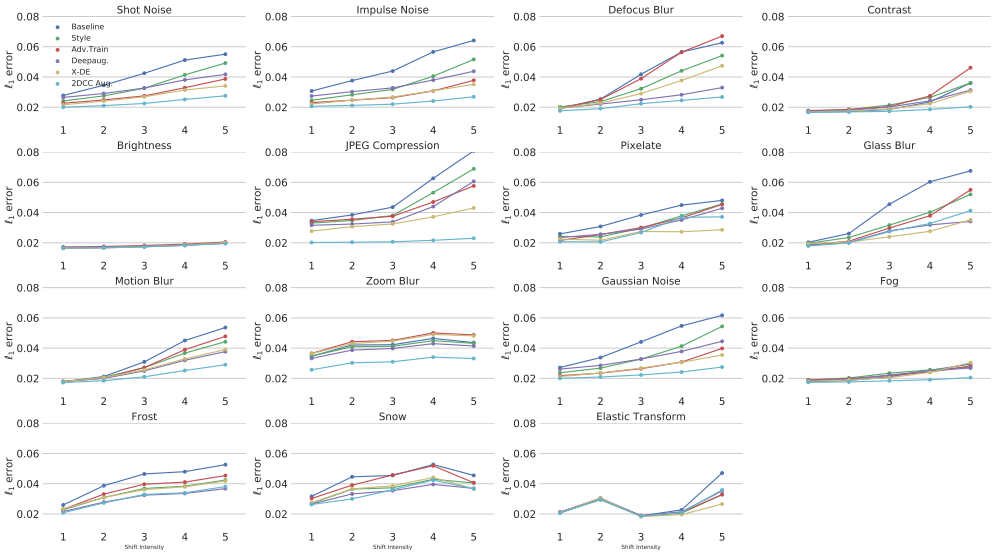

Figure A.46: **Average ℓ_1 losses of robustness mechanisms against corruptions in 2DCC for depth estimation.** Similar to Fig. 6 in the main paper, but demonstrates the performance for individual corruptions in 2DCC.

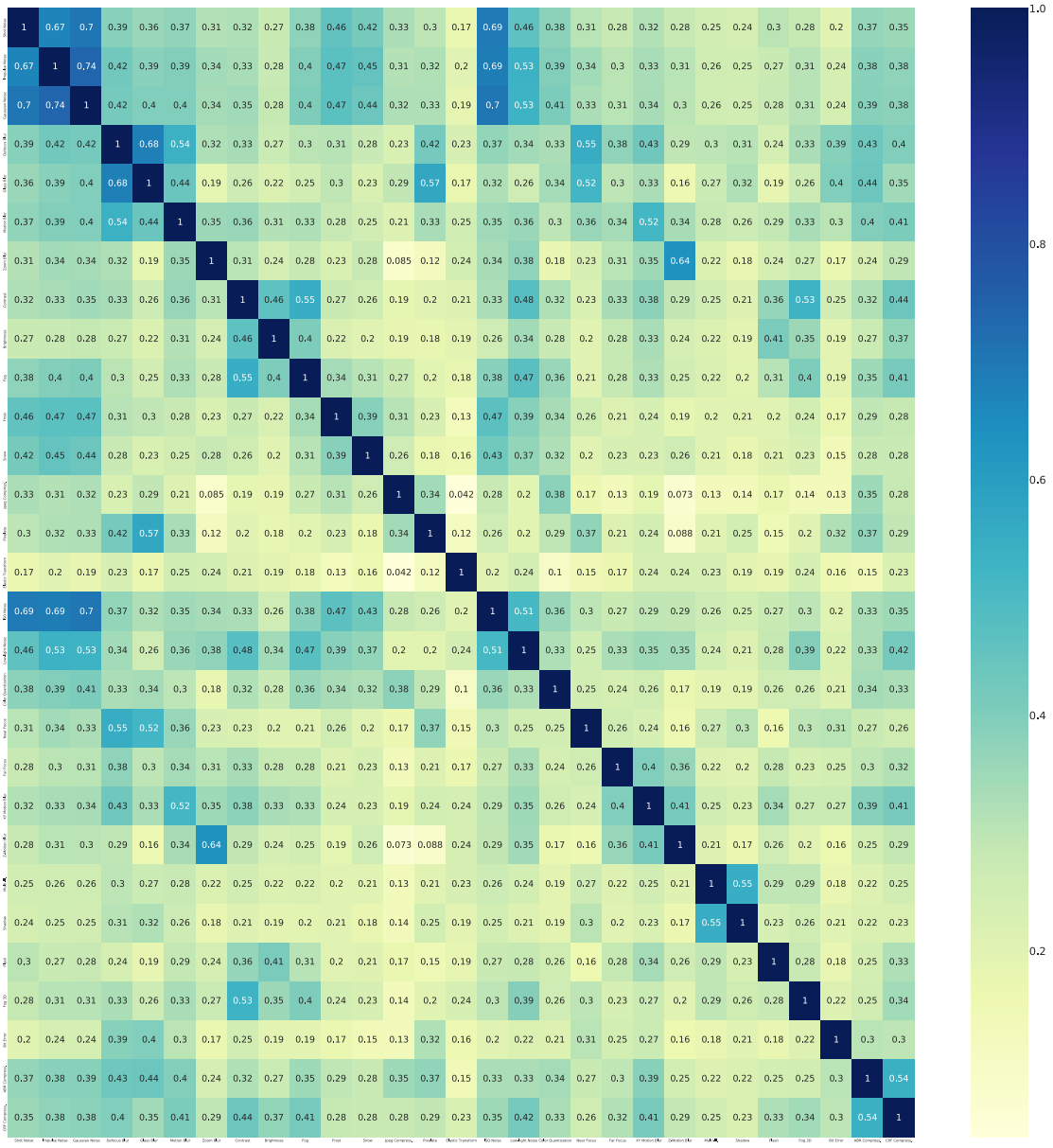

Figure A.47: **Redundancy among corruptions in 2DCC and 3DCC in the ℓ_1 errors of surface normals prediction.** Similar to Fig. 9 in the main paper, but are shown for all corruptions.

