EPFL

Thèse n° 8988

Deep Learning Generalization with Limited and Noisy Labels

Présentée le 14 juillet 2023

Faculté informatique et communications
Laboratoire de la dynamique de l'information et des réseaux 2
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Mahsa FOROUZESH

Acceptée sur proposition du jury

Prof. A. Argyraki, présidente du jury
Prof. P. Thiran, directeur de thèse
Prof. M. Verleysen, rapporteur
Dr C. Zhang, rapporteur
Prof. S. Süsstrunk, rapporteuse

École
polytechnique
fédérale
de Lausanne

2023

Logic will get you from A to B.
Imagination will take you everywhere.
— Albert Einstein

To my family
Laleh, Nader, and Kianoush

Acknowledgements

I have been incredibly fortunate to have received tremendous support from family, friends, and colleagues throughout my Ph.D. journey. The following is a short summary of the generous contributions they have made towards the development of this thesis and my growth as a researcher.

First and foremost, I would like to express my heartfelt appreciation to my thesis advisor, Patrick Thiran. I am particularly thankful for his encouragement to explore diverse ideas without imposing any constraints, which allowed me the freedom to pursue unconventional ideas with confidence. Under his supervision, I have progressed to develop a more rigorous approach to problem-solving, and to analyze and articulate concepts in a more concrete manner. I am grateful for his patience and support, especially during the early stages of my research, as I was just beginning to navigate the world of academia.

I am sincerely grateful to Katerina Argyraki, Sabine Süsstrunk, Michel Verleysen, and Chiyuan Zhang for their participation as members of my jury committee. Throughout my doctoral journey, I have had the privilege of studying your exceptional contributions to the field. It is a great honor to engage in discussions regarding my research with such esteemed scholars.

I am thankful for the kind support I have received from the staff at the lab. Holly Cogliati-Bauereis has been instrumental in proofreading and enhancing my manuscripts, significantly improving my writing skills through her valuable feedback. Patricia Hjelt and Angela Devenoge have provided invaluable assistance with administrative tasks, consistently demonstrating kindness. I thank Marc-André Lüthi and Yves Lopes for their assistance with our IT infrastructure.

I would like to extend my gratitude to my academic collaborators for their valuable contributions. I am particularly grateful to Hanie Sedghi, who embraced my ideas wholeheartedly and dedicated extensive hours to discussing and refining them, playing a crucial role in shaping my research vision. Working alongside Hanie has been an absolute pleasure and a genuine source of inspiration. I would also like to express my thanks to Farnood Salehi, who co-authored my first research paper, and to Yasaman Haghighi and Loic Signer, who provided assistance in implementing certain aspects of my ideas.

I had the pleasure to work alongside a group of smart and nice people throughout my time in the INDY Lab. Thank you Alexandre, Arnout, Aswin, Brunella, Daniyar, Greg, Guillaume, Jalal, Lars, Lucas, Maximilien, Mladen, Saber, Sadegh, Saeed, Sébastien, Sepehr, Suryanarayana, Victor, and William!

Acknowledgements

To the wonderful friends I have been so lucky to have all around the world, I cannot express how grateful I am. Spending time with you has made the Ph.D. journey a possible reality. Thank you all for making life easier!

Lastly, I would like to thank my family, Laleh, Nader and Kianoush. Without their unwavering support, I would not have reached this point in my journey. Thank you for your unconditional love and constant presence. You are truly my rock. A special thanks goes to Ashkan, for being by my side every step of the way, no matter how tough it got.

Lausanne, June 26, 2023 M. F.

Abstract

Deep neural networks have become ubiquitous in today's technological landscape, finding their way in a vast array of applications. Deep supervised learning, which relies on large labeled datasets, has been particularly successful in areas such as image classification. However, the effectiveness of these networks heavily depends on the quality of the data they can use. In most practical applications, obtaining high-quality labeled data is expensive, time-consuming, and sometimes even impossible, making the available dataset limited both in size and in quality, as the labeled data may contain noise due to various reasons such as human labeling errors. The main objective of this thesis is to develop practical methods measuring the quality of the generalization of deep neural networks in such settings with limited and/or noisy labeled data. We propose novel methods and metrics for estimating generalization, overfitting, and memorization throughout training, which are easy to deploy, which eliminate the need for a high-quality validation/test set and which optimize the use of the available data.

First, we establish a connection between neural network output *sensitivity* and variance in the bias-variance decomposition of the loss function. Through extensive empirical results, we show that sensitivity is strongly correlated with the test loss and can serve as a promising tool for selecting neural network architectures. We find that sensitivity is particularly effective in identifying the benefits of certain architectural choices, such as convolutional layers. Additionally, we promote sensitivity as a zero-cost metric that can estimate model generalization even before training. Our results show that sensitivity effectively captures the benefits of specific regularization and initialization techniques, such as batch normalization and Xavier parameter initialization.

Second, we introduce *generalization penalty*, which measures how much a gradient step on one mini-batch negatively affects the performance on another mini-batch. From this, we derive a new metric called *gradient disparity* and propose it as an early stopping criterion for deep neural networks trained with mini-batch gradient descent. Our extensive empirical experiments demonstrate that gradient disparity is strongly correlated with the generalization error in state-of-the-art configurations. Moreover, it is very efficient to use because of its low computational tractability. Gradient disparity even outperforms traditional validation methods such as k-fold cross-validation when the available data is limited, because it can use all available samples for training. When the available data has noisy labels, it signals overfitting better than the validation data.

Third, we propose a metric called *susceptibility* to evaluate neural network robustness against label noise memorization. Susceptibility is easy to compute during training and

requires only unlabeled data, making it practical for real-world applications. We demonstrate its effectiveness in tracking memorization on various architectures and datasets, accurately distinguishing models that maintain low memorization on the training set. We also provide theoretical insights into the design of susceptibility as a metric for tracking memorization. We demonstrate its effectiveness through thorough experiments on several datasets with synthetic and real-world label noise. Susceptibility and the overall training accuracy complement each other and can distinguish models that maintain low memorization and generalize well on unseen clean data.

Finally, in the last part of this thesis, we tackle the challenge of filtering *noisy samples from hard-to-learn samples* in labeled datasets. To gain a better understanding of these two types of data, we design synthetic datasets with varying levels of hardness and noisiness. Through a systematic empirical study on these datasets, we study the ability of various metrics to distinguish hard samples from noisy samples. The results of this study lead us to propose a simple and effective method for filtering out noisy-labeled samples while retaining the hard samples. We demonstrate its effectiveness through empirical evaluations, which paves the way for future developments in this important yet under-explored topic.

Keywords deep neural networks, generalization, limited data, label noise, early stopping, memorization, deep supervised learning

Résumé

Les réseaux de neurones profonds sont devenus omniprésents dans le paysage technologique d'aujourd'hui, trouvant leur place dans une vaste gamme d'applications. L'apprentissage supervisé profond, qui repose sur de grands ensembles de données étiquetées, a particulièrement bien réussi dans des domaines tels que la classification d'images. Cependant, l'efficacité de ces réseaux dépend fortement de la qualité des données qu'ils peuvent utiliser. Dans la plupart des applications pratiques, l'obtention de données étiquetées de haute qualité est coûteuse, chronophage et parfois même impossible, ce qui rend l'ensemble de données disponible limité à la fois en taille et en qualité, car les données étiquetées peuvent contenir du bruit pour diverses raisons, telles que des erreurs d'étiquetage humaines. L'objectif principal de cette thèse est de développer des méthodes pratiques pour mesurer la qualité de la généralisation des réseaux de neurones profonds dans de telles configurations avec des données étiquetées en quantité limité et/ou corrompus par le bruit. Nous proposons de nouvelles méthodes et métriques pour estimer la généralisation, le surapprentissage et la mémorisation tout au long de l'entraînement, qui sont faciles à déployer, qui éliminent le besoin d'un ensemble de validation/test de haute qualité et qui optimisent l'utilisation des données disponibles.

Tout d'abord, nous établissons un lien entre la *sensibilité* de la sortie du réseau de neurones et la variance dans la décomposition biais-variance de la fonction de perte. À travers des résultats empiriques approfondis, nous montrons que la sensibilité est fortement corrélée avec la fonction de perte de test et peut servir d'outil prometteur pour sélectionner des architectures de réseau de neurones. Nous constatons que la sensibilité est particulièrement efficace pour identifier les avantages de certains choix architecturaux, tels que les couches de convolution. De plus, nous promouvons la sensibilité comme une métrique bon marché, qui peut estimer la généralisation du modèle même avant l'entraînement. Nos résultats montrent que la sensibilité capture efficacement les avantages de certaines techniques de régularisation et d'initialisation spécifiques, telles que la normalisation de lot et l'initialisation des paramètres Xavier.

Deuxièmement, nous introduisons la notion de *pénalité de généralisation*, qui quantifie dans quelle mesure une étape de gradient sur un mini-lot affecte négativement les performances sur un autre mini-lot. À partir de cela, nous dérivons une nouvelle métrique appelée *disparité de gradient* et la proposons comme critère d'arrêt précoce pour les réseaux de neurones profonds entraînés pour descente de gradient par mini-lot. Nos expériences empiriques approfondies démontrent que la disparité de gradient est fortement corrélée à l'erreur de généralisation dans les configurations, utilisées dans l'état de l'art actual . De plus, elle est

très efficace à utiliser en raison de sa faible capacité de calcul. La disparité de gradient surpasse même les méthodes de validation traditionnelles telles que la validation croisée "k-fold" lorsque les données disponibles sont limitées, car elle peut utiliser tous les échantillons disponibles pour l'entraînement. Lorsque les données disponibles ont des étiquettes corrompus par le bruit, elle signale mieux le surapprentissage que les données de validation.

Troisièmement, nous proposons une métrique appelée *susceptibilité* pour évaluer la robustesse des réseaux de neurones à la mémorisation de l'étiquette bruyante. La susceptibilité est facile à calculer pendant l'entraînement et ne nécessite que des données non étiquetées, ce qui la rend pratique pour les applications du monde réel. Nous démontrons son efficacité dans le suivi de la mémorisation sur diverses architectures et ensembles de données, en distinguant avec précision les modèles qui maintiennent une faible mémorisation sur l'ensemble d'entraînement. Nous fournissons également une intuition théorique sur la capacité la susceptibilité à mesurer la mémorisation. Nous démontrons son efficacité grâce à des expériences approfondies sur plusieurs ensembles de données avec des étiquettes bruyantes synthétiques et réelles. La susceptibilité et la précision globale de l'entraînement se complètent mutuellement et peuvent distinguer les modèles qui maintiennent une faible mémorisation et généralisent bien sur des données propres non vues.

Enfin, dans la dernière partie de cette thèse, nous abordons le défi de filtrer les *échantillons bruyants des échantillons difficiles à apprendre* dans les ensembles de données étiquetés. Pour mieux comprendre ces deux types de données, nous concevons des ensembles de données synthétiques avec des niveaux variables de difficulté et de bruit. À travers une étude empirique systématique sur ces ensembles de données, nous étudions la capacité de diverses métriques à distinguer les échantillons difficiles des échantillons bruyants. Les résultats de cette étude nous conduisent à proposer une méthode simple et efficace pour filtrer les échantillons étiquetés mais corrompus par le bruit tout en conservant les échantillons difficiles. Nous en démontrons l'efficacité par des évaluations empiriques, ce qui ouvre la voie à des développements futurs dans ce domaine important mais peu exploré.

Mots-clés réseaux de neurones profonds, généralisation, données limitées, bruit d'étiquette, arrêt précoce, mémorisation, apprentissage supervisé profond

Contents

Contents

1 Introduction

Deep learning, which involves training deep neural networks with a large number of parameters in order to learn complex data representations, has become a popular method in many applications. One widely used approach is supervised learning that has demonstrated success in tasks such as image classification, due in part to powerful computing resources and to the availability of large amounts of high-quality labeled data. However, obtaining such labeled data can be costly, time-consuming, and challenging in real-world settings. Therefore, it is crucial to develop robust methods that can address this limitation and to propose techniques that work even in the absence of clean labeled data. In this thesis, we address the issues of how, without a clean and accurate validation set, to gain insights into model generalization, learning, memorization, and data quality. We propose several methods as a replacement for the validation set; they enable us to extract information from the model trained on available data. In this chapter, we first introduce the background and state the research problem in Section 1.1. Then, in Section 1.2, we highlight the significance of our research and provide an outline of the thesis.

1.1 Background

1.1.1 Deep Supervised Learning

In order to carry out a certain task at hand, instead of directly programming computers, machine learning can be an effective means. The underlying reasons for this are that the task is too complicated to extract a well-defined program or that the data might be too large and complex to analyze, hence learning to detect meaningful underlying patterns is a promising use-case of machine learning [Shalev-Shwartz and Ben-David 2014]. Moreover, machine-learning tools are adaptive in nature hence provide the possibility to change over time from one user to the next.

Machine learning can be broadly categorized into supervised learning and unsupervised learning, depending on the *experience* they are permitted to have on a data set during the

learning process [Goodfellow et al. 2016]. A data set is a collection of some data samples/points. Supervised-learning algorithms experience a data set that contains input features $x \in \mathscr{X}$ that are associated with labels or targets $y \in \mathscr{Y}$. The data samples $z = (x, y) \sim \mathscr{D}$ in the data set come from an unknown distribution $\mathscr{D}$, and the learning task is then formulated as finding a predictor or a function $f : \mathscr{X} \to \mathscr{Y}$ such that the expected loss $\mathbb{E}_{z\sim\mathscr{D}}\left[l(f,z)\right]$ on the data samples is minimized, where l is the loss function, which for instance can be $\mathbb{1}\{f(x) \neq y\}$.

One of the first predictor models used in supervised learning is Perceptron: it was introduced by Rosenblatt [1958], the oldest artificial neural network still in use today. The introduction of Perceptron in 1958 is due to the early work of McCulloch and Pitts [1943] in 1943; this was the first work to begin exploring how a network of artificial neurons could replicate brain-like processes. The structure of Perceptron resembles human learning and was used to recognize simple patterns in images. Later, in 1984, the Boltzmann machine was introduced by Hinton et al. [1984]; it showed that neural networks could learn small-scale complex problems. This was then completed with the introduction of back-propagation in 1986 by Rumelhart et al. [1986], where the potential of neural networks to learn sophisticated tasks was demonstrated.

In principle, a two-layer neural network should be capable of performing most tasks. However, getting a back-propagation network to learn can be problematic for many practical applications [Stone 2019]. One possible solution is to increase the number of hidden units and to add more hidden layers, but this approach was initially deemed infeasible. It was not until the advent of faster computers, larger training datasets, and advancements in learning algorithms that the era of deep neural networks, also known as deep learning, came to fruition. In particular for the task of image classification, convolutional neural networks (CNNs) were introduced, with the LeNet architecture [LeCun et al. 1989] being the first to emerge in 1989. In 2012, the ImageNet computer vision challenge [Russakovsky et al. 2015] brought about a revolution, with the AlexNet architecture [Krizhevsky et al. 2017] dominating the competition. Following this, in 2013, many computer vision systems based on the AlexNet architecture showed significant progress [Simonyan et al. 2013; Sermanet et al. 2013].

In 2014, the modern era of deep learning continued with the introduction of a deep network called VGG [Simonyan and Zisserman 2014]. Subsequently, the ResNet architecture [He et al. 2016a] introduced the concept of residual connections in 2015, which once again led to a significant improvement in computer vision. In recent years, numerous successful architectures were introduced and consistently demonstrate exceptional performance. Most common modern state-of-the-art deep neural networks used in image classification share some traits, such as having a large number of layers or being *deep*, using the ReLU non-linearity [Fukushima 1975], pooling [LeCun et al. 1998], and utilizing dropout [Hinton et al. 2012] and batch normalization [Ioffe and Szegedy 2015] regularizations.

1.1.2 Generalization and Overfitting

As mentioned in the preceding section, the objective of a deep supervised learning algorithm is to minimize the expected loss $\mathbb{E}_{z\sim\mathscr{D}}\left[l(f_\theta,z)\right]$, where the function f_θ is a deep neural network with parameters θ. As the distribution $\mathscr{D}$ is typically unknown, the learner cannot evaluate or minimize $\mathbb{E}_{z\sim\mathscr{D}}\left[l(f_\theta,z)\right]$ directly. However, it can estimate it by using the training set S that is a collection of n input-output pairs, $S=\{z_1,\cdots,z_n\}$. Therefore, the learning task is typically approached by solving:

$$\min_\theta \hat{\mathbb{E}}_S\left[l(f_\theta,z)\right]=\min_\theta L_S(f_\theta)=\frac{1}{n}\sum_{i=1}^{n} l(f_\theta,z_i), \tag{1.1}$$

where $\hat{\mathbb{E}}_S\left[l(f_\theta,z)\right]$ (also denoted by $L_S(f_\theta)$) is known as the training loss. Minimizing the above equation is known as the empirical risk minimization (ERM) problem. To minimize the training loss, the algorithm typically starts with a randomly initialized parameter vector θ_{init} [Glorot and Bengio 2010; He et al. 2015; Hanin and Rolnick 2018], then uses mini-batch stochastic gradient descent (SGD) or one of its variants [Ruder 2016]. Each step of the mini-batch gradient descent at time/iteration t is as follows:

$$\theta_{t+1}=\theta_t-\gamma\nabla_\theta L_{S_{\text{b}}^t}\left(f_\theta\right), \tag{1.2}$$

where S_{b}^t is the mini-batch selected at iteration t and γ is the learning rate.

Minimizing the loss in Eq. (1.1) is a necessary but insufficient condition for achieving low expected loss $\mathbb{E}_{z\sim\mathscr{D}}\left[l(f_\theta,z)\right]$. A model that, by memorizing to output the right label for the data, fits the training set S too closely, can obtain zero training loss while having a high expected loss. We are therefore interested in controlling the difference $\mathbb{E}_{z\sim\mathscr{D}}\left[l(f_\theta,z)\right]-\hat{\mathbb{E}}_S\left[l(f_\theta,z)\right]$ known as the *generalization error/gap*. This quantity reflects the difference between memorizing and learning and is a measure of the model's ability to generalize beyond the training set. A model with a high generalization gap is said to experience *overfitting*.

To estimate the generalization gap (particularly the expected loss $\mathbb{E}_{z\sim\mathscr{D}}\left[l(f_\theta,z)\right]$), a common practice is to use an independent held-out validation/test[1] set $S_v=\{z_{n+1},\cdots,z_{n+n_v}\}$. The validation set is a collection of n_v input-output pairs that are randomly sampled from the same distribution as the training set. The model is trained on the training set and evaluated on the validation set, thus enabling us to estimate its expected loss on new data. This estimate is reliable, as long as the validation set is truly independent of the training set, the number of samples n_v is large enough, and the samples of S_v are sampled from the true data distribution $\mathscr{D}$.

Controlling the generalization gap is crucial for achieving good performance on real-world tasks. If the model is overfitting, it will perform well on the training set but poorly on new data.

[1] In this thesis, we use validation and test sets interchangeably. In practice, they serve slightly different purposes, though in the context studied in this thesis, they can be referred to as the same.

Whereas, if it is underfitting, it will perform poorly on both. By estimating the generalization gap by using a validation set, we can select the best model that balances between underfitting and overfitting and that achieves good performance on unseen data. This process is called *model selection* and is an essential part of the machine-learning workflow.

1.1.3 Data-Collection Challenges

As mentioned in the preceding section, a standard practice for estimating the generalization gap is to employ an independently held validation set S_v. Before training begins, the given dataset is partitioned into two subsets, namely the training set S and the validation set S_v. Care must be taken to balance the sample ratio in S and S_v. If the number of samples in S_v (denoted by n_v) is too large, the machine-learning model will exhibit high variance during training. Conversely, if n_v is too small, the generalization gap estimate and model evaluation will have a larger variance. In practice, a common ratio is to set $n : n_v$ as $80 : 20$. We will now explore the cost of the additional 20% of labeled data (or the extra n_v input-output pairs) present in the validation set S_v: (i) cost in terms of performance, (ii) cost in terms of data collection.

Cost in Terms of Performance Because of finite resources, this validation set must be a subset of the overall dataset available, which necessitates removing samples from the training set S. These removed samples have an effect on the performance of the trained model. For example, consider the MRNet dataset [Bien et al. 2018]: an image classification dataset for detecting knee injuries. The dataset contains 1370 magnetic resonance imaging (MRI) exams for studying the presence of abnormality, anterior cruciate ligament (ACL) tears and meniscal tears. The labeled data in the MRNet dataset is very limited. From these 1370 MRI scans, 120 of them are kept private and are not accessible. Setting aside another 120 samples for local evaluation results in 1130 samples remaining. If we were to use 20% of this available dataset for the validation set S_v and 80% for the training set S, then the size of the training set reduces to 904. As the data in MRNet dataset is imbalanced (more negatives than positives), instead of the classification accuracy, the evaluation metric that is used is the area under the receiver operating characteristics curve (AUC). The receiver operating characteristics curve is created by plotting the true positive rate against the false positive rate for a binary classifier system at various discrimination threshold values. One interpretation of AUC is to view it as the probability that the model ranks a random positive example more highly than a random negative example. In Fig. 1.1 we show the difference in the test area under curve (AUC) scores for a model trained on all the available data (1130), and one trained on a smaller dataset due to the requirement of having a validation set (the one with size 904).

From Fig. 1.1, we can see that in order to improve the test AUC score from 0.89 to 0.91 (values at the end of the training) in this setting, we would need 226 more labeled samples. This example illustrates the importance of each sample in the training set and the criticality of losing samples when we have to separate a held-out subset of the dataset for validation.

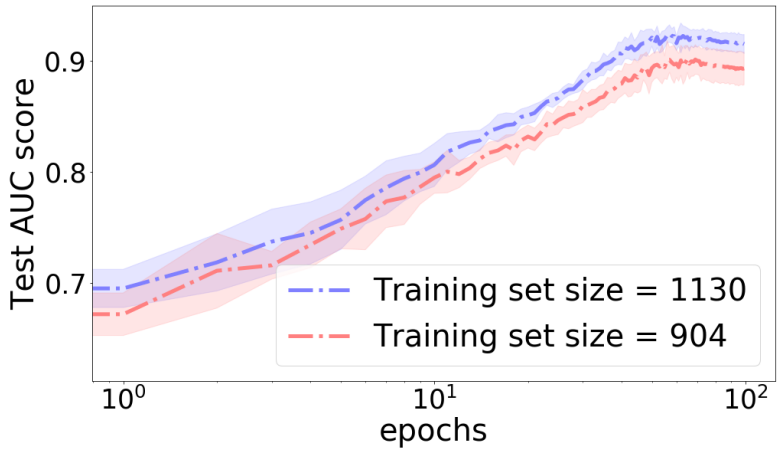


Figure 1.1: Two area under curve (AUC) scores for models trained on MRNet [Bien et al. 2018] training sets with different sizes. The scores belong to the detection of anterior cruciate ligament (ACL) tears; and as the dataset is imbalanced, instead of the test accuracy, AUC is reported. We observe the rather significant improvement in the generalization performance resulting from adding more training samples to the training set. In this dataset, an improvement in the generalization performance results in an accurate prediction in detecting ACL tears for a few new patients and hence facilitates their diagnostic process.

Utilizing every sample in our training can significantly improve model performance, especially as improving performance can be challenging. Therefore, we must make the most of our samples, because each data point we add to the training set is valuable. The importance of labeled data is crucial, as we can see from the above example where we needed 25% additional data to improve performance by 2.2%. However, acquiring additional labeled data can be challenging. In the following, we will discuss some of the challenges associated with collecting extra labeled data.

Cost in Terms of Data Collection In this thesis, our primary focus is on the task of image classification. Data collection for this task involves the gathering of images and labeling them accordingly. However, both image and label collection pose their own unique challenges, with the latter being particularly complex.

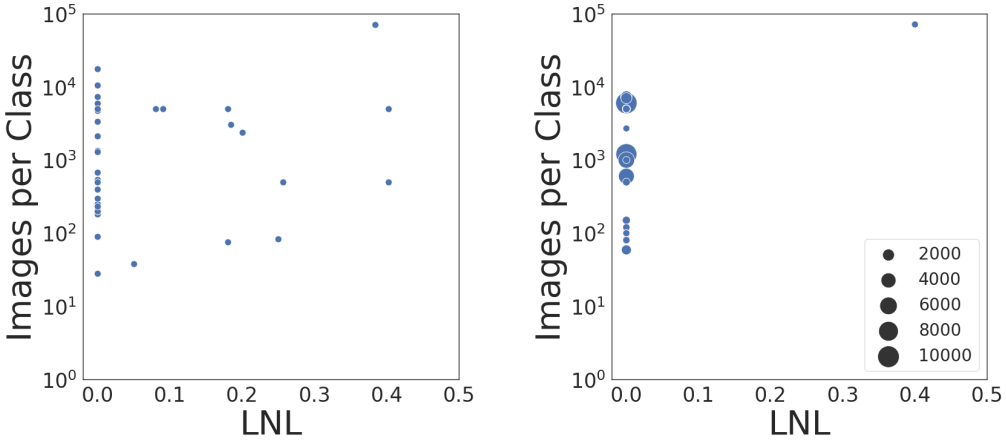
Collecting image data can be a costly process because of the requirement of expensive cameras or the need to hire additional workforce. For instance, when capturing images of wildlife, obtaining a picture of a particular animal can take several days or even weeks, in addition to the installment of cameras in place and workforce to check them [Van Horn and Perona 2017; Beery et al. 2020]. Additionally, there are ethical and legal constraints associated with image collection [Kaissis et al. 2020]. For instance, when facial recognition is involved, collecting biometric data can be challenging; and if not done correctly, it can lead to lawsuits. It is also challenging to determine the amount of data required in advance to ensure a smooth training process. That necessitates the collection of as many images as possible, which is both time-consuming and expensive.

Once the image data is collected, the next step is to label them. Data labeling is one of the most expensive tasks in machine-learning algorithms. There are two main approaches to labeling data: outsourcing it or conducting in-house manual data labeling. Organizations that outsource data labeling generally have to choose between paying for data labeling services per hour or per task. Paying per task is more cost-effective, but it can incentivize rushed work as labelers try to complete more tasks within a given time frame. Either option requires a significant budget and can be time consuming, particularly if multiple labels are required per sample to ensure high quality. In-house manual data-labeling teams, even small ones, can also be expensive due to the time and training required to attain true expertise.

Deep-learning algorithms have shown remarkable performance in detecting and classifying patterns, making them valuable also in applications where labeling is a sensitive and complex task. Although annotating objects in the real world, such as a cat or a dog, is relatively straightforward, annotating samples that exceed simple common knowledge can be challenging. For instance, annotating medical data is an expensive, tedious, and time-consuming process that requires extensive input from experts, specifically because of the sensitivity of the domain with respect to the potential impact on people's health. Furthermore, annotation might not always be possible, in particular in the case of rare medical conditions [Razzak et al. 2018; Hesamian et al. 2019].

To reduce the cost and time associated with label collection, various alternatives have emerged. One of the most popular alternatives is crowdsourcing. Crowdsourcing services, such as Amazon Mechanical Turk, enable the distribution of small labeling tasks to a large number of workers, thus making it a cost-effective option. However, it has its drawbacks. Manually verifying the quality of the submitted results can be challenging, leading to issues with malicious workers who may intentionally submit low-quality labels, resulting in label noise in the collected dataset [Ipeirotis et al. 2010]. Label noise can also result from the inherent difficulty in annotating certain types of images. In some cases, image labels can be obtained from accompanying text on the web, which can lead to inconsistencies and inaccuracies in labeling. These and other factors can contribute to label noise in real-world labeled datasets [Frénay and Verleysen 2013; Algan and Ulusoy 2020; Cordeiro and Carneiro 2020; Karimi et al. 2020; Xiao et al. 2015].

As highlighted by Frénay and Verleysen [2013], label noise can significantly impact the accuracy estimation of a model when the test samples are also corrupted with noise [Lam and Stork 2003]. This can also lead to biased model comparison. For instance, a spam filter with a true error rate of 0.5% can be estimated to have an error rate between 5.5% and 6.5% when evaluated using labels with an error rate of 6.0% [Cormack and Kolcz 2009]. Therefore, it is crucial to have a high-quality validation set S_v to assess the model's performance. However, as we have discussed in this section, creating such a validation set is a difficult and costly task.



(a) Datasets

(b) Number of Papers

Figure 1.2: (a) Number of images per class (a token for dataset size) and the label noise level (LNL) $\in [0,1]$ of datasets introduced in recent image classification survey papers: Hyperspectral image classification [Li et al. 2019b; Jia et al. 2021], Cell image classification [Shifat-E-Rabbi et al. 2020], Diabetic Retinopathy image classification [Kandel and Castelli 2020], COVID-19 [Aggarwal et al. 2022], Medical images [Karimi et al. 2020], general survey [Song et al. 2022]. (b) The number of images per class and the label noise level (LNL) of datasets in https://paperswithcode.com/ website under the filtration of image classification. Number of papers for each dataset is indicated with the size of the marker. We only show those datasets with more than 200 papers.

1.2 Goals and Contributions of the Thesis

In deep supervised learning (introduced in Section 1.1.1), it is important to estimate generalization (described in Section 1.1.2), but doing so using a validation set poses challenges (highlighted in Section 1.1.3). One can wonder how relevant these challenges are and why one should care about data collection challenges. Indeed, in presence of unlimited resources to collect and label data, one can afford to set aside a validation set and obtain information about the model's generalization, learning, memorization and even data quality. However, this is not always feasible, especially given the limited sizes and significant label noise levels that affect datasets in practice.

To shed some light on this, we look at datasets recalled in recent survey/review papers on image classification and find that datasets can have as little as 28 images per class, with label noise levels (LNL) varying from 0% to around 50%. A summary of these statistics is given in Fig. 1.2 (a). It is worth noting that being in the top left part of this figure is costly, and even with large budgets, it may not be possible to achieve this level of quality in some settings, such as datasets for COVID. This indicates that datasets with label noise and limited data are a common reality. Interestingly, when we look at the number of papers with experiments on each dataset, shown in Fig. 1.2 (b), we find that the image classification research community mostly focuses on datasets with a large number of samples per class and low label noise levels,

which are in the top left part of the spectrum. In particular, these machine-learning algorithms are often bench-marked on such large and clean datasets. This discrepancy between the research community's focus and the practical demand for datasets highlights the need to study image classification methods on datasets that are not in the top left part of the spectrum.

The motivation behind this thesis is to bridge the gap highlighted in Fig. 1.2. In this thesis, we focus on datasets that have limited labeled data and high label noise levels, which are in the bottom right of the spectrum of datasets in Fig. 1.2. These datasets are often overlooked by researchers, but they are of great interest to those who do not have unlimited resources and encounter label noise. The solutions developed in this thesis share a common feature: they are all *practical* tools that individuals or companies can use in their datasets, particularly when they have *limited* and/or *noisy* labeled datasets and value each labeled sample highly. To keep the problem practically manageable, we propose computationally efficient methods that do not add computational overhead.