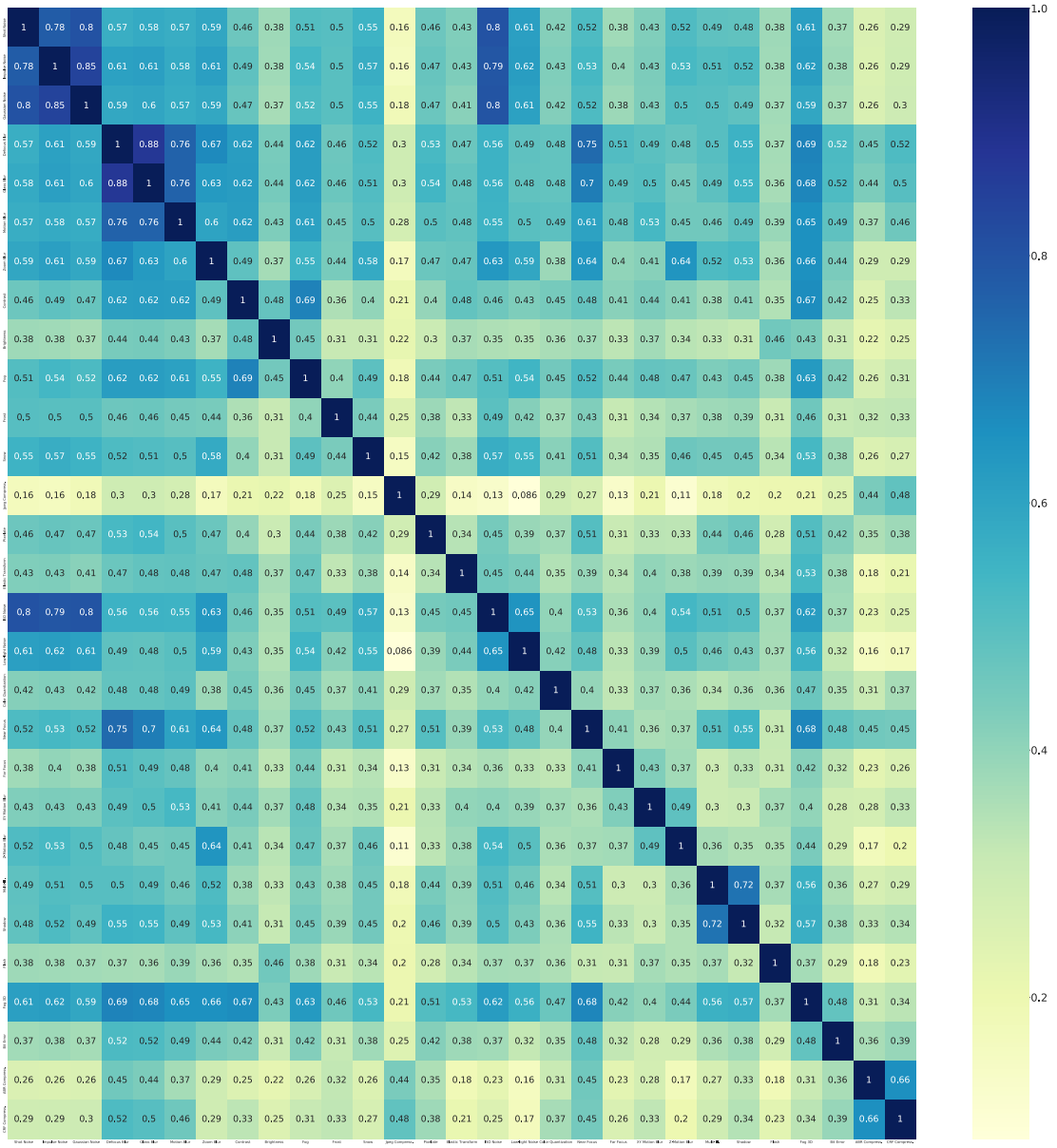

Figure A.48: **Redundancy among corruptions in 2DCC and 3DCC in the ℓ_1 errors of depth prediction.** Similar to Fig. 9 in the main paper, but are shown for all corruptions.

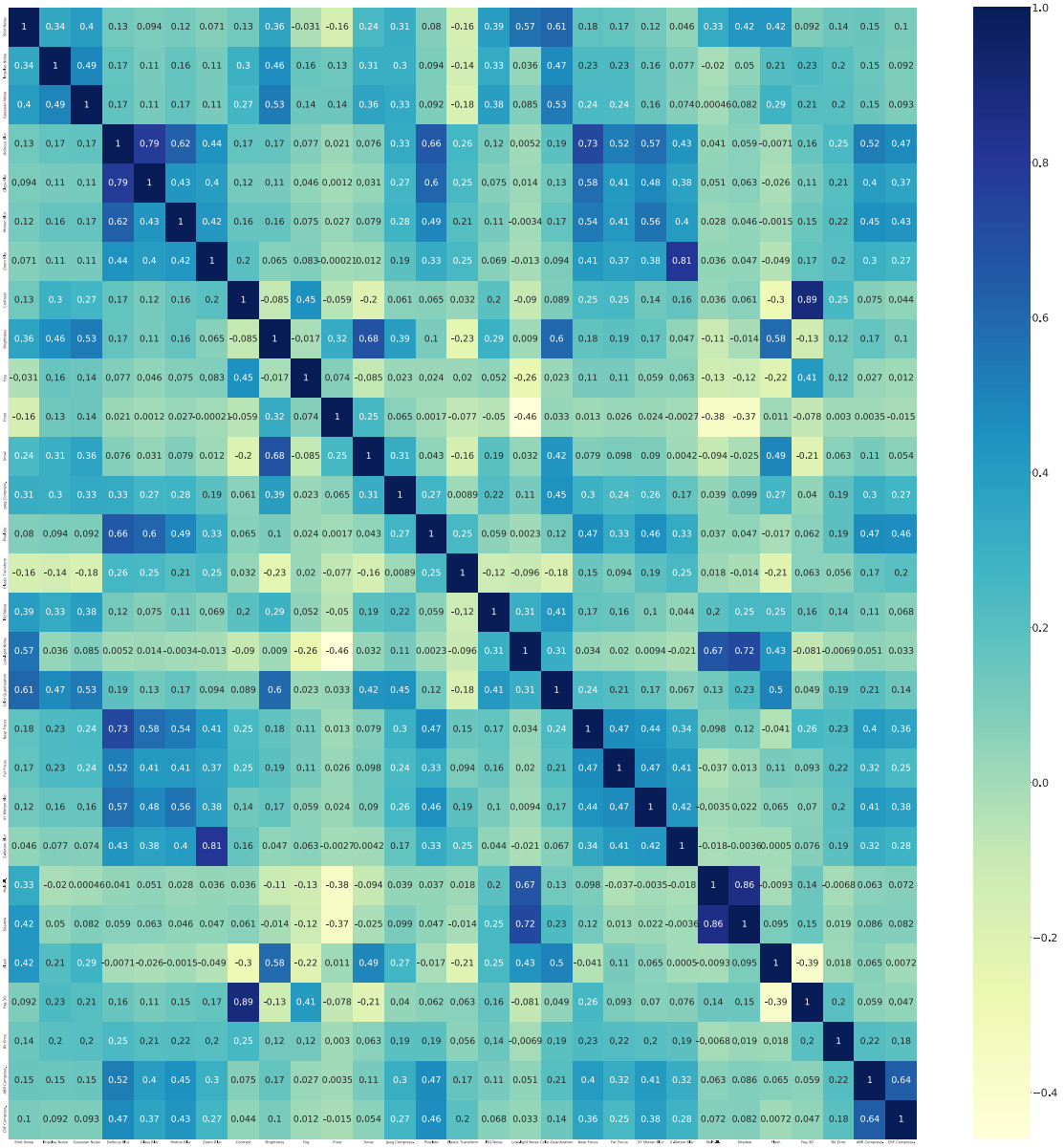

Figure A.49: **Redundancy among corruptions in 2DCC and 3DCC in the ℓ_1 errors of RGB images.** Similar to Fig. 9 in the main paper, but are shown for all corruptions.

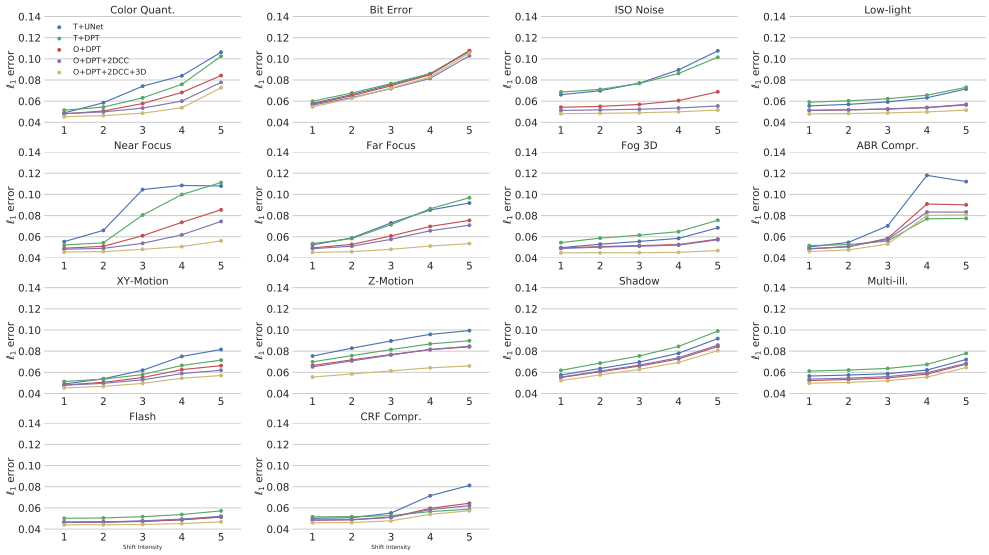

Figure A.50: **Average ℓ_1 losses of 3D data augmentation and baselines against corruptions in 3DCC for surface normals estimation.** Similar to Tab. 1 in the main paper, but demonstrates the performance for individual corruptions in 3DCC.