In Chapter 2, we investigate the question *When and why certain models generalize to unseen data?* We do so by establishing a link between the test loss and the neural network output sensitivity to small input perturbations, i.e., the effect of adding an external injected noise ε_x to inputs x on f_θ. This link is particularly interesting as the test loss requires a test set, whereas sensitivity does not require any labeled data. This link is due to the connection between sensitivity and the variance term in the bias-variance decomposition of the loss function. Our results suggest that the bias term should be negligible in order to predict test loss by using sensitivity. We demonstrate the connection between sensitivity and loss for a broad range of settings, beyond fully connected networks and image-classification tasks. Furthermore, in Section 2.1.4, we compute an expression for the test loss as a function of sensitivity. We empirically observe a rather strong match between this expression and the experimental results on state-of-the-art models. Our results in this chapter fully explore the extent to which this connection is accurate, as this plays an important role for the deep learning community. Because, as explored by Jiang et al. [2019], measuring generalization is one of the most important tasks in machine learning. Measuring generalization is also the objective of the recent NeurIPS competition [Jiang et al. 2020]. It is, however, very important to explore the exact settings, advantages, and limitations of each particular metric, to know which metric to choose in which practical scenario; but this is beyond the scope of the competition. Nevertheless, this is precisely what we do in Chapter 2 particularly for the sensitivity metric. In the competition, the Jacobian of the neural network output with respect to its inputs, which is very similar to our sensitivity metric, is presented as one of the baselines. The Jacobian, as a generalization measure, was originally proposed in [Novak et al. 2018]. A practical motivation for using sensitivity in real-world applications is its computational tractability as it avoids backward pass or matrix multiplications contrary to the Jacobian. The wall-clock time drops from 1.16 seconds to 0.08 seconds when we shift from using the Jacobian to using sensitivity. But the main reason for using sensitivity is that it is computed *before* and not *after* the softmax layer, contrary to the Jacobian. In Chapter 2, we also provide an alternative explanation for the benefits of certain design choices. For instance, sensitivity is particularly effective in

identifying the benefits of certain architectural choices, such as convolutional layers, and it decreases when we add depth instead of width. We also demonstrate that sensitivity effectively captures the benefits of specific regularization and initialization techniques, such as batch normalization and Xavier parameter initialization.

In Chapter 3, we address the question *When should we stop training in the absence of sufficient high-quality labeled data to form a held-out validation set?* Specifically, we focus on settings with limited and/or noisy labeled data. We assume that the optimization process employs a variant of mini-batch gradient descent (Eq. (1.2)). We first introduce a new concept called *generalization penalty*. It measures the difference between the loss on a single batch, if another batch is selected for the optimization step, and its loss value, if the batch itself is selected for the optimization step. The expected penalty quantifies how much, in an iteration, a model updated on one mini-batch is able to generalize on average to another mini-batch from the data set. Under the PAC-Bayesian framework [McAllester 1999a;b; 2003], we establish a probabilistic upper bound on the generalization penalty. This upper bound is then simplified to a factor of the ℓ_2-norm distance between the gradient vectors of the two mini-batches. We call this ℓ_2 distance *gradient disparity*. We first observe that the value of gradient disparity computed between two batches from the training data is highly correlated to gradient disparity between one batch from the training set and one batch from the validation set. This suggests that, even if we compute gradient disparity entirely on the training set, it still holds information about the held-out validation set. We propose gradient disparity as a promising early stopping criterion and, through extensive experiments, compare it to k-fold cross-validation [Stone 1974]. We choose k-fold cross-validation as the main baseline because it, similar to our approach, takes all samples in the available dataset and uses all of them both for training and validation. Our empirical results on settings with limited datasets demonstrate a significant performance improvement when using gradient disparity as an early stopping criterion compared to k-fold cross-validation. This mainly occurs because gradient disparity is computed entirely on the training set hence enables us to use all the available data only for training. In contrast, with k-fold cross-validation, $1/k$-th of the data is still set aside and not used for training in each of the k folds. Furthermore, our empirical results on settings with noisy-labeled data also demonstrate a significant performance improvement when using gradient disparity as an early stopping criterion compared to k-fold cross-validation. This is primarily due to the fact that gradient disparity is, when the data contains noisy labels, a more accurate predictor of overfitting than a validation set. Overall, we observe that gradient disparity is a very robust and effective early stopping criterion that outperforms other early stopping measures. It has low sensitivity to the early stopping threshold and has a high correlation to the value of the test error. Gradient disparity is a useful tool in practice, particularly in settings where the available data is limited and of low quality, such as in medical datasets (e.g., the MRNet dataset [Bien et al. 2018]).

In Chapter 4, we address the issue *How do we track memorization of models when the available training set contains noisy labels?* A recent study by Garg et al. [2021a] shows theoretically that for models trained on a mixture of clean and noisy data, a low accuracy on the noisy

subset of the training set and a high accuracy on the clean subset of the training set guarantee a low generalization error. However, computing these accuracies requires ground-truth labels that are not feasible in practice. In this chapter, we propose a practical approach to track, without ground-truth label access or access to a clean validation set, the accuracy on the noisy subset of the training set. Our approach is based on our theoretical and empirical observation that models with a high test accuracy are resistant to memorizing a randomly labeled held-out set. Building on this result, we propose an easy-to-compute metric called *susceptibility* to noisy labels; it is the difference in the objective function of a single mini-batch from a held-out randomly-labeled set, before and after taking an optimization step on it. During training, the larger the difference is, the more susceptible the model is to the noisy labels in the mini-batch. Our empirical results demonstrate a strong correlation between susceptibility and accuracy on the noisy subset of the training set. Unlike the training accuracy on the noisy subset, susceptibility requires no ground-truth label access, is computed using only unlabeled data, and is computationally very cheap. Moreover, it outperforms related work in tracking label-noise memorization. We observe that models that are trainable and resistant to memorization, i.e., having a high training accuracy and a low susceptibility, have high test accuracies. We exploit this observation to propose a model-selection method in the presence of noisy labels. Our approach is shown to be effective in a variety of experimental settings and datasets with label noise, ranging from selecting the "best" models from "good" models for easy datasets, such as Animal-10N [2], to selecting "good" models from "bad" models for more complex datasets, such as Clothing-1M [Xiao et al. 2015]. Overall, susceptibility is a simple but surprisingly effective approach to tracking memorization by using only a single mini-batch of unlabeled data. It is a promising technique for practical model selection in presence of label noise. Our proposed model-selection method works well for real-world label noise, and our results are persistent across various datasets, architectures, hyper-parameters, label-noise levels, and label-noise types.

Lastly, in Chapter 5, we shift our focus from dealing with limited or noisy data to improving dataset quality. Whereas, in the first three chapters, we emphasize the importance of including every single sample in the training set, in this chapter, we explore another approach of selecting high-quality samples. Similarly to Chapters 3 and 4, the settings studied in this chapter involve datasets with label noise. However, the specific topic of interest is the presence of hard-to-learn samples, in addition to noisy-labaled samples. The challenge here is that these two types of data share many characteristics, but should be treated in opposite ways: noisy-labeled samples should be removed or given less emphasis during training, whereas hard-to-learn samples should be kept or given more emphasis. Therefore, it is crucial to study these two types of data and to distinguish between them. However, one challenge in this regard is that hard-to-learn samples are difficult to quantify. In Chapter 5, we propose a systematic approach to studying the similarities and differences between hard and noisy samples by introducing a novel framework for creating synthetically difficult samples. We take three different approaches: imbalanced-ness, diversification, and closeness to the decision boundary, and we use them

[2] https://dm.kaist.ac.kr/datasets/animal-10n/

to make samples in a baseline dataset more or less difficult. This way, we are able to assign a hardness score to each individual sample and study them. We introduce and evaluate various metrics for detecting noisy labels and demonstrate that our proposed static centroid distance (SCD) metric, inspired by the metric proposed in [Zhang et al. 2022], is the most effective in distinguishing between hard and noisy-labeled samples. We propose and evaluate different methods for data cleansing and sample selection and show that a two-dimensional Gaussian mixture model, which uses the accuracy over the training and SCD as features, is the most effective at filtering out noisy samples while retaining hard ones. We demonstrate the superiority of our approach in both synthetic datasets and real-world datasets with label noise.

Overall, a summary of our contributions in a machine learning framework, including an exact placement of each chapter is given in Fig. 1.3.

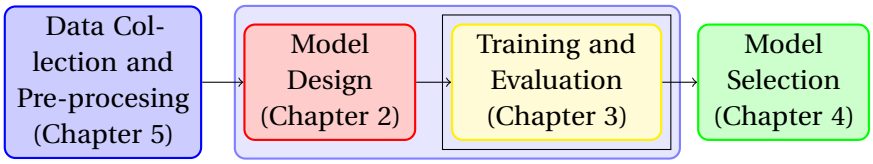


Figure 1.3: Schematic overview of a typical machine-learning framework starting with data collection and model design, followed by training and ending with model selection. In Chapter 5, we propose a label noise detection approach which is compatible with datasets with hard samples; which is useful in the data pre-processing stage. In Chapter 2, we show that sensitivity recovers model design choices that result in high generalization performance; which is useful in the model design stage. In Chapter 3, we propose an early stopping criterion which could be incorporated into training; which is useful in the training and evaluation stages. Finally, in Chapter 4, we propose a model-selection method which recovers models with a low memorization of noisy labels; which is useful in the model-selection stage. It is important to note that the metrics and methods proposed in each chapter of this thesis can also be applied in other stages of a machine-learning project. However, the figure presented here serves as an example of how different chapters can be incorporated into a machine-learning setting that is particularly suitable for working with limited and noisy-labeled datasets.

2 Neural Network Output Sensitivity

2.1 Generalization Comparison of Deep Neural Networks

Although recent works have brought some insights into the performance improvement of techniques used in state-of-the-art deep-learning models, more work is needed to understand their generalization properties. In this chapter[1], we shed light on this matter by linking the loss function to the output's sensitivity to its input. We find a rather strong empirical relation between the output sensitivity and the variance in the bias-variance decomposition of the loss function, which hints on using sensitivity as a metric for comparing the generalization performance of networks, without requiring labeled data. We find that sensitivity is decreased by applying popular methods which improve the generalization performance of the model, such as (1) using a deep network rather than a wide one, (2) adding convolutional layers to baseline classifiers instead of adding fully-connected layers, (3) using batch normalization, dropout and max-pooling, and (4) applying parameter initialization techniques.

2.1.1 Introduction

In machine-learning tasks, the main challenge a network designer faces is to find a model that learns the training data and that is able to predict the output of unseen data with high accuracy. The first part is quite easily achievable by current over-parameterized deep neural networks, but the second part, referred to as generalization, demands careful expert hand-tuning [LeCun et al. 2015; Goodfellow et al. 2016]. Modern convolutional neural network (CNN) architectures that achieve state-of-the-art results in computer-vision tasks, such as ResNet [He et al. 2016a] and VGG [Simonyan and Zisserman 2014], attain high-generalization performance. Part of their success is due to recent advances in hardware and the availability of large amounts of data, but their generalization performance remains unequal. Therefore, knowing when and why some models generalize, still remain open questions to a large extent [Neyshabur et al. 2017a].

[1] This chapter is based on [Forouzesh et al. 2021].

In this chapter, by investigating the link between sensitivity and generalization, we get one step closer to understanding the generalization properties of deep neural networks. Our findings suggest a relation between the sensitivity metric, a measure of uncertainty of the output with respect to input perturbations, and the variance term in the bias-variance decomposition of the test loss. This relation gives insight in the link between sensitivity and loss when the bias is small, not only for classification tasks, but also for regression tasks.

Leveraging this relation, we can use the sensitivity metric to examine which network is more prone to overfitting. Our numerical results suggest sensitivity as an appealing metric that captures the generalization improvements brought by a large class of architectures and techniques used in state-of-the-art models. In summary, we make the following contributions:

- We provide an approximate relation between sensitivity and test loss, via the relation between sensitivity and variance in the bias-variance decomposition of the loss. Our empirical results on state-of-the-art convolutional neural networks suggest a surprisingly strong match between experimental results and this (rather crude) approximation.
- We propose sensitivity as a promising architecture-selection metric and show that sensitivity, similarly to the test loss, promotes certain architectures compared to others. We in particular study the addition of convolutional layers versus fully-connected ones, and depth versus width. Sensitivity can potentially be used as a neural architecture search (NAS) tool, a priori (before training), to automate the architecture-design process.
- We provide an alternative explanation for the success of batch normalization in terms of sensitivity. We further give a new viewpoint on the performance improvement of dropout and max-pooling, as networks with these methods have a lower sensitivity alongside a lower test loss. We show that sensitivity retrieves the effectiveness of He and Xavier parameter initialization techniques.

2.1.2 Related Work

To the best of our knowledge, Dimopoulos et al. [1995] was the first study to suggest a possible relation between sensitivity and generalization in multi-layer perceptrons, where the numerical results were limited to synthetic data. Recently, Sokolić et al. [2017] suggested bounding the generalization error of deep neural networks with the spectral norm of the input-output Jacobian matrix, a measure of output sensitivity to its inputs. Reference [Novak et al. 2018] empirically compares sensitivity, measured by the norm of the Jacobian of the *output* of the softmax layer, and the generalization gap for fully-connected neural networks in image-classification tasks, leaving more complex architectures and other machine learning tasks as future work. Our empirical results presented in Section 2.1.4, together with the computations in Section 2.1.4, suggest that sensitivity *before* the softmax layer is related to the test loss, and that computing the sensitivity before (as in our work) or after (as in [Novak et al. 2018]) the softmax layer makes a strong difference (see e.g., Fig. 2.13). In our work, we elaborate on

the relation between sensitivity and loss for a wide range of settings, beyond fully-connected networks and image-classification tasks. We also show a rather strong match between the expression computed in Section 2.1.4 and the experimental results on state-of-the-art models.

To avoid overfitting in deep-learning architectures, regularization techniques are applied, such as weight decay, early stopping, dropout [Srivastava et al. 2014], and batch normalization (BN) [Ioffe and Szegedy 2015]. A popular explanation for the improved generalization of dropout is that it combines exponentially many networks to prevent overfitting [Srivastava et al. 2014]. Reference [Ioffe and Szegedy 2015] argues that the reason for the success of BN is that it addresses the internal-covariant-shift phenomenon. However, Santurkar et al. [2018] argue against this belief and explains that the success of BN is due to its ability to make the optimization landscape smoother. In this chapter, we look at the success of dropout and BN from another perspective: These methods decrease the output sensitivity to random input perturbations in a same manner as they decrease the test loss, resulting in better generalization performance.

Designing neural network architectures is one of the main challenges in machine-learning tasks. One major line of work in this regard compares deep and shallow networks [Bengio and Delalleau 2011; Mhaskar et al. 2017; Wu et al. 2019; Ba and Caruana 2014; Montufar et al. 2014; Simonyan and Zisserman 2014]. It is shown in [Telgarsky 2016] that to approximate a deep network, a shallow network requires an exponentially larger number of units per layer. After finding a satisfactory architecture, the trainability of the network needs to be carefully assessed. To avoid exploding or vanishing gradients, [Glorot and Bengio 2010] and [He et al. 2015] introduce parameter initialization techniques that are widely used in current frameworks. By linking sensitivity and generalization, we present a new viewpoint on understanding the success of current state-of-the-art architectures and initialization techniques.

Previous theoretical studies attempting to solve the mystery of generalization include generalization error (GE) bounds that use complexity measures such as VC-dimension and Rademacher complexities [Mohri et al. 2018]. Encouraged by the ability of neural networks to fit an entire randomly labeled dataset [Zhang et al. 2016a], studies on data-dependent GE bounds have recently emerged [Kawaguchi et al. 2017; Bartlett et al. 2017; Arora et al. 2018]. Computing a practical non-vacuous GE bound that completely captures the generalization properties of deep neural networks is still an evolving area of research [Dziugaite and Roy 2017; Neyshabur et al. 2017a; Nagarajan and Kolter 2019]. In this chapter, we do not study GE bounds. We propose sensitivity as a practical proxy for generalization in a large number of settings.

There has been research on sensitivity analysis in neural networks with sigmoid and tanh activation functions [Dimopoulos et al. 1995; Fu and Chen 1993; Zeng and Yeung 2001]. Reference [Yang et al. 2013] introduces a sensitivity-based ensemble approach which selects individual networks with diverse sensitivity values from a pool of trained networks. Reference [Piche 1995] performs a sensitivity analysis in neural networks to determine the required

precision of the weight updates in each iteration. In this work, we extend these results to networks with ReLU non-linearity with a different goal, which is to study the relation between sensitivity and generalization in state-of-the-art deep neural networks. Moreover, we provide a link between sensitivity and the variance in the bias-variance decomposition of the loss function.

There have been recent attempts to predict the test loss for supervised-learning tasks [Novak et al. 2018; Jiang et al. 2018b; Wang et al. 2018]. Reference [Chatterji et al. 2019] studies the module criticality, which is a weighted average over the distance of the network parameter vectors from their initial values. Although there seems to be a positive correlation between module criticality and generalization among different architectures, the correlation becomes negative when comparing the same architecture with different widths (as reported in Table 4 in [Chatterji et al. 2019]). Reference [Philipp and Carbonell 2018] introduces the so-called non-linearity coefficient (NLC) as a gradient-based complexity measure of the neural network, which is empirically shown to be a predictor of the test error for fully-connected neural networks. According to our results on both fully-connected and convolutional neural networks, sensitivity predicts the test loss, even before the networks are trained, which suggests sensitivity as a computationally inexpensive architecture-selection metric. Among the mentioned metrics, the Jacobian norm, studied in [Novak et al. 2018], does not require the computation of the parameter gradients nor the storage of large parameter vectors, as our metric, and therefore we compare it to sensitivity in Table 2.1.

Chapter Outline. We formally define loss and sensitivity metrics in Section 2.1.3. In Section 2.1.4, we state the main findings of the chapter and present the numerical and analytical results supporting them. Later in Section 2.1.5, we propose a possible proxy for generalization properties of certain structures and certain methods and present the empirical results for a regression task with the Boston housing dataset. Finally in Section 2.1.6, we further discuss the observations followed up by a conclusion. The empirical results for image-classification tasks presented in the main part of the chapter are on the CIFAR-10 dataset and the empirical results for MNIST and CIFAR-100 datasets are deferred to Section 2.F.

2.1.3 Preliminaries

Consider a supervised-learning task, where the model predicts a ground-truth output $y \in \mathscr{Y} := \mathbb{R}^K$ for an input $x \in \mathscr{X} := \mathbb{R}^D$. The predictor $f_\theta : \mathscr{X} \to \mathscr{Y}$ is a deep neural network parameterized by the parameter vector θ that is learned on the training dataset S by using the stochastic learning algorithm $\mathscr{A}$. The training dataset S and the testing (validation) dataset S_v consist of i.i.d. samples drawn from the same data distribution $\mathscr{D}$. With some abuse of notation, we use $\sim$ when the samples are uniformly drawn from a set of samples or from a probability distribution.

Loss

Our main focus is a classification task where the loss function is the cross-entropy criterion. The average test loss can be defined as

$$L = \mathbb{E}_{\theta^*}[L_{\theta^*}] = \mathbb{E}_{\theta^*}\left[\mathbb{E}_{(x,y)\sim S_v}\left[-\sum_{k=1}^{K} y^k \log f_{\theta^*}^k(x)\right]\right], \tag{2.1}$$

where θ^* is the random[2] parameter vector found by $\mathscr{A}$, which minimizes the training loss defined on S; K is the number of classes and $f_{\theta^*}^k$ is the k-th entry of the vector f_{θ^*}, which is the output of the softmax layer, i.e., $f_{\theta^*}(x) = \text{softmax}(F_{\theta^*}(x))$, where $F_{\theta^*}(x)$ is the output of the last layer of the network. In classification tasks, the output space is $\mathscr{Y} := [0,1]^K$, and the output vector is the probability assigned to each class.

Sensitivity

Let us inject an external noise to the input of the network and compute the resulting noise in the output. If the original input vector is $x \in \mathscr{X}$ to which we add an i.i.d. normal noise vector $\varepsilon_x \sim \mathcal{N}(0, \sigma_{\varepsilon_x}^2 I)$, then the output noise due to $\varepsilon_x \in \mathscr{X}$ is $\varepsilon_y = F_\theta(x+\varepsilon_x) - F_\theta(x)$. We use the variance of the output noise, averaged over its K entries, as a measure of sensitivity: $Sen_\theta = \text{Var}(\overline{\varepsilon_y})$. The average sensitivity is therefore

$$Sen = \mathbb{E}_\theta[Sen_\theta] = \mathbb{E}_\theta\left[\text{Var}_{x,\varepsilon_x}\left[\frac{1}{K}\sum_{k=1}^{K}\varepsilon_y^k\right]\right], \tag{2.2}$$

where ε_y^k is the k-th entry of the vector ε_y. To distinguish the sensitivity Sen computed on untrained networks from trained ones, we denote $Sen_{\text{before}} = \mathbb{E}_\theta[Sen_\theta]$ and $Sen_{\text{after}} = \mathbb{E}_{\theta^*}[Sen_{\theta^*}]$ when the expectation is over the network parameters before and after training, respectively. We consider an "unspecific" sensitivity (meaning that the average is taken over all the entries of the output noise), which requires unlabeled data samples, as opposed to the "specific" sensitivity (which is limited to the output of the desired class) defined in Tartaglione et al. [2018]. In our work, the input vectors x used for computing Sen are drawn from S_v, so that given a new test data point, the sensitivity Sen predicts which trained network performs better for this particular point, and therefore gives a real-time uncertainty metric for predicting unseen data. For a few network architectures, we computed Sen on the training set S and observed that its value is practically the same as Sen computed on the testing set S_v (see Fig. 2.14), which suggests that Sen as a generalization metric does not require sacrificing a set of training points for validation.

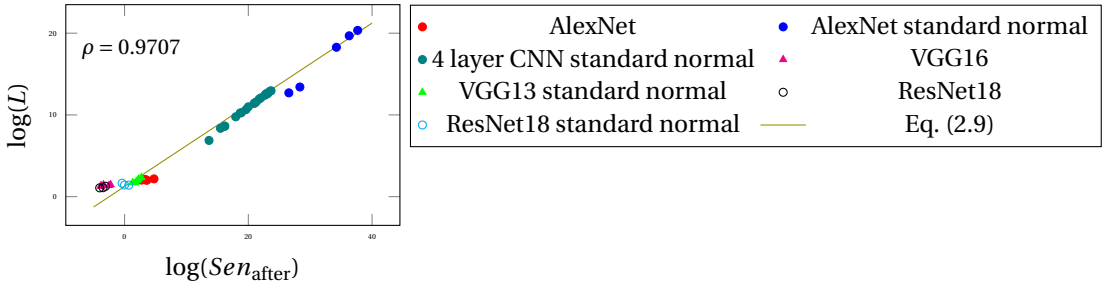


Figure 2.1: Test loss L versus sensitivity Sen_{after} for popular CNN architectures. The parameter initialization is Xavier [Glorot and Bengio 2010] with uniform distribution, unless stated as standard normal distribution. The networks are trained on a subset of the CIFAR10 training dataset and are evaluated on the entire CIFAR10 test dataset. Each point of the plot indicates a network with a different number of channels and hidden units, and its coordinates $\log(L)$ and $\log(Sen_{\text{after}})$ are averaged over 10 runs. For more details on configurations refer to Section 2.A. The Pearson correlation coefficient ρ between the data points is 0.9707.

2.1.4 Sensitivity versus Loss

Numerical Experiments

An ideal predictor should be robust: given similar inputs, the outputs should be close to each other. Assuming that the unseen data is drawn from the same distribution as the training data, the two concepts of robustness and generalization should therefore be related. Robustness here is the average-case robustness, not the worst-case robustness (adversarial robustness). We measure it by computing Sen (Eq. (2.2)), and considering near-zero training loss, we refer to the test loss L (Eq. (2.1)) as the generalization error. According to our observations on a wide set of experiments, including ResNets [He et al. 2016a] and VGGs [Simonyan and Zisserman 2014], we find a rather strong relation between Sen and L. State-of-the-art networks decrease the generalization error alongside with the sensitivity of the output of the network with respect to the input (Fig. 2.1).

Many factors influence the generalization performance of deep-learning models, among which network topology, initialization technique, and regularization method. In Section 2.1.5, we study the influence of each of these three factors on Sen and keep all the other factors, including the learning algorithm, the same throughout the experiments. These experiments suggest the use of Sen_{after} as a proxy to the test loss, which is particularly advantageous for settings where labeled training data is limited; assessing generalization performance can then be done without having to sacrifice training data for the validation set. Furthermore, Sen_{before} can potentially be used as an architecture-selection metric before training the models. We refer to fully-connected neural networks as FC, and to convolutional neural networks as CNN.

[2] The randomness is introduced by the stochastic optimization algorithm $\mathscr{A}$ and the randomized parameter initialization technique.

Bias-Variance Decomposition

In this section, a crude approximate relation between sensitivity and generalization error is established through the link between sensitivity and the variance term in the bias-variance decomposition of the mean square error (MSE). First, we find the link between the cross-entropy loss and MSE. Next, we develop the relation between sensitivity and the variance term, and finally, the link between sensitivity Sen and generalization error L.

When the predictor $f_{\theta^*}(x)$ assigns the probability $f^c_{\theta^*}(x)$ to the correct class c and $1-f^c_{\theta^*}(x)$ to another class (see Section 2.C for details), the cross-entropy loss L can be approximated as[3]

$$L \approx \mathbb{E}_{(x,y,\theta^*)}\left[\frac{1}{\sqrt{2}}\sqrt{\sum_{k=1}^{K}\left(f^k_{\theta^*}(x)-y^k\right)^2}\right]. \tag{2.3}$$

We roughly approximate the right-hand side in Eq. (2.3) with $\sqrt{L_{\text{MSE}}/2}$, where L_{MSE} is the mean square error criterion defined as

$$L_{\text{MSE}} = \mathbb{E}_{\theta^*}[L_{\theta^*_{\text{MSE}}}] = \mathbb{E}_{\theta^*}\left[\mathbb{E}_{(x,y)\sim S_v}\left[\left\|f_{\theta^*}(x)-y\right\|_2^2\right]\right]. \tag{2.4}$$

Consider the classic notion of bias-variance decomposition for the MSE loss [Geman et al. 1992; Tibshirani 1996; Neal et al. 2018; Mehta et al. 2019], where the generalization error is the sum of three terms: bias, variance and noise, i.e., $L_{\text{MSE}} = \varepsilon_{\text{bias}} + \varepsilon_{\text{variance}} + \varepsilon_{\text{noise}}$. In this chapter, we consider the labels to be noiseless and neglect the third term $\varepsilon_{\text{noise}}$. The bias term is formally defined as

$$\varepsilon_{\text{bias}} = \mathbb{E}_{x,y}\left[\left\|\mathbb{E}_{\theta^*}[f_{\theta^*}(x)]-y\right\|_2^2\right], \tag{2.5}$$

and the variance term is

$$\varepsilon_{\text{variance}} = \sum_{k=1}^{K}\mathbb{E}_x\left[\text{Var}_{\theta^*}(f^k_{\theta^*}(x))\right]. \tag{2.6}$$

Let us now draw an again crude approximate relation between $\varepsilon_{\text{variance}}$ and Sen under strong assumptions on the probability distributions of θ, x, and ε_x (refer to Section 2.B for more details). Given a feed-forward neural network with M hidden layers and H_l units per layer, $1 \le l \le M$, where the non-linear activation function is positive homogeneous[4] with parameters α and β (Eq. (2.11) in Section 2.B), we have

$$\varepsilon_{\text{variance}} \approx \left(\frac{K-1}{K}\right)\left(Sen \cdot \frac{\sigma_x^2}{\sigma_{\varepsilon_x}^2} + \Sigma\right), \tag{2.7}$$

where

$$\Sigma = \frac{1}{K}\sum_{l=1}^{M}\sigma_{b_l}^2\prod_{i=l+1}^{M}\left(\frac{\alpha^2+\beta^2}{2}\right)\sigma_{w_i}^2 H_i, \tag{2.8}$$

[3] This is more accurate for over-confident predictors (see Section 2.C).

[4] ReLU is a positive homogeneous function with $\alpha = 1$ and $\beta = 0$.

where K is the number of units in the output of the softmax layer and $\sigma^2_{w_l}$, $\sigma^2_{b_l}$, σ^2_x, and $\sigma^2_{\varepsilon_x}$ are the second moment of weights and biases of layer l, input x and input noise ε_x, respectively. Eq. (2.8) can be extended to convolutional neural networks[5] by replacing H_i with fan_{in} of layer i.

Given an infinite amount of training data, the bias represents the best performance of the model, which can be approximated by the training loss [Mehta et al. 2019; Ng 2012; Fortmann-Roe 2012]. In deep learning settings (and thus in our experiments), the training loss is close to zero, hence if we neglect the bias term $\varepsilon_{\text{bias}}$ in the decomposition of L_{MSE} we have

$$L \approx \sqrt{\frac{1}{2}\left(\frac{K-1}{K}\right)\left(Sen \cdot \frac{\sigma^2_x}{\sigma^2_{\varepsilon_x}} + \Sigma\right)}, \tag{2.9}$$

where Σ is given by Eq. (2.8). In the experiments, we observe that $\sigma^2_{b_l}$ is usually very small or zero (for instance in ResNets because $b_l = 0$), making $\Sigma \approx 0$.

According to Eq. (2.7) and the relation between L_{MSE} and L, to compare networks with a small value of $\varepsilon_{\text{bias}}$ (which is usually the case in deep neural networks where the bias is approximated by the near-zero training loss), the test loss can be approximated using the sensitivity by Eq. (2.9). Despite the strong assumptions and crude approximations to get Eq. (2.9), the numerical experiments show a rather surprisingly good match with Eq. (2.9) (Fig. 2.1, Fig. 2.2 and Fig. 2.3), even if Σ is neglected in Eq. (2.9). It is interesting to note that the right-hand side of Eq. (2.9) is computed without requiring labeled data points, whereas the left-hand side requires the ground-truth output vector y.

If $\varepsilon_{\text{bias}}$ can no longer be approximated by the training loss, which may in part explain the poorer match in lower values of Sen_{after} in Fig. 2.1, we need more training data to make this approximation valid. In Section 2.1.6 we train the networks with more data samples and observe that numerical results become closer to Eq. (2.9).

Sensitivity Before Versus After The Softmax Layer

Metric	ρ	Computation time
J after softmax	0.116	1.166 ± 0.111
J before softmax	0.414	1.165 ± 0.111
Sen after softmax	0.381	0.086 ± 0.006
Sen before softmax	**0.648**	**0.085** ± 0.006

Table 2.1: Pearson's correlation coefficient ρ between each metric (Sen, J) and the test loss (L), and average computation time (in seconds) of each metric for VGG13, VGG16, ResNet18 and ResNet34 networks trained on the CIFAR-10 dataset. The test accuracy of the networks are up to 87%.

[5] fan_{in} = the number of input channels $*$ the kernel size.

It is interesting to compare the sensitivity Sen given by Eq. (2.2) with the Frobenius norm of the Jacobian matrix J of the output of the softmax layer [Novak et al. 2018], in terms of their ability to gauge the generalization error L. A practical motivation for using Sen instead of J in real-world applications is computational tractability: to find the network architecture(s) with the best generalization ability among a collection of trained networks, the computation of Sen does not require to make a backward pass through each network architecture, contrary to J. In Table 2.1 we observe that computing Jacobian is more than 10 times slower than computing sensitivity. But the main motivation for using Sen is that it is computed *before* and not *after* the softmax layer, contrary to J in [Novak et al. 2018]. Because of the chain rule, J depends on the derivative of the softmax function with respect to the logits, which has very low values for highly confident predictors (the ones which assign a very high probability to one class and almost zero probability to the other classes). For instance, if the predictor erroneously assigns a high probability to a wrong class, the derivative of the softmax function is very low, resulting in a very low J. In this case, J would be misleading as it would mistakenly indicate good generalization. In contrast, Sen does not depend on the confidence level of the predictor. The difference is illustrated in Table 2.1 (see also Fig. 2.13 in the appendix), where the strong correlation between Sen and L (and the good match with Eq. (2.9)) is not found when Sen is replaced by the sensitivity *after* the softmax layer. Therefore, we observe from Table 2.1 that Sen computed before the softmax layer (given by Eq. (2.2)), is preferred to J (defined in [Novak et al. 2018]), both in terms of correlation to the test loss and of computation time.

2.1.5 Sensitivity as a Proxy for Generalization

In this section, we argue that methods improving the generalization performance of neural networks remarkably reduce the sensitivity Sen. We also present the experimental results for a regression task.