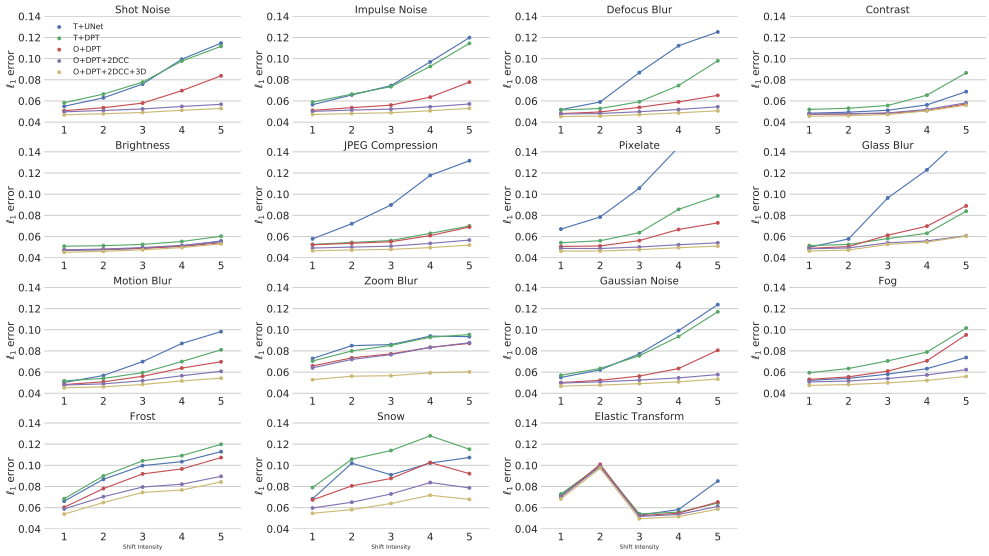

Figure A.51: **Average ℓ_1 losses of 3D data augmentation and baselines against corruptions in 2DCC for surface normals estimation.** Similar to Tab. 1 in the main paper, but demonstrates the performance for individual corruptions in 2DCC.

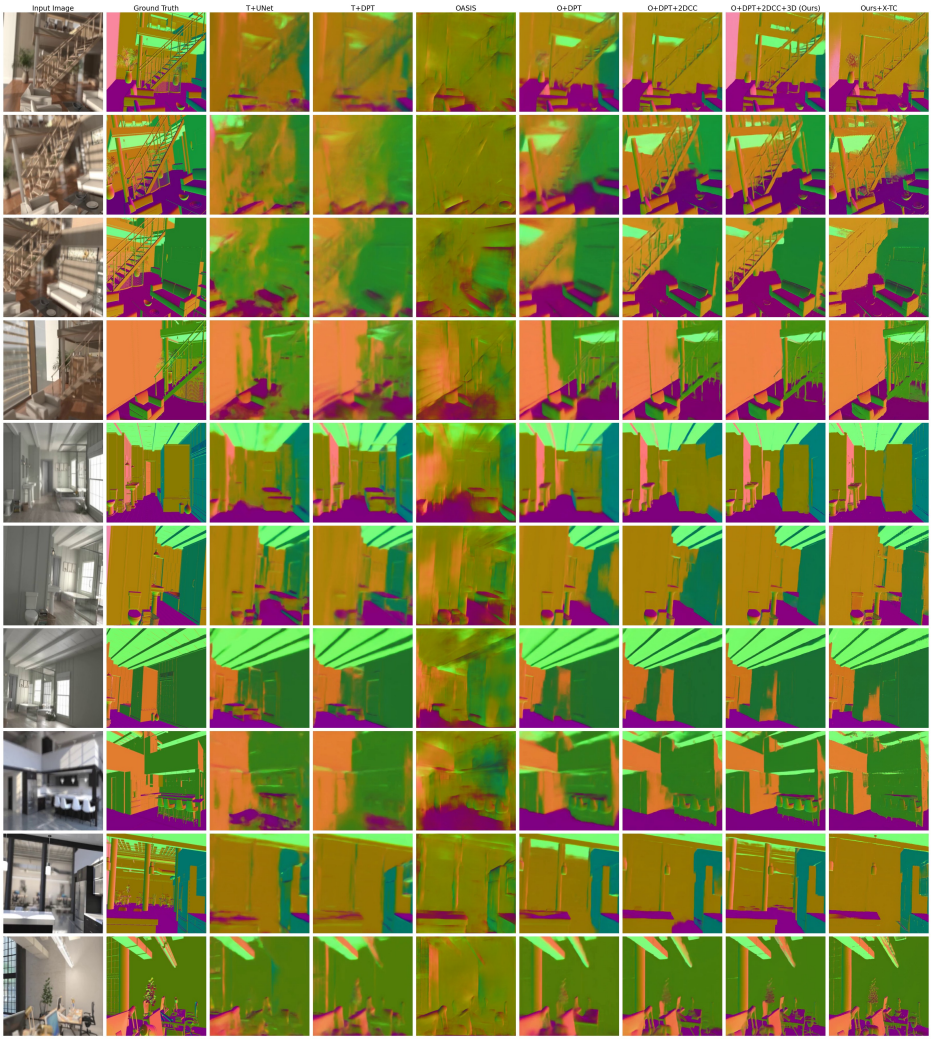

Figure A.52: **Qualitative results on corruptions generated with After Effects.** An extension of the qualitative results in Fig. 8 of the main paper showing the benefits of 3D augmentation. See Sec. 5.2.3 of the main paper for details on how the corruptions were generated.