Comparing Different Architectures

Convolutional vs Fully-Connected Layers. The relation between the sensitivity Sen and the generalization error L supports the common view that CNNs outperform FCs in image-classification tasks. In Fig. 2.2 (a) we empirically observe that, given a CNN and an FC with the same number of parameters, the CNN has lower sensitivity and test loss than the FC. Moreover, some CNNs with more parameters than FCs have both lower sensitivity and lower test loss, even though they are more over-parameterized.
Let us start from a baseline classifier with one hidden layer (2 layers in total displayed in teal blue points in Fig. 2.2 (a), where each point represents a network with a different number of hidden units). We compare the effect of adding another fully-connected layer with adding a convolutional layer in Fig. 2.2 (a). We vary the number of parameters of 1-layer CNNs (which consist of 2 fully-connected (fc) layers and 1 conv layer, displayed by pink points) from 450k to 10M by increasing the number of channels and hidden units, whereas the number of parameters for 3-layer FCs varies from 320k to 1.7M (displayed by dark blue points). Despite

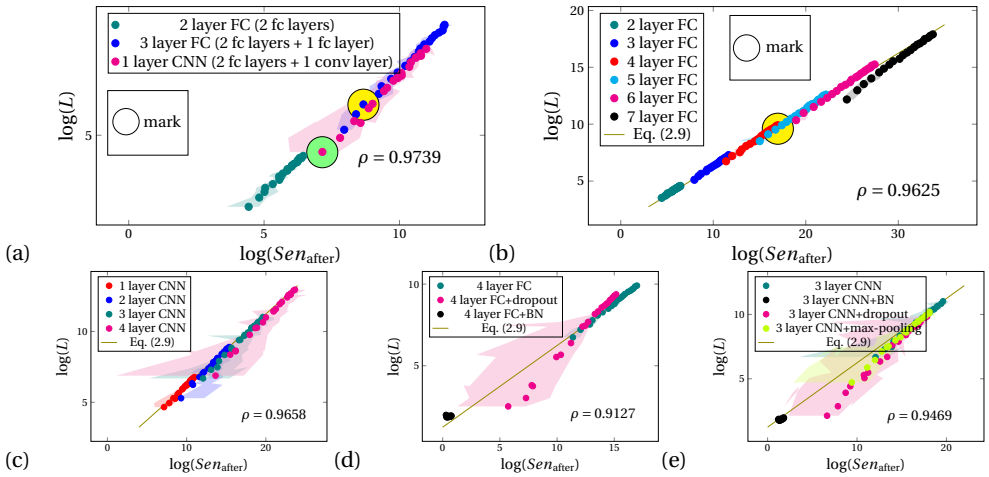


Figure 2.2: Test loss L versus sensitivity Sen_{after} for networks trained on a subset of the CIFAR-10 training dataset where the network parameters are initially drawn from a standard normal distribution. Each point of the plot indicates a network with a different number of channels and hidden units, and its coordinates $\log(L)$ and $\log(Sen_{\text{after}})$ are averaged over 10 runs. The shaded areas are contained by the minimum and maximum values of $\log(Sen_{\text{after}})$ over multiple runs of each experiment (point). **(a)** Comparison between adding a convolutional layer and adding a fully-connected layer to a baseline classifier that is a fully-connected neural network with one hidden layer. **(b)** Fully-connected neural networks. **(c)** Convolutional neural networks. **(d)** 4-layer FC trained with or without regularization. **(e)** 3-layer CNN trained with or without regularization.

the large number of parameters of CNNs, they suffer from less overfitting and have a lower sensitivity Sen than FCs. Next, let us compare a FC to a CNN with the same number of parameters in Fig. 2.2 (a): A 3-layer FC with 140 units in each layer (yellow mark) and a 1-layer CNN with 5 channels and 100 units (green mark), both have 450k parameters. The CNN has remarkably lower sensitivity and test loss than the FC, which indicates better performance compared to the FC with the same number of parameters.

Depth vs Width. Consider a feedforward FC with ReLU activation function where all the network parameters follow the standard normal distribution and are independent from each other and from the input. If we have M layers with H units in each hidden layer, K units in the output layer and D units in the input layer, then (see Section 2.D for details)

$$Sen = \frac{D}{K}\left(\frac{H}{2}\right)^{M}\sigma_{\varepsilon_x}^2. \tag{2.10}$$

According to Eq. (2.10), considering two neural networks with the same value for H^M, one deep and narrow (higher M and lower H), and the other one shallow and wide (lower M and higher H), the deeper network has lower sensitivity Sen. Assuming that both networks have near-zero training losses, depth is better than width regarding generalization in fully-connected neural networks. The empirical results in Fig. 2.2 (b) support Eq. (2.10). For instance,

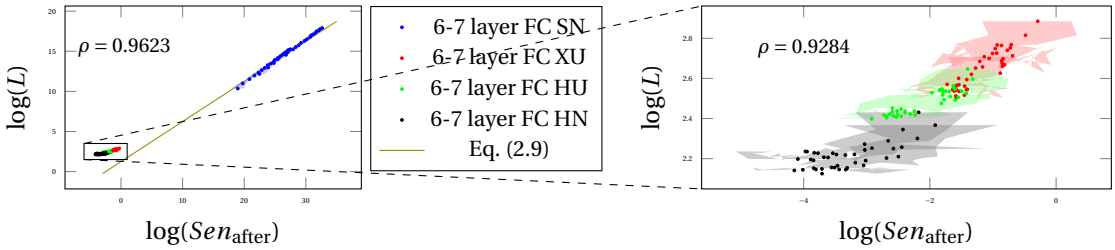


Figure 2.3: Test loss L versus sensitivity Sen_{after} for networks trained on a subset of the CIFAR-10 training dataset where networks are initialized with different methods. On the right, we have a zoom in plot of the bottom left frame of the left figure.

a 4-layer FC with 500 units per layer (the top right most point among all 4-layer FCs, indicated by a yellow mark), has the same value for $(H/2)^M$ as a 5-layer FC with 165 units per layer (the 4th point among 5-layer FCs, which exactly matches the yellow mark). In Fig. 2.2 (b), these two networks have the same values of both Sen_{after} and L, and all narrower 5-layer networks (with 100, 120, and 140 units) have better performance than the wide 4-layer network (with 500 units). A similar trend is observed for CNNs in Fig. 2.2 (c): having a narrower and deeper CNN is preferable to having a wider and shallower CNN.

Regularization Techniques

Fig. 2.2 (d) and (e) show the sensitivity Sen_{after} versus the test loss L, for different regularization methods. In particular, we study the effect of dropout [Srivastava et al. 2014] and batch normalization (BN) [Ioffe and Szegedy 2015] on the sensitivity in the FCs; and we apply dropout, BN and max-pooling for the CNNs. The results are consistent with the relation between sensitivity Sen_{after} and loss L. For all these regularization techniques, we observe a shift of the points towards the bottom left. This shift shows that these techniques known to improve generalization simultaneously decrease the network sensitivity to input perturbations. This is particularly noticeable in the BN case, where both the sensitivity and test loss decrease dramatically. This suggests that batch normalization improves performance by making the network less sensitive to input perturbations.

Initialization Methods

Another interesting observation is the effect of various parameter initialization techniques on the sensitivity and loss values, after the networks are trained (Fig. 2.3). We consider four initialization techniques for network parameters in our experiments: (i) Standard Normal distribution (SN), (ii) Xavier [Glorot and Bengio 2010] initialization method with uniform distribution (XU), (iii) He [He et al. 2015] initialization method with uniform distribution (HU), and (iv) He initialization method with normal distribution (HN). As shown in Fig. 2.3, the relation between sensitivity Sen_{after} and test loss L provides us with a new viewpoint on the success of the state-of-the-art initialization techniques; HN has the best generalization

performance, alongside the lowest sensitivity value (the black points in Fig. 2.3).

Sensitivity of Untrained Networks as a Proxy for Generalization Loss

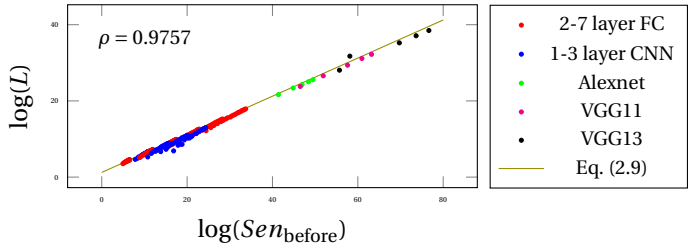


Figure 2.4: Test loss of trained models, L, versus sensitivity of untrained models, Sen_{before}, for networks whose parameters are initially drawn from the standard normal distribution. Note that the regularization techniques BN, dropout and max-pooling are removed from Alexnet, VGG11, and VGG13 configurations.

A similar trend is observed for neural networks that are not yet trained. In Fig. 2.4, the sensitivity Sen_{before} is measured before the networks are trained, and the test loss L is measured after the networks are trained. The parameters in the fully-connected and convolutional networks are initialized by sampling from the standard normal distribution, and no explicit regularization (dropout, BN, max-pooling) is used in the training process. These two conditions are necessary, because regularization techniques only affect the training process, hence Sen_{before} is the same for networks with or without regularization layers, and the He and Xavier initialization techniques force the sensitivity to be the same regardless of the number of units in hidden layers. Therefore, under these two conditions, the generalization performance of untrained networks with different architectures can be compared. The strong link between the sensitivity of untrained networks Sen_{before} and the test loss L observed in Fig. 2.4 suggests that the generalization of neural networks can be compared before the networks are even trained, making sensitivity a computationally inexpensive architecture-selection method. Later in this chapter, in Section 2.2, we present the results of using sensitivity before training as a neural architecture search (NAS) zero-cost tool in more details.

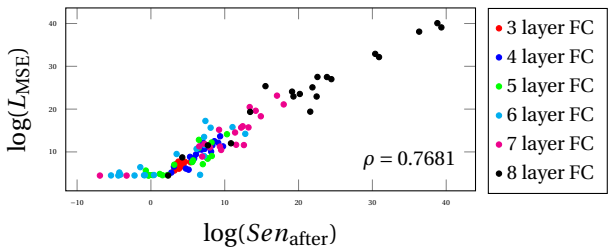


Figure 2.5: Test loss L_{MSE} versus sensitivity Sen_{after} for a regression task with the MSE loss criterion. The networks are trained and evaluated on the Boston house price dataset. Each point of the plot indicates a network with a different width and its coordinates are averaged over 10 runs.

Regression Task and MSE Loss

In this part, we investigate the relation between sensitivity and generalization error for regression tasks with the mean square error criterion (MSE). The loss function in this setting is defined in Eq. (2.4) where θ^* is found by minimizing MSE on training dataset S using the stochastic learning algorithm $\mathscr{A}$. Note that the output is the last layer of the neural network (the softmax layer is not applied), and that the output layer has 1 unit, i.e., $K = 1$, and that y is a scalar. The sensitivity is defined as $Sen = \mathbb{E}_{\theta^*}\left[\mathrm{Var}_{x,\varepsilon_x}\left[f_{\theta^*}(x+\varepsilon_x) - f_{\theta^*}(x)\right]\right]$ and the bias and variance terms are defined by Eq. (2.5) and Eq. (2.6), respectively. We consider the Boston housing dataset where the objective is to predict the price of a house given 14 features (including crime rate, distance to employment centers, etc.). Fig. 2.5 shows sensitivity versus test loss among fully-connected neural networks with 3-8 layers and 100-500 hidden units per layer; the networks are trained on 70% of the dataset and then evaluated on the remaining 30%. The results are consistent with the relation between sensitivity Sen and generalization error, which for the regression task is L_{MSE}. For a more detailed view, we observe that sensitivity is related to the variance in the bias-variance decomposition of the MSE loss (Fig. 2.11 (d) in the appendix), and the MSE loss is the sum of bias and variance terms (Fig. 2.11 (c) in the appendix).

2.1.6 Discussion and Conclusion

Discussion Regarding Bias

In this part, we discuss the role of the number of training samples and of the stage of the training on the validity of the approximation made in Eq. (2.9) where we neglect $\varepsilon_{\mathrm{bias}}$. In our experiments, we observe that when the number of training samples is low (see for instance Fig. 2.12 (a) for ResNet18 and ResNet34 networks), the match between experiments and Eq. (2.9) is rather poor. We show (in Fig. 2.12 (b) in the appendix) that this problem can be solved (at least in part) by training the networks with more samples: for instance, in Fig. 2.12 (b) the yellow marks are ResNet18, the green marks are ResNet34, and the results show a relation between $\log(Sen_{\mathrm{after}})$ and $\log(L)$ that becomes linear as we add more training data samples in the training process. Therefore, the larger the number of training samples is, the better the approximation $\varepsilon_{\mathrm{bias}} \approx trainloss$ becomes, and $\varepsilon_{\mathrm{variance}}$ becomes the more dominant term in the test loss. We also observe that, when computing sensitivity and loss at different stages of training, the bias term $\varepsilon_{\mathrm{bias}}$ in the test loss cannot be neglected at initial stages of the training. As the training progresses, the experimental results get closer to Eq. (2.9) (see Fig. 2.12 (d) in the appendix).

Final Remarks

As discussed in Section 2.1.4, the loss can be decomposed in three terms: $\varepsilon_{\mathrm{variance}}$, $\varepsilon_{\mathrm{bias}}$, and $\varepsilon_{\mathrm{noise}}$. The proposed relation between sensitivity Sen and variance $\varepsilon_{\mathrm{variance}}$ is extended to a

relation between Sen and generalization loss L when $\varepsilon_{\text{variance}}$ is the dominant term in the decomposition of the loss, which is often the case in deep learning settings. In the previous section, we discussed the possibility that the bias term $\varepsilon_{\text{bias}}$ might not be negligible compared to $\varepsilon_{\text{variance}}$, when the number of training samples is low and when the training loss is large. When the available data contains randomly labeled samples, then $\varepsilon_{\text{noise}}$ can no longer be neglected. As Sen does not depend on the labels, the randomness in the labels, and therefore the generalization performance of the model, can no longer be entirely captured by sensitivity in this setting. Furthermore, the pixel-wise linear input perturbations considered in our experiments might not be realistic; ideally, we would like to perturb the input in the latent space of the generative model of the input image. Also, the relation between Sen and L requires the non-linearity to be positive homogeneous. The generalization properties of networks with sigmoid and tanh activation functions are left for future work.

The sensitivity Sen changes with input-output re-scaling: For a homogeneous predictor, if the input data scale is multiplied by a factor α, and the output is divided by the factor α, then L remains unchanged, whereas Sen gets divided by α^4. However, as long as we compare networks subject to the same input data distribution, this re-scaling obviously does not happen. Moreover, Sen can be affected by output re-scaling: If the output of a classifier is divided by a factor α, then the classification accuracy remains the same, whereas L and Sen get divided by (approximately) α and α^2, respectively. While the relation between L and Sen remains valid, there is a mismatch between accuracy and loss, which suggests that the networks are miscalibrated. Applying network calibration methods such as *temperature scaling* [Guo et al. 2017] can potentially increase the correlation between the cross-entropy loss L and the classification error (i.e., 1 - accuracy), as well as the correlation between the sensitivity Sen and the classification error (see e.g., Table 2.3 in the appendix).

The relation between sensitivity and variance can be extended to any loss that admits a bias-variance decomposition. Therefore, if such a decomposition is found for the classification error (which might not be purely additive [Domingos 2000]), which is still an active research topic, then the link between sensitivity and error would follow. We note that there is a difference between causality and correlation between a complexity measure and generalization, as discussed in [Jiang et al. 2019]. We study the correlation between sensitivity Sen and generalization loss L, however, this does not imply that there is a causal relation between the two.

Conclusion. We find that the sensitivity metric is a strong indicator of overfitting. Given multiple networks having near-zero training losses to choose from with different hyper-parameters, the best architecture appears to be the one with the lowest sensitivity value. Sensitivity can also potentially be used as an architecture-selection method. One of the advantages of the sensitivity metric is that it can provide a loose prediction of the test loss without the use of any labeled data. This is especially important in applications where labeling data is expensive.

2.2 Neural Architecture Search Without Training

Neural Architecture Search (NAS) is a research field focused on automating neural network design. The process typically involves selecting a base network and modifying its architecture to optimize test performance. However, current NAS algorithms can be computationally expensive, limiting their practical applications. To address this issue, in this chapter, we examine the neural network output sensitivity, computed prior to training, as a zero-cost metric that could accelerate NAS algorithms. Our results demonstrate that this metric can be efficiently calculated for networks with a high number of parameters and effectively utilized as a pruner for the NAS search space.