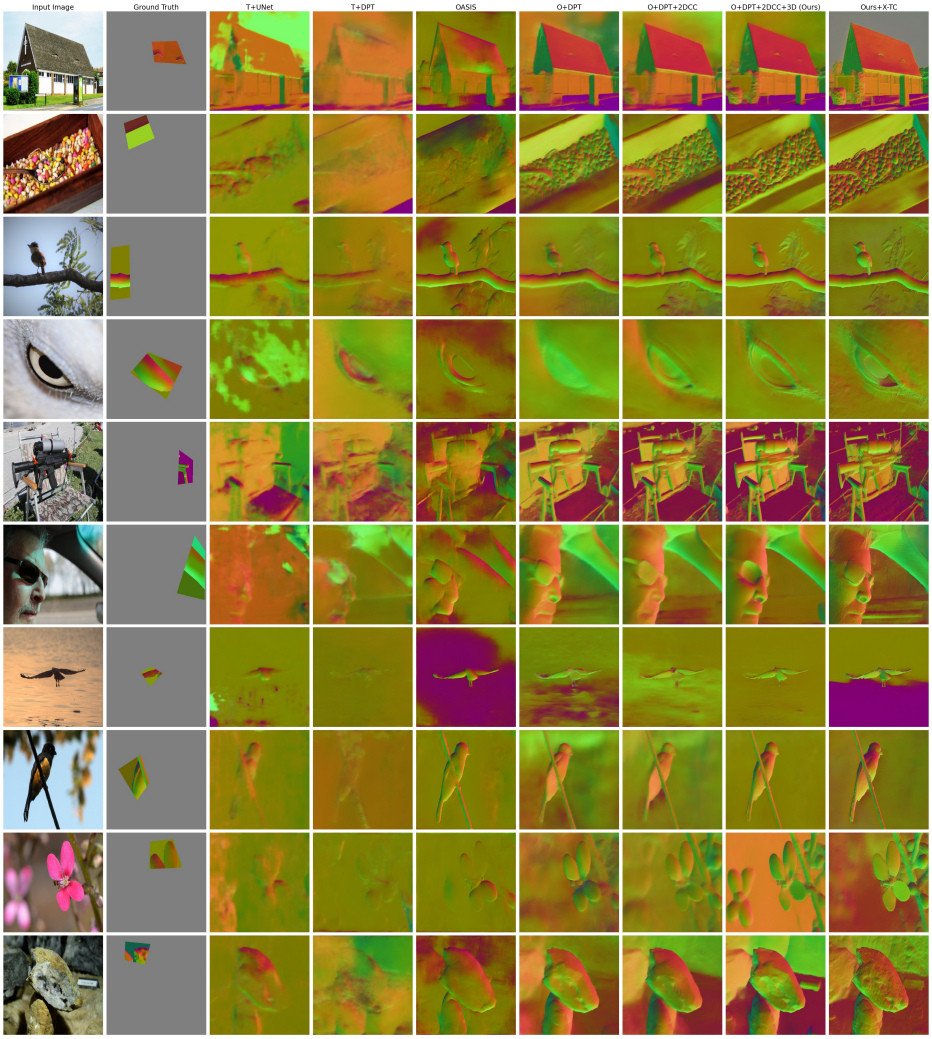

Figure A.53: **Qualitative results on OASIS [270].** An extension of the qualitative results in Fig. 8 of the main paper showing the benefits of 3D augmentation.

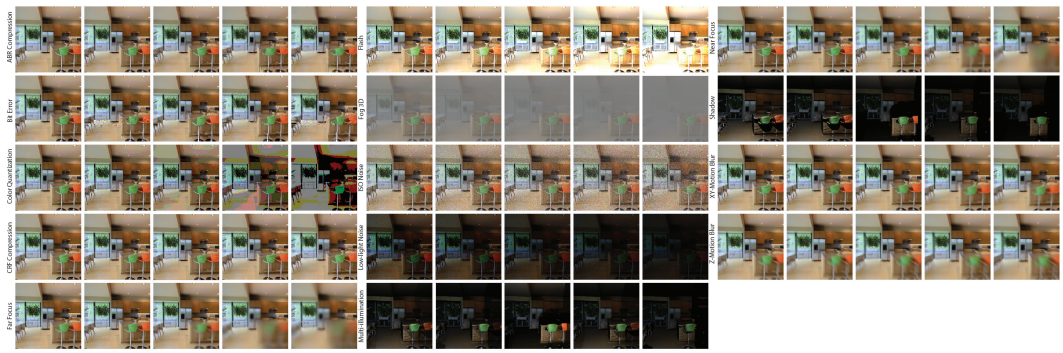

Figure A.54: Visualizations of corruptions from 3DCC for 5 shift intensities.

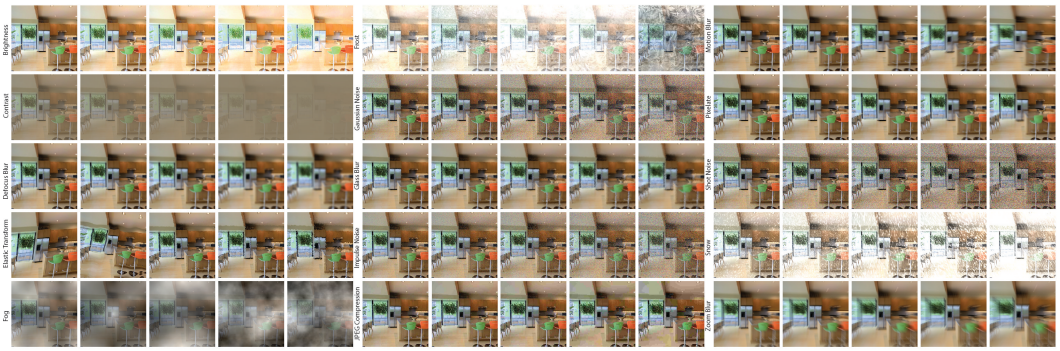

Figure A.55: Visualizations of corruptions from 2DCC for 5 shift intensities.

Bibliography

[1] O. Wiles, S. Gowal, F. Stimberg, S. Alvise-Rebuffi, I. Ktena, K. Dvijotham, and T. Cemgil, "A fine-grained analysis on distribution shift," *arXiv preprint arXiv:2110.11328*, 2021.

[2] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[3] Y. Dong, S. Ruan, H. Su, C. Kang, X. Wei, and J. Zhu, "Viewfool: Evaluating the robustness of visual recognition to adversarial viewpoints," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36789–36803, 2022.

[4] O. F. Kar, T. Yeo, A. Atanov, and A. Zamir, "3d common corruptions and data augmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18963–18974, 2022.

[5] Y. Wang, X. Chen, Y. You, L. E. Li, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "Train in germany, test in the usa: Making 3d object detectors generalize," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11713–11723, 2020.

[6] A. Dubey, V. Ramanathan, A. Pentland, and D. Mahajan, "Adaptive methods for real-world domain generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14340–14349, 2021.

[7] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, *et al.*, "The many faces of robustness: A critical analysis of out-of-distribution generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021.

[8] D. Dai and L. Van Gool, "Dark model adaptation: Semantic image segmentation from daytime to nighttime," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3819–3824, IEEE, 2018.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fer-

gus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[10] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[12] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International conference on machine learning*, pp. 1802–1811, PMLR, 2019.

[13] A. Atanov, A. Filatov, T. Yeo, A. Sohmshetty, and A. Zamir, "Task discovery: Finding the tasks that neural networks generalize on," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15702–15717, 2022.

[14] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?," *Journal of Machine Learning Research*, vol. 20, no. 184, pp. 1–25, 2019.

[15] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *IEEE winter conference on applications of computer vision*, pp. 75–82, IEEE, 2014.

[16] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR 2011*, pp. 1521–1528, IEEE, 2011.

[17] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*, pp. 77–91, PMLR, 2018.

[18] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International Conference on Machine Learning*, pp. 5637–5664, PMLR, 2021.

[19] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The pitfalls of simplicity bias in neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9573–9585, 2020.

[20] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

[21] F. Wenzel, J. Snoek, D. Tran, and R. Jenatton, "Hyperparameter ensembles for robustness and uncertainty quantification," *arXiv preprint arXiv:2006.13570*, 2020.

[22] S. Zaidi, A. Zela, T. Elsken, C. Holmes, F. Hutter, and Y. W. Teh, "Neural ensemble search for performant and calibrated predictions," *arXiv preprint arXiv:2006.08573*, 2020.

[23] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," *arXiv preprint arXiv:1901.08846*, 2019.

[24] S. Kariyappa and M. K. Qureshi, "Improving adversarial robustness of ensembles with diversity training," *arXiv preprint arXiv:1901.09981*, 2019.

[25] H. Yang, J. Zhang, H. Dong, N. Inkawhich, A. Gardner, A. Touchet, W. Wilkes, H. Berry, and H. Li, "Dverge: Diversifying vulnerabilities for enhanced robust generation of ensembles," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[26] A. Rame and M. Cord, "Dice: Diversity in deep ensembles via conditional redundancy adversarial estimation," *arXiv preprint arXiv:2101.05544*, 2021.

[27] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, *et al.*, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," in *International Conference on Machine Learning*, pp. 23965–23998, PMLR, 2022.

[28] A. Rame, M. Kirchmeyer, T. Rahier, A. Rakotomamonjy, P. Gallinari, and M. Cord, "Diverse weight averaging for out-of-distribution generalization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10821–10836, 2022.

[29] A. Rame, K. Ahuja, J. Zhang, M. Cord, L. Bottou, and D. Lopez-Paz, "Model ratatouille: Recycling diverse models for out-of-distribution generalization," 2023.

[30] R. He, S. Sun, X. Yu, C. Xue, W. Zhang, P. Torr, S. Bai, and X. Qi, "Is synthetic data from generative models ready for image recognition?," *arXiv preprint arXiv:2210.07574*, 2022.

[31] J. Yuan, F. Pinto, A. Davies, A. Gupta, and P. Torr, "Not just pretty pictures: Text-to-image generators enable interpretable interventions for robust representations," *arXiv preprint arXiv:2212.11237*, 2022.

[32] M. B. Sariyildiz, K. Alahari, D. Larlus, and Y. Kalantidis, "Fake it till you make it: Learning transferable representations from synthetic imagenet clones," in *CVPR 2023–IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[33] V. Besnier, H. Jain, A. Bursuc, M. Cord, and P. Pérez, "This dataset does not exist: training models from generated images," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2020.

[34] A. Jahanian, X. Puig, Y. Tian, and P. Isola, "Generative models as a data source for multiview representation learning," *arXiv preprint arXiv:2106.05258*, 2021.

[35] E. Wong and J. Z. Kolter, "Learning perturbation sets for robust machine learning," *arXiv preprint arXiv:2007.08450*, 2020.

[36] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple data processing method to improve robustness and uncertainty," *arXiv preprint arXiv:1912.02781*, 2019.

[37] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[38] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

[39] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

[40] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[41] E. Rusak, L. Schott, R. S. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel, "A simple way to make neural networks robust against diverse image corruptions," in *European Conference on Computer Vision*, pp. 53–69, Springer, 2020.

[42] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," in *Advances in Neural Information Processing Systems*, pp. 13276–13286, 2019.

[43] G. Blanchard, G. Lee, and C. Scott, "Generalizing from several related classification tasks to a new unlabeled sample," *Advances in neural information processing systems*, vol. 24, 2011.

[44] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *International conference on machine learning*, pp. 10–18,

PMLR, 2013.

[45] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[46] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[47] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pp. 443–450, Springer, 2016.

[48] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5400–5409, 2018.

[49] A. Rame, C. Dancette, and M. Cord, "Fishr: Invariant gradient variances for out-of-distribution generalization," in *International Conference on Machine Learning*, pp. 18347–18377, PMLR, 2022.

[50] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[51] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," *arXiv preprint arXiv:1911.08731*, 2019.

[52] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv preprint arXiv:2007.01434*, 2020.

[53] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *International Conference on Learning Representations*, 2020.

[54] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International Conference on Machine Learning*, pp. 6028–6039, PMLR, 2020.

[55] J. Baxter, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.

[56] I. P. Howard and B. J. Rogers, *Seeing in depth, Vol. 2: Depth perception.* University

of Toronto Press, 2002.

[57] I. P. Howard, B. J. Rogers, *et al.*, *Binocular vision and stereopsis*. Oxford University Press, USA, 1995.

[58] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[59] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[60] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, pp. 1–15, Springer, 2000.

[61] Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma, "Rethinking bias-variance trade-off for generalization of neural networks," *arXiv preprint arXiv:2002.11328*, 2020.

[62] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural Safety*, vol. 31, no. 2, pp. 105–112, 2009.

[63] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Advances in Neural Information Processing Systems*, pp. 5574–5584, 2017.

[64] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, pp. 1050–1059, 2016.

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[66] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

[67] A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, pp. 2348–2356, 2011.

[68] C. Louizos and M. Welling, "Multiplicative normalizing flows for variational bayesian neural networks," *arXiv preprint arXiv:1703.01961*, 2017.

[69] C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix gaussian posteriors," in *International Conference on Machine Learning*, pp. 1708–1716, 2016.

[70] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," *arXiv preprint arXiv:1803.04386*, 2018.

[71] C. Riquelme, G. Tucker, and J. Snoek, "Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling," *arXiv preprint arXiv:1802.09127*, 2018.

[72] A. Zamir, A. Sax, T. Yeo, O. Kar, N. Cheerla, R. Suri, Z. Cao, J. Malik, and L. Guibas, "Robust learning through cross-task consistency," *arXiv preprint arXiv:2006.04096*, 2020.

[73] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, pp. 1321–1330, 2017.

[74] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *International Conference on Machine Learning*, pp. 2796–2804, 2018.

[75] D. Hafner, D. Tran, T. Lillicrap, A. Irpan, and J. Davidson, "Noise contrastive priors for functional uncertainty," in *Uncertainty in Artificial Intelligence*, pp. 905–914, PMLR, 2020.

[76] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint arXiv:1812.04606*, 2018.

[77] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[78] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," *arXiv preprint arXiv:1711.09325*, 2017.

[79] M. Leordeanu, M. Pirvu, D. Costea, A. Marcu, E. Slusanschi, and R. Sukthankar, "Semi-supervised learning for multi-task scene understanding by neural graph consensus," *arXiv preprint arXiv:2010.01086*, 2020.

[80] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2223–2232, 2017.

[81] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 675–684, 2018.

[82] I. Kim, Y. Kim, and S. Kim, "Learning loss for test-time augmentation," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[83] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," *arXiv preprint arXiv:2002.06470*, 2020.

[84] N. Ford, J. Gilmer, N. Carlini, and D. Cubuk, "Adversarial examples are a natural consequence of test error in noise," *arXiv preprint arXiv:1901.10513*, 2019.

[85] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, pp. 13991–14002, 2019.

[86] J. Hartung, G. Knapp, and B. K. Sinha, *Statistical meta-analysis with applications*, vol. 738. John Wiley & Sons, 2011.

[87] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[88] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722, 2018.

[89] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347, 2019.

[90] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[91] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[92] T. Acharya and A. K. Ray, *Image processing: principles and applications*. John Wiley & Sons, 2005.

[93] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[94] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 234–241, Springer, 2015.

[95] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[96] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018.

[97] Y. Wen, D. Tran, and J. Ba, "Batchensemble: an alternative approach to efficient ensemble and lifelong learning," in *International Conference on Learning Representations*, 2020.

[98] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[99] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[100] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[101] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[102] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt, "Measuring robustness to natural distribution shifts in image classification," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18583–18599, 2020.

[103] J. P. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt, "Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization," in *International Conference on Machine Learning*, pp. 7721–7735, PMLR, 2021.

[104] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[105] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," *arXiv preprint arXiv:2302.05543*, 2023.

[106] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," *arXiv preprint arXiv:2208.01618*, 2022.

[107] S. Jain, H. Lawrence, A. Moitra, and A. Madry, "Distilling model failures as directions in latent space," *arXiv preprint arXiv:2206.14754*, 2022.

[108] C. Mou, X. Wang, L. Xie, J. Zhang, Z. Qi, Y. Shan, and X. Qie, "T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models," *arXiv preprint arXiv:2302.08453*, 2023.

[109] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, "Prompt-to-prompt image editing with cross attention control," *arXiv preprint arXiv:2208.01626*, 2022.

[110] G. Couairon, J. Verbeek, H. Schwenk, and M. Cord, "Diffedit: Diffusion-based semantic image editing with mask guidance," *arXiv preprint arXiv:2210.11427*, 2022.