2.2.1 Introduction and Background

Over the past decade, deep learning has demonstrated remarkable effectiveness in automating various tasks, particularly in Computer Vision. This is primarily due to the network's ability to learn relevant features for each domain, the abundance of data available from various disciplines, and the increase in computational power facilitated by GPUs and TPUs [Jouppi et al. 2017]. However, deep learning also requires architectural engineering to design a neural network that can be trained efficiently and effectively for a given task. This can also be done via automating the network design process. Neural architecture search (NAS) [Zoph and Le 2016; Elsken et al. 2019] is a research field that has emerged from various efforts to automate architectural design and has been able to produce numerous state-of-the-art networks. A NAS procedure typically involves the search space, optimization method, and candidate evaluation method, which we briefly discuss below.

The Search Space is crucial for NAS procedures as it determines the networks we examine to create the final architecture. Choosing a suitable search space can reduce complexity and computational demands, allowing random searches to produce excellent results. However, creating a good search space is challenging; Firstly, it needs prior dataset knowledge, limiting its usefulness for new areas. Secondly, it introduces bias by disregarding unexplored architectures that may provide superior performance [Kyriakides and Margaritis 2020].

The Optimization Method used in NAS affects the search efficiency and the final architecture. It involves balancing exploration and exploitation. Choosing the right strategy can ensure adequate exploration of the search space and optimal architecture. Various methods, including evolutionary [Chen et al. 2018] and reinforcement learning algorithms [Zoph and Le 2016], have been proposed in recent years.

The Evaluation Method in NAS refers to the process of assessing the performance of candidate architectures to identify the most effective one. Typically, the evaluation involves training and testing the architecture on a given dataset using a specific metric or a combination of metrics, such as accuracy [Kyriakides and Margaritis 2020].

In this study, our aim is to utilize the output sensitivity approach introduced in Section 2.1 to accelerate the search space by pruning the available networks. We do not focus on the optimization method and keep it consistent with the NAS benchmark used in our study. Furthermore, we use test accuracy as the evaluation method for finally selecting models. Our study focuses on the NAS-bench201 [Dong and Yang 2020] environment, which is a popular NAS benchmark. NAS benchmarks provide a standardized and cost-effective environment for evaluating various algorithms by pre-computing network evaluations for the entire search space and using table look-up for easy access to architecture performance. NAS-bench201 is a NAS benchmark that uses a fixed cell search space inspired by popular neural cell-based searching algorithms. Each architecture has a predefined skeleton and a searched cell, which is represented as a directed acyclic graph (DAG). The benchmark has 5 operation candidates and generates 15625 architectures, each trained on three datasets (CIFAR10[6], CIFAR100, and ImageNET [Deng et al. 2009]) multiple times. Training statistics and performance, including accuracy, loss, and number of parameters and floating-point operations (FLOPs), are provided for every run.

2.2.2 Zero-Cost Metrics

To evaluate candidate architectures for NAS, the most straightforward approach is to train and evaluate each one on a dataset. However, this is computationally expensive and time-consuming. To reduce costs, alternative methods emerge such as training for fewer epochs or using transfer learning. Another approach is to use zero-cost proxies to estimate the accuracy of a network before training. These proxies should be correlated with validation accuracy and preserve the ranking between models. In this section, we recall zero-cost proxies proposed in recent years. Furthermore, we suggest sensitivity, studied in Section 2.1 as an alternative zero-cost metric for estimating network test accuracy, *before* training.

We denote the i-th input sample of the dataset by $x_i \in \mathbb{R}^D$, and the network parameter vectors by θ. The neural network is denoted by $f : \mathbb{R}^D \to \mathbb{R}^K$, where K is the number of classes.

Synflow: a parameter pruning metric which is computed at initialization and proposed by Abdelfattah et al. [2021]. This criterion approximates the change in loss when a specific parameter is removed:

$$\text{Synflow} : \mathscr{S}_p(\theta) = \frac{\partial \mathscr{L}}{\partial \theta} \odot \theta$$

where $\mathscr{L}$ is the product of all parameters in the network, $\odot$ is the Hadamard product, and $\mathscr{S}_p$ is the per-parameter synflow values. The network metric will be then calculated by taking the sum of synflow over all parameters. No data is needed to compute synflow, and hence it is a zero-cost metric as it doesn't require network training.

Jacob-cov: for a mini-batch of data $\{x_i\}_1^{n_B}$ with size n_B with the Jacobian vector $J = \left(\frac{\partial f(x_1)}{\partial x_1}, \frac{\partial f(x_2)}{\partial x_2}, \cdots, \frac{\partial f(x_{n_B})}{\partial x_{n_B}}\right)$, the Pearson correlation coefficient matrix of J is denoted by C.

[6] https://www.cs.toronto.edu/~kriz/cifar.html

Jacob-cov [Mellor et al. 2021] is then defined as $\sum_{p=1}^{P}\left(\log(\lambda_p+\varepsilon)+\frac{1}{\lambda_p+\varepsilon}\right)$, where λ_p is the p-th eigenvalue of C and ε is a small constant. Mellor et al. [2021] suggest that this metric is negatively correlated with the test accuracy of the networks.

EPE: is similar to Jacob-cov, but the Jacobians are grouped together based on the label of each data point. Lopes et al. [2021] suggest that the test accuracy is positively related to the sum of the calculated correlation values.

Condition value of the neural tangent kernel (NTK): The condition value of the corresponding kernel matrix H for NTK is proposed as a metric of trainability by Chen et al. [2021b] and is suggested that this metric is negatively correlated to the final network test accuracy.

Number of linear regions: Chen et al. [2021b] propose that expressivity plays a critical role in the performance of the network. The expressivity of a ReLU network is measured by the number of linear regions it can generate.

Sensitivity: In Section 2.1, we show that the output sensitivity of a trained neural network with respect to its inputs can estimate the generalization performance of the network. It was also briefly studied how the sensitivity computed before training could also estimate generalization. We use sensitivity metric Sen_{before} defined in Equation 2.2 for networks that are not yet trained. Sensitivity is computationally efficient as it does not require any backward computation, and is zero-cost as it is computed at initialization, before training.

2.2.3 Experiments

In this section, we empirically compare the zero-cost metrics recalled in the previous section in terms of their correlation to the test accuracy and their applicability as a NAS search space pruner.

Correlation Between Each Metric and Test Accuracy: We conducted a comparison of the correlation between each metric and the test accuracy of the NAS-bench201 networks in Table 2.2 for CIFAR-10, CIFAR-100, and ImageNet datasets. Our analysis shows that synflow and Sen_{before} exhibit the highest correlations across all three datasets, with Sen_{before} consistently outperforming synflow. Furthermore, NTK and number of linear regions display the lowest correlations among all metrics. Additionally, the correlation values reveal that Jacob-cov is not a stable metric, as it positively correlates with the test accuracy of the CIFAR-10 and CIFAR-100 networks but negatively with the ImageNet networks.

Using Each Metric in NAS Search Space: One approach to reduce the search space in NAS is to leverage the proposed metrics. To implement this approach, we first calculate each metric for all networks and identify the top 10% networks based on each metric. We then select the best 10% networks with the highest test accuracy and compare their mean and maximum accuracy with the mean and maximum accuracy of the networks selected by each metric. The results for each dataset are presented in Fig. 2.6. The results reveal that, synflow and Sen_{before}

Metric	Spearman	Kendall τ
Synflow	**0.777**	**0.581**
Jacob-cov	0.689	0.524
EPE	0.675	0.496
NTK	0.331	0.244
Linear Regions	0.267	0.218
Sen_{before}	**0.792**	**0.607**

(a) CIFAR-10

Metric	Spearman	Kendall τ
Synflow	**0.763**	**0.568**
Jacob-cov	0.687	0.523
EPE	0.694	0.515
NTK	0.519	0.377
L inear Regions	0.477	0.345
Sen_{before}	**0.763**	**0.583**

(b) CIFAR-100

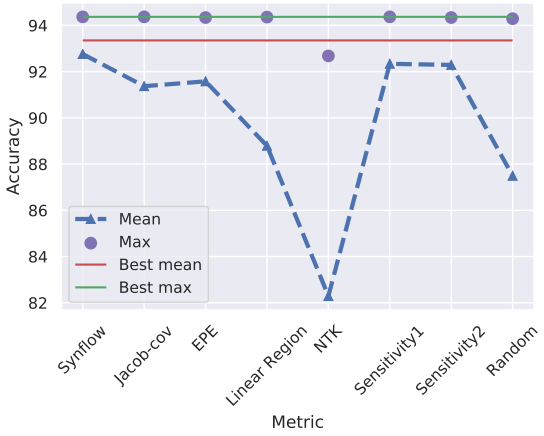
Metric	Spearman	Kendall τ
Synflow	**0.751**	**0.561**
Jacob-cov	-0.713	-0.534
EPE	0.634	0.457
NTK	0.414	0.295
Linear Regions	0.278	0.222
Sen_{before}	**0.764**	**0.585**

(c) ImageNet

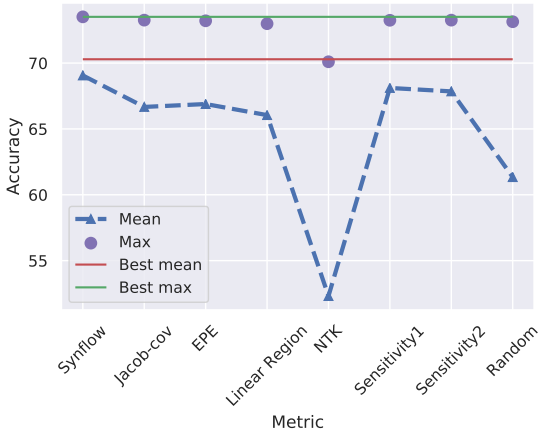
Table 2.2: Spearman and Kendall τ correlation coefficients between each metric and the test accuracy of the networks trained on CIFAR-10, CIFAR-100 and ImageNet datasets in NAS-bench201. We can observe that Sen_{before} consistently outperforms other metrics in all three datasets.

consistently select the best networks for all three datasets. Since selecting networks based on these metrics is equivalent to choosing the top 10% networks based on test accuracy, we can conclude that these metrics effectively function as a pruner, significantly reducing the number of networks that need to be trained.

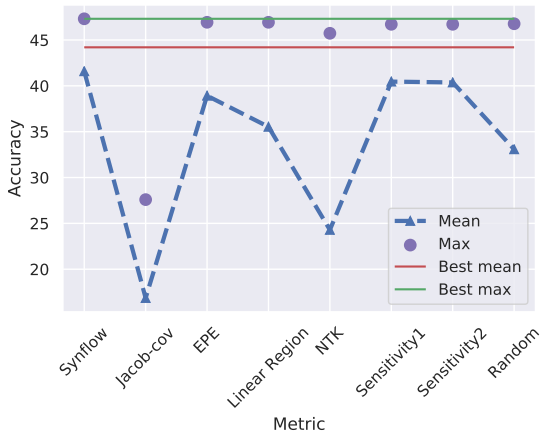
Computational Cost: The primary goal of utilizing metrics to prune the NAS search space is to reduce computational costs and accelerate the search process. Therefore, a suitable metric should not impose significant computational overhead. Sensitivity Sen_{before}, for instance, requires only two forward passes, while other metrics require both forward and backward passes, as well as additional computations, such as eigenvalue decomposition. Since final computations are more expensive than the forward pass, it can be inferred that Sen_{before} is



(a) CIFAR-10



(b) CIFAR-100



(c) ImageNet

Figure 2.6: The mean and maximum accuracy of the top 10% networks based on each metric. Sensitivity1 and sensitivity2 are both variants of Sen_{before}, where sensitivity1 is exactly computed according to Eq. (2.2) and sensitivity2 is the expectation of the ℓ_2 norm of ε_y. Random refers to the uniform selection of 10% of networks. The blue dashed line and the purple points refer to the average and to the maximum test accuracy of the selected networks according to each metric, respectively. The red and the green lines show the average and the maximum test accuracy of the top 10% models with highest test accuracies, respectively. These two lines serve as a reference to the optimal pruner. We can observe that Synflow and both variants of Sen_{before} perform rather comparatively to the optimal pruner in all three datasets. Hence, these metrics can be used to select models without any training, at least to prune the NAS search space.

more computationally efficient than other metrics. Moreover, other metrics necessitate a considerable amount of memory as they perform memory-intensive computations, such as computing the correlation matrix and eigenvalue decomposition. In contrast, Sen_{before} does not require as much memory.

2.2.4 Conclusion

In our study, we focused on sensitivity as a zero-cost metric that correlates with the final test accuracy of a network. We found that this metric can be efficiently calculated even for networks with a large number of parameters, where other existing metrics may struggle. Our results demonstrate that sensitivity is a reliable indicator of a network's performance and can serve as a useful tool for network architecture search (NAS).

One significant advantage of sensitivity is that it can be used as a NAS search space pruner. By ranking candidate architectures based on their sensitivity values, we can significantly reduce the search space and focus on the most promising architectures. This can save considerable computational resources and accelerate the search process.

Looking ahead, a future direction is to extend our study to non-vision tasks such as natural language processing (NLP). The definition of input perturbation differs for NLP tasks, and the approach to compute sensitivity needs to change accordingly. Nevertheless, it would be interesting to explore the potential of sensitivity as a valuable metric for NLP tasks, as we believe it could yield promising results in this domain. Overall, our study demonstrates the versatility and utility of sensitivity as a metric for network architecture search.

Appendices

2.A Experimental Details

The CIFAR-10[7] and the Boston house pricing[8] datasets are used for the image-classification and regression experiments presented in the main part of the chapter. The fully-connected neural networks have the same number of units in the hidden layers, varying from 100 to 500 with a step size of 20. For the convolutional neural networks the number of channels in convolutional layers vary from 5 to 25 with a step size of 5 (note that each time a channel is added in the convolutional layers, an extra 20 units is added to the fully-connected layers of the CNN accordingly). As it is computationally expensive to reach zero training loss for the entire dataset, we choose a randomly sampled subset of the training set containing 1000 samples of the CIFAR-10 dataset. Zero training loss is necessary for a fair comparison between different networks since we would like to have the same value for $\varepsilon_{\text{bias}}$ and $\varepsilon_{\text{noise}}$ among them. For the optimization algorithm, we choose the Adam optimizer with $lr = 0.001$ and $betas = (0.9, 0.999)$. We initialize the weights and biases with random values drawn from the distribution stated in each figure. The non-linear activation function is set to be the ReLU function throughout the experiments. We stop the training when the training loss reaches below the threshold 10^{-5} for 10 times. In case this condition is not met, we stop the training after 2000 epochs (each epoch is iterations over the mini-batches with size 128 of the training set). The noise added to the input image is a random tensor with the same size as the input and is drawn from the Gaussian distribution with zero mean and 0.1 standard deviation. The output noise is first averaged over all its K entries (for CIFAR-10 the number of classes is $K = 10$), then we take its variance over inputs of the testing dataset and the input noise. All the reported experimental results are averaged over 10 runs. Each experiment took few hours on one Nvidia Titan X Maxwell GPU.

We use the notations:

- Conv(number of filters, kernel size, stride, padding)
- Maxpool(kernel size)

[7] https://www.cs.toronto.edu/~kriz/cifar.html
[8] https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

- Linear(number of units)
- Dropout(dropout rate)

for layers of a convolutional neural network where Conv and Linear layers also include the ReLU non-linearity except the very last linear layer. The configurations that are used are:

- The Alexnet [Krizhevsky et al. 2012]: Conv(h, 3, 2, 1) - Maxpool(2) - Conv(3*h, 3, 1, 1) - Maxpool(2) - Conv(6*h, 3, 1, 1) - Conv(4*h, 3, 1, 1) - Conv(4*h, 3, 1, 1) - Maxpool(2) - Dense layer - Dropout(0.5) - Linear(4096) - Dropout(0.5) - Linear(4096) - Linear(K)
  where h $\in [16, 32, 48, 64, 80]$
- VGG13 [Simonyan and Zisserman 2014] : 2 x Conv(64*s, 3, 1, 1) - Maxpool(2) - 2 x Conv(128*s, 3, 1, 1) - Maxpool(2) - 2 x Conv(256*s, 3, 1, 1) - Maxpool(2) - 2 x Conv(512*s, 3, 1, 1) - Maxpool(2) - 2 x Conv(512*s, 3, 1, 1) - Maxpool(2) - Avgpool(2) - Dense layer - Linear(K)
  where s $\in [0.25, 0.5, 1, 1.5, 2]$ and all Conv layers have batch normalization
- Each block of a ResNet [He et al. 2016a] configuration: 2 x Conv(h, 3, 1, 1) + Conv(h, 1, 1, 1) which Conv layers include BN and ReLU and the result of the summation goes into a ReLU layer and h is the number of channels.

VGG16 is the same as VGG13 with the difference that it has three layers in the last three blocks. VGG11 configuration is the same as VGG13 except that in the first and second block it has one convolutional layer instead of 2. VGG19 is the same as VGG13 except that there is 4 conv layers instead of 2 in the last three blocks. ResNet18 has 2 blocks with h=64*s, 2 blocks with h=128*s, 2 blocks with h=256*s, and 2 blocks with h=512*s where s $\in [0.25, 0.5, 1, 1.5, 2]$. ResNet34 has 3 blocks with h=64*s, 4 blocks with h=128*s, 6 blocks with h=256*s, and 3 blocks with h=512*s where s $\in [0.25, 0.5, 1, 1.5, 2]$.

2.B Computation of Eq. (2.7): The Relation between Variance and Sensitivity

Computations of this section do not depend on the stage of the training, hence θ denotes the parameter vector at any stage of training. Let us recall the sensitivity metric (Eq. (2.2)) definition

$$Sen = \mathbb{E}_\theta \left[\mathrm{Var}_{x,\varepsilon_x} \left[\overline{\varepsilon_y} \right] \right],$$

where $\overline{\varepsilon_y} = 1/K \sum_{k=1}^{K} \varepsilon_y^k$, ε_y^k is the k-th entry of output noise vector ε_y given by

$$\varepsilon_y^k = F_\theta^k(x + \varepsilon_x) - F_\theta^k(x) \approxeq \varepsilon_x \cdot \nabla_x^\top F_\theta^k(x),$$

where we apply a first order Taylor expansion of the output. For a one hidden layer fully-connected neural network with D input units, H hidden units, and K output units, we have

$\theta = \{w_1 \in \mathbb{R}^{D\times H}, w_2 \in \mathbb{R}^{H\times K}, b_1 \in \mathbb{R}^H, b_2 \in \mathbb{R}^K\}$ where w_l and b_l are the weights and biases of layer l ($l = 1$ is the hidden layer and $l = 2$ is the output layer), which are independently drawn from a zero-mean normal distribution: $w_1 \sim \mathcal{N}(0, \sigma^2_{w_1} I)$, $w_2 \sim \mathcal{N}(0, \sigma^2_{w_2} I)$, $b_1 \sim \mathcal{N}(0, \sigma^2_{b_1} I)$, and $b_2 \sim \mathcal{N}(0, \sigma^2_{b_2} I)$ (this assumption has been studied in [Bellido and Fiesler 1993]). We have

$$F^k_\theta(x) = \sum_{h=1}^{H} w_2^{hk} a(p^h) + b_2^k,$$

where w_l^{ij} is the weight connecting unit i in layer l to unit j in layer $l+1$, b_l^h is the bias term added to unit h in layer $l+1$, p^h is the output of the linear transformation in the hidden unit h, i.e.,

$$p^h = \sum_{d=1}^{D} w_1^{dh} x^d + b_1^h,$$

and the non-linear activation function $a(\cdot)$ is a positive homogeneous function of degree 1; i.e.,

$$a(x) = \begin{cases} \alpha x & x > 0, \\ \beta x & \text{otherwise}, \end{cases} \tag{2.11}$$

where α and β are non-negative hyper-parameters. ReLU follows Eq. (2.11) with $\alpha = 1$ and $\beta = 0$. By applying the chain rule we obtain

$$\varepsilon_y^k \approx \sum_{d=1}^{D} \varepsilon_x^d \frac{\partial F^k_\theta(x)}{\partial x^d} = \sum_{d=1}^{D} \varepsilon_x^d \sum_{h=1}^{H} w_2^{hk} w_1^{dh} \frac{\partial a(p^h)}{\partial p^h}.$$

Therefore, we have

$$\overline{\varepsilon_y} = \frac{1}{K} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{d=1}^{D} \varepsilon_x^d w_2^{hk} w_1^{dh} \frac{\partial a(p^h)}{\partial p^h}.$$

The network parameters are assumed to be independent from each other, and it is assumed that $x \perp\!\!\!\perp \theta$, and $\varepsilon_x \perp\!\!\!\perp \{\theta, x\}$. Moreover, the entries of the input vector x are independent from each other with the same second moment, i.e., $\sigma_x^2 = \mathbb{E}[(x^d)^2]$ for $1 \le d \le D$. Consider the input noise ε_x to be a vector of zero mean random variables, hence $Sen = \mathbb{E}_{\theta,x,\varepsilon_x}[\left(\overline{\varepsilon_y}\right)^2]$. Then the sensitivity becomes

$$\begin{aligned} Sen &= \frac{1}{K^2} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{d=1}^{D} \mathbb{E}_{\varepsilon_x}[(\varepsilon_x^d)^2] \mathbb{E}_{\theta,x}\left[(w_2^{hk})^2 (w_1^{dh})^2 \left(\frac{\partial a(p^h)}{\partial p^h} \right)^2 \right] \\ &= \frac{1}{K^2} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{d=1}^{D} \sigma^2_{\varepsilon_x} \sigma^2_{w_2} \sigma^2_{w_1} \frac{\alpha^2+\beta^2}{2} = \frac{HD}{K} \sigma^2_{\varepsilon_x} \sigma^2_{w_2} \sigma^2_{w_1} \frac{\alpha^2+\beta^2}{2}, \end{aligned} \tag{2.12}$$

where the second equation follows by computing the expectation for zero-mean normal parameters. Let

$$var = \mathbb{E}_x\left[\mathrm{Var}_\theta[out]\right], \tag{2.13}$$

where $out = 1/K \sum_{k=1}^{K} F_\theta^k(x)$. Because of the homogeneity of the non-linearity $a(\cdot)$, we have $a(p^h) = p^h \cdot \frac{\partial a(p^h)}{\partial p^h}$. Hence

$$out = \frac{1}{K} \sum_{k=1}^{K} \left[\sum_{h=1}^{H} w_2^{hk} \left(\sum_{d=1}^{D} w_1^{dh} x^d + b_1^h \right) \frac{\partial a(p^h)}{\partial p^h} + b_2^k \right].$$

Because the parameters are zero-mean, $var = \mathbb{E}_{\theta,x}\left[out^2\right]$ and we have

$$\begin{aligned}
var &= \frac{1}{K^2} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{d=1}^{D} \mathbb{E}\left[(x^d)^2 (w_2^{hk})^2 (w_1^{dh})^2 \left(\frac{\partial a(p^h)}{\partial p^h} \right)^2 \right] \\
&+ \frac{1}{K^2} \sum_{k=1}^{K} \sum_{h=1}^{H} \mathbb{E}\left[(w_2^{hk})^2 \right] \mathbb{E}\left[(b_1^h)^2 \left(\frac{\partial a(p^h)}{\partial p^h} \right)^2 \right] + \frac{1}{K^2} \sum_{k=1}^{K} \mathbb{E}\left[(b_2^k)^2 \right] \\
&= \frac{1}{K^2} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{d=1}^{D} \sigma_x^2 \sigma_{w_2}^2 \sigma_{w_1}^2 \frac{\alpha^2 + \beta^2}{2} \\
&+ \frac{1}{K^2} \sum_{k=1}^{K} \sum_{h=1}^{H} \sigma_{w_2}^2 \sigma_{b_1}^2 \frac{\alpha^2 + \beta^2}{2} + \frac{1}{K^2} \sum_{k=1}^{K} \sigma_{b_2}^2 \\
&= \frac{HD}{K} \sigma_x^2 \sigma_{w_2}^2 \sigma_{w_1}^2 \frac{\alpha^2 + \beta^2}{2} + \frac{H}{K} \sigma_{w_2}^2 \sigma_{b_1}^2 \frac{\alpha^2 + \beta^2}{2} + \frac{1}{K} \sigma_{b_2}^2,
\end{aligned}$$

which follows by taking the expectations over the parameters with zero-mean normal distributions. Therefore, we obtain

$$var = Sen \cdot \frac{\sigma_x^2}{\sigma_{\varepsilon_x}^2} + \frac{H}{K} \sigma_{w_2}^2 \sigma_{b_1}^2 \frac{\alpha^2 + \beta^2}{2} + \frac{\sigma_{b_2}^2}{K},$$

where σ^2 denotes the second moment of a random variable. Following the same computations for a neural network with M hidden layers, we have

$$var = Sen \cdot \frac{\sigma_x^2}{\sigma_{\varepsilon_x}^2} + \frac{1}{K} \sum_{l=1}^{M} \sigma_{b_l}^2 \prod_{i=l+1}^{M} \frac{\alpha^2 + \beta^2}{2} \sigma_{w_i}^2 H_i, \tag{2.14}$$

where K is the number of units of the output layer $M+1$. We refer to the second term in the right-hand side of Eq. (2.14) as Σ. Its value is a very rough approximation given the numerous assumptions made above, but in practice it can often be neglected because $\sigma_{b_l}^2$ is very small or zero (the ResNet configurations do not have biases) in most of our experiments. So far, an approximate relation between sensitivity and variance before the softmax function was established. Next, we find a relation between sensitivity before the softmax Sen and variance after the softmax layer $\varepsilon_{\text{variance}}$.

The first order Taylor expansion for an arbitrary function at the average of its input is

$$g(x) \approx g(\mathbb{E}[x]) + g'(\mathbb{E}[x])(x - \mathbb{E}[x]).$$

Taking the variance of $g(x)$, we have

$$\mathrm{Var}(g(x)) \approx \left(g'(\mathbb{E}[x])\right)^2 \mathrm{Var}(x).$$

Here the function $g(\cdot)$ is the softmax function with input vector $F_\theta(x)$ and output indices

$$f_\theta^k(x) = \frac{\exp(F_\theta^k(x))}{\sum_{i=1}^K \exp(F_\theta^i(x))},$$

for $1 \le k \le K$. The input of the softmax function is a K-dimensional vector, so the variance of the output includes the vector-matrix multiplication of the covariance matrix of the input and the gradient vector. We assume that the outputs of the last layer are independent from each other ($F_\theta^i \perp\!\!\!\perp F_\theta^j$ for $1 \le i, j \le K, i \ne j$), so the covariance matrix is a diagonal matrix. Because the parameters are considered to be zero-mean, the input of the softmax has zero mean, $\mathbb{E}[F_\theta^k(x)] = 0$ for $1 \le k \le K$, then

$$\begin{aligned}\mathrm{Var}(f_\theta^k(x)) &\approx \sum_{i=1}^K \left(\left.\frac{\partial f_\theta^k(x)}{\partial F_\theta^i(x)}\right|_{\mathbb{E}[F_\theta^k(x)]=0} \right)^2 \mathrm{Var}(F_\theta^i(x)) \\ &\approx \left(\frac{1}{K}\cdot\left(1-\frac{1}{K}\right)\right)^2 \mathrm{Var}(F_\theta^k(x)) + \left(-\frac{1}{K^2}\right)^2 \sum_{\substack{i=1 \\ i\ne k}}^K \mathrm{Var}(F_\theta^i(x)),\end{aligned}$$

as softmax(0) = $1/K$. Therefore,

$$\varepsilon_{\text{variance}} = \sum_{k=1}^K \mathbb{E}_x\left[\mathrm{Var}(f_\theta^k(x))\right] \approx K\left(\frac{(K-1)^2}{K^4} + \frac{K-1}{K^4}\right) K\cdot var = \left(\frac{K-1}{K}\right)\left(Sen\cdot\frac{\sigma_x^2}{\sigma_{\varepsilon_x}^2} + \Sigma\right),$$

which completes the computations.

2.C The Relation between the Cross Entropy Loss and the Mean Square Error

We rewrite the cross-entropy loss (Eq. (2.1)) as

$$L = \mathbb{E}_{\theta^*}[L_{\theta^*}] = \mathbb{E}_{x,c,\theta^*}\left[-\log(f_{\theta^*}^c)\right],$$

where $1 \le c \le K$ is the index of the true class for the input x, i.e., $y^c = 1$ and $y^k = 0$ for $k \ne c$. For simplicity we use the notation $f_{\theta^*}^c$ instead of $f_{\theta^*}^c(x)$ in this section. For the MSE loss we have

$$L_{\text{MSE}} = \mathbb{E}_{x,y,\theta^*}\left[\sum_{k=1}^K \left(f_{\theta^*}^k - y^k\right)^2\right].$$

Because $\sum_{k=1}^{K} f_{\theta^*}^k = f_{\theta^*}^c + \sum_{\substack{j=1 \\ j\neq c}}^{K} f_{\theta^*}^j = 1$ the summation inside the above expectation, can be rewritten by replacing y^k by their $0-1$ values

$$\sum_{k=1}^{K}\left(f_{\theta^*}^k - y^k\right)^2 = \left(1-f_{\theta^*}^c\right)^2 + \sum_{\substack{j=1 \\ j\neq c}}^{K}\left(f_{\theta^*}^j\right)^2 = \left(1-f_{\theta^*}^c\right)^2 + \left(1-f_{\theta^*}^c\right)^2 - \sum_{\substack{i=1 \\ i\neq c}}^{K}\sum_{\substack{j=1 \\ j\neq i,c}}^{K} f_{\theta^*}^i f_{\theta^*}^j$$

$$= 2\left(1-f_{\theta^*}^c\right)^2 - \sum_{\substack{i=1 \\ i\neq c}}^{K}\sum_{\substack{j=1 \\ j\neq i,c}}^{K} f_{\theta^*}^i f_{\theta^*}^j.$$

Since $0 \leq f_{\theta^*}^j \leq (1-f_{\theta^*}^c)$ for $1 \leq j \leq K, j \neq c$ and $\sum_{\substack{j=1 \\ j\neq c}}^{K} f_{\theta^*}^j = 1 - f_{\theta^*}^c$, the above equation is bounded by

$$\left(\frac{K}{K-1}\right)\left(1-f_{\theta^*}^c\right)^2 \leq \sum_{k=1}^{K}\left(f_{\theta^*}^k - y^k\right)^2 \leq 2\left(1-f_{\theta^*}^c\right)^2. \tag{2.15}$$

The lower bound in the above inequality occurs when $f_{\theta^*}^j = (1-f_{\theta^*}^c)/(K-1)$ for $1 \leq j \leq K, j \neq c$ and the upper bound above occurs when all the remaining probability (i.e., $1-f_{\theta^*}^c$) is given to one class besides the true class c, and the rest of the classes are assigned with zero probability. The inequality in Eq. (2.15) can be rewritten in the following inequality

$$\sqrt{\frac{1}{2}\sum_{k=1}^{K}\left(f_{\theta^*}^k - y^k\right)^2} \leq 1 - f_{\theta^*}^c \leq \sqrt{\frac{K-1}{K}\sum_{k=1}^{K}\left(f_{\theta^*}^k - y^k\right)^2}. \tag{2.16}$$

Intuitively, the upper bound above is preferable in practice, because we would like the network to be less confident in assigning probabilities to wrong classes. If we take expectations in Eq. (2.16) and apply Jensen's inequality, the upper bound is upper bounded by $\sqrt{\frac{K-1}{K}}\sqrt{L_{\text{MSE}}}$. However, in our experiments, we often observe that the network assigns the probability $1-f_{\theta^*}^c$ to a wrong class and zero to the remaining classes, i.e., the network is over-confident. Therefore, if we consider this scenario, we then approximate $1-f_{\theta^*}^c$ with the lower bound above. Hence, by approximating the expectation of a squared root with the squared root of expectation, and by applying a first order Taylor expansion for the logarithm, i.e., $-\log(f_{\theta^*}^c) \approx 1-f_{\theta^*}^c$, we have[9]

$$L \approx \sqrt{\frac{L_{\text{MSE}}}{2}}. \tag{2.17}$$

[9] Note that if instead we would have considered the scenario that the network is not over-confident, then by approximating $1-f_{\theta^*}^c$ with the upper bound of inequality Eq. (2.16), we would have had $L \approx \sqrt{\frac{(K-1)L_{\text{MSE}}}{K}}$, which differs from Eq. (2.17) by only a constant scaling factor.