[111] T. Brooks, A. Holynski, and A. A. Efros, "Instructpix2pix: Learning to follow image editing instructions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18392–18402, 2023.

[112] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or, "Null-text inversion for editing real images using guided diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6038–6047, 2023.

[113] I. Huberman-Spiegelglas, V. Kulikov, and T. Michaeli, "An edit friendly ddpm noise space: Inversion and manipulations," *arXiv preprint arXiv:2304.06140*, 2023.

[114] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22500–22510, 2023.

[115] O. Avrahami, K. Aberman, O. Fried, D. Cohen-Or, and D. Lischinski, "Break-a-scene: Extracting multiple concepts from a single image," *arXiv preprint arXiv:2305.16311*, 2023.

[116] L. Han, Y. Li, H. Zhang, P. Milanfar, D. Metaxas, and F. Yang, "Svdiff: Compact parameter space for diffusion fine-tuning," *arXiv preprint arXiv:2303.11305*, 2023.

[117] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, pp. 2256–2265, PMLR, 2015.

[118] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[119] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[120] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[121] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," *arXiv preprint arXiv:2108.01073*, 2021.

[122] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein, "Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery," *arXiv preprint arXiv:2302.03668*, 2023.

[123] S. Beery, E. Cole, and A. Gjoka, "The iwildcam 2020 competition dataset," *arXiv preprint arXiv:2004.10340*, 2020.

[124] S. Beery, A. Agarwal, E. Cole, and V. Birodkar, "The iwildcam 2021 competition dataset," *arXiv preprint arXiv:2105.03494*, 2021.

[125] L. Dunlap, A. Umino, H. Zhang, J. Yang, J. E. Gonzalez, and T. Darrell, "Diversify your vision datasets with automatic diffusion-based augmentation," *arXiv preprint arXiv:2305.16289*, 2023.

[126] G. Le Moing, T.-H. Vu, H. Jain, P. Pérez, and M. Cord, "Semantic palette: Guiding scene generation with class proportions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9342–9350, 2021.

[127] M. Zhang, S. Levine, and C. Finn, "Memo: Test time robustness via adaptation and augmentation," *arXiv preprint arXiv:2110.09506*, 2021.

[128] Y. Gandelsman, Y. Sun, X. Chen, and A. A. Efros, "Test-time training with masked autoencoders," *arXiv preprint arXiv:2209.07522*, 2022.

[129] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk, "Improving robustness without sacrificing accuracy with patch gaussian augmentation," *arXiv preprint arXiv:1906.02611*, 2019.

[130] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," in *International Conference on Machine Learning*, pp. 2712–2721, PMLR, 2019.

[131] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2020.

[132] A. Eftekhar, A. Sax, J. Malik, and A. Zamir, "Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10786–10796, 2021.

[133] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[134] S. M. Xie, A. Kumar, R. Jones, F. Khani, T. Ma, and P. Liang, "In-n-out: Pre-training and self-training using auxiliary information for out-of-distribution robustness," *arXiv preprint arXiv:2012.04550*, 2020.

[135] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, 2022.

[136] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

[137] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International conference on machine learning*, pp. 2990–2999, PMLR, 2016.

[138] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," *arXiv preprint arXiv:2103.14586*, 2021.

[139] R. Shao, Z. Shi, J. Yi, P.-Y. Chen, and C.-J. Hsieh, "On the adversarial robustness of visual transformers," *arXiv preprint arXiv:2103.15670*, 2021.

[140] X. Mao, G. Qi, Y. Chen, X. Li, R. Duan, S. Ye, Y. He, and H. Xue, "Towards robust vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12042–12051, 2022.

[141] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for

the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

[142] T. Yeo, O. F. Kar, and A. Zamir, "Robustness via cross-domain ensembles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12189–12199, October 2021.

[143] S. Jain, D. Tsipras, and A. Madry, "Combining diverse feature priors," in *International Conference on Machine Learning*, pp. 9802–9832, PMLR, 2022.

[144] M. Pagliardini, M. Jaggi, F. Fleuret, and S. P. Karimireddy, "Agree to disagree: Diversity through disagreement for better transferability," *arXiv preprint arXiv:2202.04414*, 2022.

[145] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[146] D. Kang, D. Dhar, and A. Chan, "Incorporating side information by adaptive convolution," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[147] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[148] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," *arXiv preprint arXiv:1610.07629*, 2016.

[149] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," *arXiv preprint arXiv:1705.06830*, 2017.

[150] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1501–1510, 2017.

[151] B. Oreshkin, P. Rodríguez López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," *Advances in neural information processing systems*, vol. 31, 2018.

[152] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, "Fast and flexible multi-task classification using conditional neural adaptive processes," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[153] L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson, "Fast context adaptation via meta-learning," in *International Conference on Machine Learning*,

pp. 7693–7702, PMLR, 2019.

[154] E. Triantafillou, H. Larochelle, R. Zemel, and V. Dumoulin, "Learning a universal template for few-shot dataset generalization," in *International Conference on Machine Learning*, pp. 10424–10433, PMLR, 2021.

[155] X. Jiang, M. Havaei, F. Varno, G. Chartrand, N. Chapados, and S. Matwin, "Learning to learn with conditional class dependencies," in *international conference on learning representations*, 2019.

[156] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio, "Feature-wise transformations," *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations.

[157] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[158] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[159] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4733–4742, 2016.

[160] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.

[161] A. R. Zamir, T.-L. Wu, L. Sun, W. B. Shen, B. E. Shi, J. Malik, and S. Savarese, "Feedback networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1308–1317, 2017.

[162] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*, pp. 1126–1135, PMLR, 2017.

[163] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. Freitas, and J. Sohl-Dickstein, "Learned optimizers that scale and generalize," in *International conference on machine learning*, pp. 3751–3760, PMLR, 2017.

[164] W.-C. Ma, S. Wang, J. Gu, S. Manivasagam, A. Torralba, and R. Urtasun, "Deep feedback inverse problem solver," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pp. 229–246, Springer, 2020.

[165] T. Chen, X. Chen, W. Chen, H. Heaton, J. Liu, Z. Wang, and W. Yin, "Learning to optimize: A primer and a benchmark," 2021.

[166] B. Amos, "Tutorial on amortized optimization for learning to optimize over continuous domains," *arXiv preprint arXiv:2202.00665*, 2022.

[167] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

[168] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[169] J. Marino, Y. Yue, and S. Mandt, "Iterative amortized inference," in *International Conference on Machine Learning*, pp. 3403–3412, PMLR, 2018.

[170] T.-H. Wang, F.-E. Wang, J.-T. Lin, Y.-H. Tsai, W.-C. Chiu, and M. Sun, "Plug-and-play: Improve depth prediction via sparse data propagation," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5880–5886, IEEE, 2019.

[171] Y. Yang, A. Wong, and S. Soatto, "Dense depth posterior (ddp) from single image and sparse range," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3353–3362, 2019.

[172] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning joint 2d-3d representations for depth completion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10023–10032, 2019.

[173] Y. Verdié, J. Song, B. Mas, B. Busam, A. Leonardis, and S. McDonagh, "Cromo: Cross-modal learning for monocular depth estimation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3927–3937, IEEE, 2022.

[174] W. Yin, J. Zhang, O. Wang, S. Niklaus, S. Chen, and C. Shen, "Towards domain-agnostic depth completion," *arXiv preprint arXiv:2207.14466*, 2022.

[175] I. Chugunov, Y. Zhang, Z. Xia, X. Zhang, J. Chen, and F. Heide, "The implicit values of a good hand shake: Handheld multi-frame neural depth refinement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2852–2862, 2022.

[176] J. Watson, O. Mac Aodha, V. Prisacariu, G. Brostow, and M. Firman, "The temporal opportunist: Self-supervised multi-frame monocular depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*,

pp. 1164–1174, 2021.

[177] Y. Kuznietsov, M. Proesmans, and L. Van Gool, "Comoda: Continuous monocular depth adaptation using past experiences," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2907–2917, 2021.

[178] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Unsupervised monocular depth and ego-motion learning with structure and semantics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[179] X. Luo, J.-B. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent video depth estimation," *ACM Transactions on Graphics (ToG)*, vol. 39, no. 4, pp. 71–1, 2020.

[180] J. Kopf, X. Rong, and J.-B. Huang, "Robust consistent video depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1611–1621, 2021.

[181] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *International conference on machine learning*, pp. 9229–9248, PMLR, 2020.

[182] Y. Liu, P. Kothari, B. van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "Ttt++: When does self-supervised test-time training fail or thrive?," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21808–21820, 2021.

[183] J. Gao, J. Zhang, X. Liu, T. Darrell, E. Shelhamer, and D. Wang, "Back to the source: Diffusion-driven test-time adaptation," *arXiv preprint arXiv:2207.03442*, 2022.

[184] M. Boudiaf, R. Mueller, I. Ben Ayed, and L. Bertinetto, "Parameter-free online test-time adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8344–8353, 2022.

[185] C. Yi, S. Yang, Y. Wang, H. Li, Y.-P. Tan, and A. C. Kot, "Temporal coherent test-time optimization for robust video classification," *arXiv preprint arXiv:2302.14309*, 2023.

[186] S. Niu, J. Wu, Y. Zhang, Z. Wen, Y. Chen, P. Zhao, and M. Tan, "Towards stable test-time adaptation in dynamic wild world," *arXiv preprint arXiv:2302.12400*, 2023.

[187] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3159–3167, 2016.

[188] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *European conference on computer vision*, pp. 549–565, Springer, 2016.

[189] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, "Extreme clicking for efficient object annotation," in *Proceedings of the IEEE international conference on computer vision*, pp. 4930–4939, 2017.

[190] G. Shin, W. Xie, and S. Albanie, "All you need are a few pixels: semantic segmentation with pixelpick," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1687–1697, 2021.

[191] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison, "In-place scene labelling and understanding with implicit scene representation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15838–15847, 2021.

[192] B. Cheng, O. Parkhi, and A. Kirillov, "Pointly-supervised instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2617–2626, 2022.

[193] Y. Xu, Q. Qian, H. Li, R. Jin, and J. Hu, "Weakly supervised representation learning with coarse labels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10593–10601, 2021.

[194] M. Huh, P. Agrawal, and A. A. Efros, "What makes imagenet good for transfer learning?," 2016.

[195] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

[196] L. Castrejon, Y. Aytar, C. Vondrick, H. Pirsiavash, and A. Torralba, "Learning aligned cross-modal representations from weakly aligned data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2940–2949, 2016.

[197] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *Proceedings of the IEEE international conference on computer vision*, pp. 609–617, 2017.

[198] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," *arXiv preprint arXiv:1908.07490*, 2019.

[199] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.

[200] J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelović, J. Ramapuram, J. De Fauw, L. Smaira, S. Dieleman, and A. Zisserman, "Self-supervised multimodal versatile networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 25–37, 2020.

[201] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "Uniter: Universal image-text representation learning," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX*, pp. 104–120, Springer, 2020.

[202] H. Akbari, L. Yuan, R. Qian, W.-H. Chuang, S.-F. Chang, Y. Cui, and B. Gong, "Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24206–24221, 2021.

[203] R. Bachmann, D. Mizrahi, A. Atanov, and A. Zamir, "Multimae: Multi-modal multi-task masked autoencoders," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVII*, pp. 348–367, Springer, 2022.

[204] R. Girdhar, M. Singh, N. Ravi, L. van der Maaten, A. Joulin, and I. Misra, "Omnivore: A single model for many visual modalities," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16102–16112, 2022.

[205] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, *et al.*, "Meta-dataset: A dataset of datasets for learning to learn from few examples," *arXiv preprint arXiv:1903.03096*, 2019.

[206] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[207] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[208] G. A. Miller, "WordNet: A lexical database for English," in *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[209] M. Huh, P. Agrawal, and A. A. Efros, "What makes imagenet good for transfer learning?," *arXiv preprint arXiv:1608.08614*, 2016.

[210] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision*

and Pattern Recognition, pp. 248–255, Ieee, 2009.

[211] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[212] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5828–5839, 2017.

[213] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind, "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10912–10922, 2021.

[214] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12179–12188, 2021.

[215] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European conference on computer vision*, pp. 402–419, Springer, 2020.

[216] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[217] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

[218] B. Roessle, J. T. Barron, B. Mildenhall, P. P. Srinivasan, and M. Nießner, "Dense depth priors for neural radiance fields from sparse input views," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12892–12901, 2022.

[219] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 4796–4803, IEEE, 2018.

[220] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with cnns: Depth completion and semantic segmentation," in *2018 International Conference on 3D Vision (3DV)*, pp. 52–60, IEEE, 2018.

[221] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?," in *International Conference on Machine Learning*, pp. 5389–5400, PMLR, 2019.

[222] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[223] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[224] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo, ""everyone wants to do the model work, not the data work": Data cascades in high-stakes ai," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2021.

[225] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna, "Data and its (dis) contents: A survey of dataset development and use in machine learning research," *arXiv preprint arXiv:2012.05345*, 2020.

[226] C. Kamann and C. Rother, "Benchmarking the robustness of semantic segmentation models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8828–8838, 2020.

[227] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," *arXiv preprint arXiv:1907.07484*, 2019.

[228] P. Chattopadhyay, J. Hoffman, R. Mottaghi, and A. Kembhavi, "Robustnav: Towards benchmarking robustness in embodied navigation," *arXiv preprint arXiv:2106.04531*, 2021.

[229] C. Yi, S. Yang, H. Li, Y.-p. Tan, and A. Kot, "Benchmarking the robustness of spatial-temporal models against corruptions," *arXiv preprint arXiv:2110.06513*, 2021.

[230] E. Mintun, A. Kirillov, and S. Xie, "On interaction between augmentations and corruptions in natural corruption robustness," *arXiv preprint arXiv:2102.11273*, 2021.

[231] J. Sun, Q. Zhang, B. Kailkhura, Z. Yu, C. Xiao, and Z. M. Mao, "Benchmarking robustness of 3d point cloud recognition against common corruptions," *arXiv preprint arXiv:2201.12296*, 2022.

[232] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "Objectnet: A large-scale bias-controlled dataset for pushing the

limits of object recognition models," 2019.

[233] G. Leclerc, H. Salman, A. Ilyas, S. Vemprala, L. Engstrom, V. Vineet, K. Xiao, P. Zhang, S. Santurkar, G. Yang, *et al.*, "3db: A framework for debugging computer vision models," *arXiv preprint arXiv:2106.03805*, 2021.

[234] J. Djolonga, J. Yung, M. Tschannen, R. Romijnders, L. Beyer, A. Kolesnikov, J. Puigcerver, M. Minderer, A. D'Amour, D. Moldovan, *et al.*, "On robustness and transferability of convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16458–16468, 2021.

[235] M. Naseer, K. Ranasinghe, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Intriguing properties of vision transformers," *arXiv preprint arXiv:2105.10497*, 2021.

[236] E. Rusak, L. Schott, R. Zimmermann, J. Bitterwolfb, O. Bringmann, M. Bethge, and W. Brendel, "Increasing the robustness of dnns against image corruptions by playing the game of noise," 2020.

[237] A. E. Orhan, "Robustness properties of facebook's resnext wsl models," *arXiv preprint arXiv:1907.07640*, 2019.

[238] R. Fattal, "Single image dehazing," *ACM transactions on graphics (TOG)*, vol. 27, no. 3, pp. 1–9, 2008.

[239] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, 2018.

[240] A. Von Bernuth, G. Volk, and O. Bringmann, "Simulating photo-realistic snow and fog on existing images for enhanced cnn training and evaluation," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 41–46, IEEE, 2019.

[241] X. Hu, C.-W. Fu, L. Zhu, and P.-A. Heng, "Depth-attentional features for single-image rain removal," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8022–8031, 2019.

[242] M. Tremblay, S. S. Halder, R. de Charette, and J.-F. Lalonde, "Rain rendering for evaluating and improving robustness to bad weather," *International Journal of Computer Vision*, vol. 129, no. 2, pp. 341–360, 2021.

[243] T. Brooks and J. T. Barron, "Learning to synthesize motion blur," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6840–6848, 2019.

[244] S. Niklaus, L. Mai, J. Yang, and F. Liu, "3d ken burns effect from a single image," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–15, 2019.

[245] M. Potmesil and I. Chakravarty, "A lens and aperture camera model for synthetic image generation," *ACM SIGGRAPH Computer Graphics*, vol. 15, no. 3, pp. 297–305, 1981.

[246] L. Wang, X. Shen, J. Zhang, O. Wang, Z. Lin, C.-Y. Hsieh, S. Kong, and H. Lu, "Deeplens: Shallow depth of field from a single image," *arXiv preprint arXiv:1810.08100*, 2018.

[247] N. Wadhwa, R. Garg, D. E. Jacobs, B. E. Feldman, N. Kanazawa, R. Carroll, Y. Movshovitz-Attias, J. T. Barron, Y. Pritch, and M. Levoy, "Synthetic depth-of-field with a single-camera mobile phone," *ACM Transactions on Graphics (ToG)*, vol. 37, no. 4, pp. 1–13, 2018.

[248] B. A. Barsky and T. J. Kosloff, "Algorithms for rendering depth of field effects in computer graphics," in *Proceedings of the 12th WSEAS international conference on Computers*, vol. 2008, World Scientific and Engineering Academy and Society (WSEAS), 2008.

[249] Z. Xu, K. Sunkavalli, S. Hadap, and R. Ramamoorthi, "Deep image-based relighting from optimal sparse samples," *ACM Transactions on Graphics (ToG)*, vol. 37, no. 4, pp. 1–13, 2018.

[250] M. E. Helou, R. Zhou, J. Barthas, and S. Süsstrunk, "Vidit: virtual image dataset for illumination transfer," *arXiv preprint arXiv:2005.05460*, 2020.

[251] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical poissonian-gaussian noise modeling and fitting for single-image raw-data," *IEEE Transactions on Image Processing*, vol. 17, no. 10, pp. 1737–1754, 2008.

[252] K. Wei, Y. Fu, J. Yang, and H. Huang, "A physics-based noise formation model for extreme low-light raw denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2758–2767, 2020.

[253] D. Kundur and D. Hatzinakos, "Blind image deconvolution," *IEEE signal processing magazine*, vol. 13, no. 3, pp. 43–64, 1996.

[254] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," in *ACM SIGGRAPH 2006 Papers*, pp. 787–794, 2006.

[255] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12,

pp. 3736–3745, 2006.

[256] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 2272–2279, IEEE, 2009.

[257] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[258] C. Chen, Q. Chen, J. Xu, and V. Koltun, "Learning to see in the dark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3291–3300, 2018.

[259] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3883–3891, 2017.

[260] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3929–3938, 2017.

[261] J. Rim, H. Lee, J. Won, and S. Cho, "Real-world blur dataset for learning and benchmarking deblurring algorithms," in *European Conference on Computer Vision*, pp. 184–201, Springer, 2020.

[262] S. Nah, S. Son, S. Lee, R. Timofte, and K. M. Lee, "Ntire 2021 challenge on image deblurring," in *CVPR Workshops*, pp. 149–165, June 2021.

[263] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1692–1700, 2018.

[264] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*, 2020.

[265] B. O. Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[266] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5664–5673, 2019.

[267] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[268] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *arXiv preprint arXiv:1907.01341*, 2019.

[269] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[270] W. Chen, S. Qian, D. Fan, N. Kojima, M. Hamilton, and J. Deng, "Oasis: A large-scale dataset for single image 3d in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 679–688, 2020.

[271] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019.

[272] A. Laugros, A. Caplier, and M. Ospici, "Using synthetic corruptions to measure robustness to natural distribution shifts," *arXiv preprint arXiv:2107.12052*, 2021.

[273] S. Sagawa, P. W. Koh, T. Lee, I. Gao, S. M. Xie, K. Shen, A. Kumar, W. Hu, M. Yasunaga, H. Marklund, *et al.*, "Extending the wilds benchmark for unsupervised adaptation," *arXiv preprint arXiv:2112.05090*, 2021.

[274] L. Smith and M. Gasser, "The development of embodied cognition: Six lessons from babies," *Artificial life*, vol. 11, no. 1-2, pp. 13–29, 2005.

[275] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[276] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18343–18362, 2022.

[277] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[278] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on*

computer vision and pattern recognition workshops, pp. 224–236, 2018.

[279] F. W. Campbell, "The depth of field of the human eye," *Optica Acta: International Journal of Optics*, vol. 4, no. 4, pp. 157–164, 1957.

B Curriculum Vitae

Email: teresa.ysq@gmail.com ◇ **Website**: aserety.github.io

EDUCATION

École Polytechnique Fédérale de Lausanne — Lausanne, CH
PhD in Computer Science — *Sept 2017 - Aug 2023*
Advisors: Amir Zamir, Pierre Dillenbourg

University of Cambridge — Cambridge, UK
MPhil in Machine Learning and Machine Intelligence — *Sept 2015 - Aug 2016*

EXPERIENCE

Teaching Assistant, EPFL — *Sept 2017 - Aug 2022*
Fall 2021: CS503 Visual Intelligence: Machines and Minds
Spring 2018, 2019, 2020: EE559 Deep Learning
Fall 2020: MATH111 Linear Algebra
Fall 2019: CS433 Machine Learning

PAPERS

Training Data Generation with Diffusion Models for Robustness — In Progress
T. Yeo, A. Atanov, A. Alekseev, H. Benoit, R. Ray, P. Esmaeil, A. Zamir

4M: Massively Multimodal Masked Modelling — NeurIPS'23 (Spotlight)
D. Mizrahi, R. Bachmann, O. F. Kar, T. Yeo, M. Gao, A. Dehghan, A. Zamir

Fast Adaptation of Neural Networks Using Test-Time Feedback — ICCV'23
T. Yeo, O. F. Kar, Z. Sodagar, A. Zamir

Task Discovery: Finding the Tasks that Neural Networks Generalize on — NeurIPS'22
A. Atanov, A. Filatov, T. Yeo, A. Sohmshetty, A. Zamir

3D Common Corruptions and Data Augmentation — CVPR'22 (Oral)
O. F. Kar, T. Yeo, A. Atanov, A. Zamir

Robustness via Cross-domain Ensembles — ICCV'21 (Oral)
T. Yeo, O. F. Kar*, A. Sax, A. Zamir*

Robust Learning Through Cross-Task Consistency — Arxiv'20
A. Zamir, A. Sax*, T. Yeo, O. F. Kar, N. Cheerla, R. Suri, Z. Cao, J. Malik, L. Guibas*

Iterative Classroom Teaching — AAAI'19 (Oral)
T. Yeo, P. Kamalaruban, A. Singla, A. Merchant, T. Asselborn, L. Faucon, P. Dillenbourg, V. Cevher