2.D Computation of Eq. (2.10)

Consider a feedforward FC with ReLU activation function ($\alpha = 1, \beta = 0$) where i.i.d. zero mean random noise ε_x with variance $\sigma^2_{\varepsilon_x}$ is added to the input. Then, assuming the output noise entries are independent from each other, we have

$$Sen = \frac{1}{K^2} \sum_{k=1}^{K} \mathrm{Var}\left[\varepsilon_y^k\right] = \frac{1}{K^2} \sum_{k=1}^{K} \mathbb{E}\left[(\varepsilon_y^k)^2\right].$$

If we have M hidden layers with $H_l, 1 \leq l \leq M$ units per layer, assuming the parameters are i.i.d. and independent from the input noise ε_x, and are drawn from the standard normal distribution, following the same computations as in Eq. (2.12) for a network with M hidden layers, D input units and K output units,

$$Sen = \frac{1}{K^2} \sum_{k=1}^{K} D\sigma^2_{\varepsilon_x} \prod_{l=1}^{M} \frac{H_l}{2}.$$

If all the hidden layers have the same number of units, $H_1 = H_2 = \cdots = H_M = H$, then,

$$Sen = \frac{D}{K} \left(\frac{H}{2}\right)^M \sigma^2_{\varepsilon_x}.$$

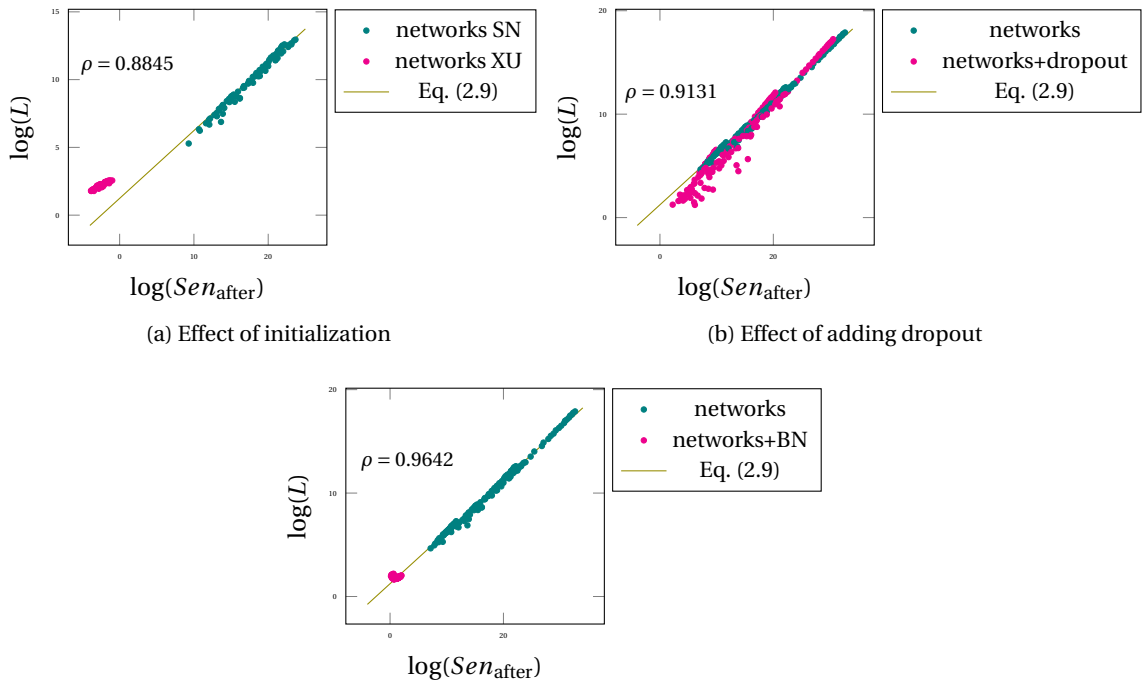
2.E CIFAR-10 Experiments

Fig. 2.7 shows the effect of different initialization techniques, and of adding dropout and batch normalization layers to fully-connected and convolutional neural networks trained on 1000 samples of the CIFAR-10 training dataset, and evaluated on the entire CIFAR-10 testing dataset. We observe again the strong relation between sensitivity Sen_{after} and generalization error L and the effect of these techniques on both Sen_{after} and L. In Fig. 2.8, we present the empirical results on the relation between var defined in Eq. (2.13) and Sen defined in Eq. (2.2). We experiment for 5 cases, where we change the second moment of the input σ_x^2 and the input noise $\sigma^2_{\varepsilon_x}$. In Fig. 2.8 (a) and (b), the original CIFAR-10 images are considered and in Fig. 2.8 (c), (d) and (e), we normalize the inputs accordingly to change σ_x^2. In all the figures, the empirical relation between var and S shows a good match with Eq. (2.14) where Σ is neglected.

2.F MNIST and CIFAR-100 Experiments

In this section, we present the experimental results for networks trained on 6000 samples of the MNIST[10] training dataset and evaluated on the entire MNIST testing dataset. Fig. 2.9 (a) and (b) show the results for fully-connected neural networks with different numbers of layers and hidden units and using regularization techniques batch normalization and dropout. Fig. 2.9 (c) and (d) show the results for convolutional neural networks. Finally, Fig. 2.9 (e)

[10] http://yann.lecun.com/exdb/mnist/



(a) Effect of initialization

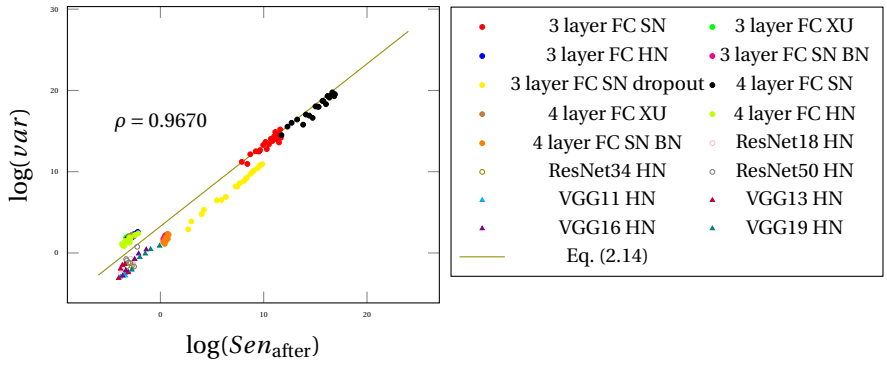
(b) Effect of adding dropout

(c) Effect of adding batch normalization

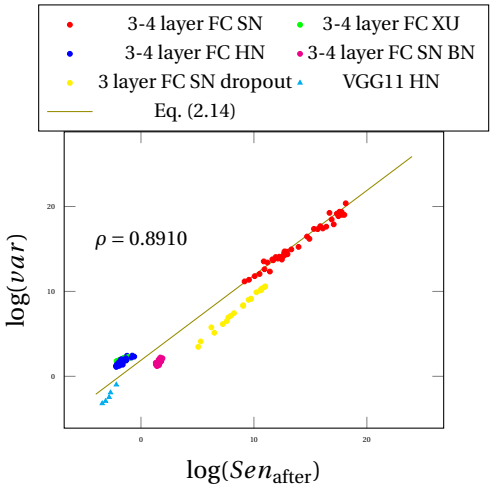
Figure 2.7: Test loss L versus sensitivity Sen_{after} for networks trained on 1000 samples of the CIFAR-10 training dataset presenting the effect of initialization, dropout and batch normalization. Each point represents a different architecture and its coordinates are averaged over 10 runs. **(a)** The networks are 5 layer FC, 2-4 layer CNN where the parameters are initially drawn from either Xavier uniform distribution (XU) or standard normal distribution (SN). **(b)** The networks are 3, 5, 7 layer FC and 1-4 layer CNN. The top right most pink point is the same network architecture as the top right most teal blue point when dropout is added to the configuration. Hence, for all network architectures we observe a shift of the numerical points towards bottom left of the figure when dropout is applied. **(c)** The networks are 3, 5, 7 layer FC and 1-4 layer CNN. In (b) and (c) the networks parameters are initially drawn from the standard normal distribution.

and (f) show the results on the comparison of the sensitivity of untrained networks Sen_{before} with the test loss L after the networks are trained. Fig. 2.10 shows the sensitivity Sen versus the loss L for networks trained on 1000 samples of the CIFAR-100 dataset[11]. The empirical results on these two datasets also show a rather strong match to Eq. (2.9), and once again we observe the relation between sensitivity and generalization and the effect of state-of-the-art techniques on both sensitivity and generalization.

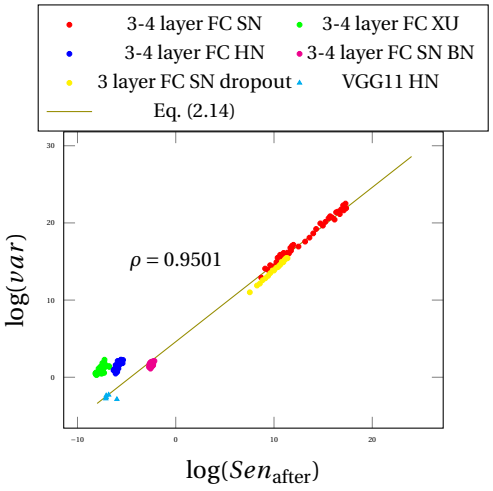
[11] https://www.cs.toronto.edu/~kriz/cifar.html



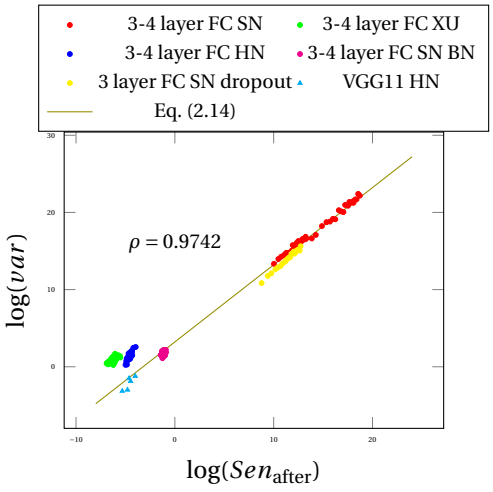
(a) $\sigma_x^2 = 0.2648$ and $\sigma_{\epsilon_x}^2 = 0.01$



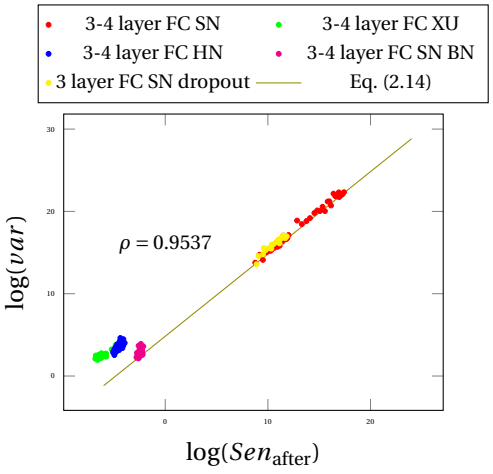
(b) $\sigma_x^2 = 0.2648$ and $\sigma_{\epsilon_x}^2 = 0.04$



(c) $\sigma_x^2 = 1$ and $\sigma_{\epsilon_x}^2 = 0.01$

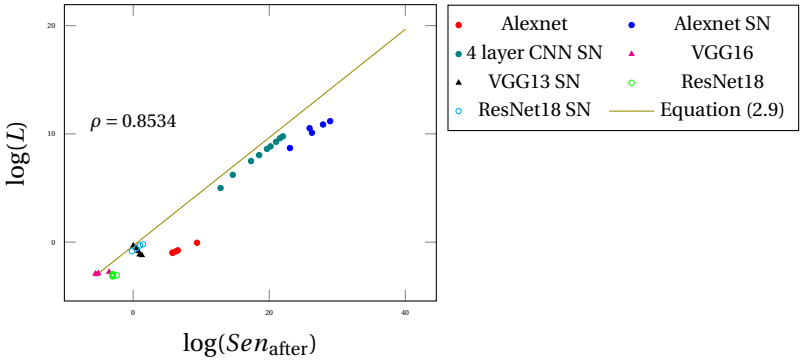


(d) $\sigma_x^2 = 1$ and $\sigma_{\epsilon_x}^2 = 0.04$

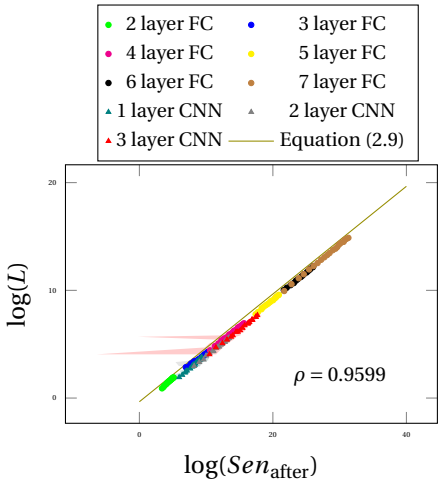


(e) $\sigma_x^2 = 1.25$ and $\sigma_{\epsilon_x}^2 = 0.01$

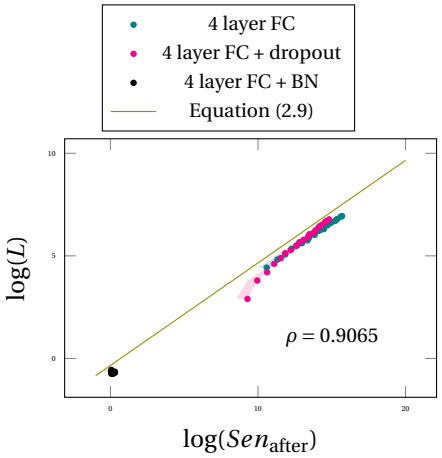
Figure 2.8: var (Eq. (2.13)) versus Sen (Eq. (2.2)) for networks trained on 1000 samples of the CIFAR-10 training dataset for different input x and input noise ϵ_x scales. The expression Σ is neglected in the computation of Eq. (2.14) in the figures. **(a), (b)** The non-normalized original CIFAR-10 input images. **(c), (d)** Normalized input images with zero-mean and unit variance. **(e)** Normalized inputs with unit variance and the same mean as the original images.



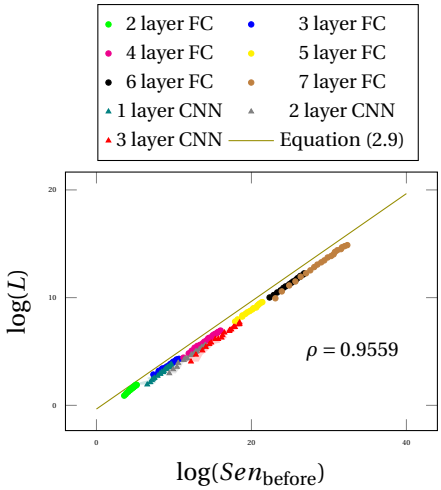
(a) Replicate of Fig. 2.1 for MNIST



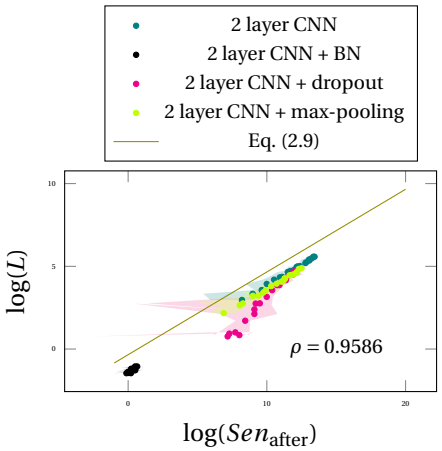
(b) Fully-connected neural networks



(c) 4-layer fully-connected neural networks trained with or without regularization

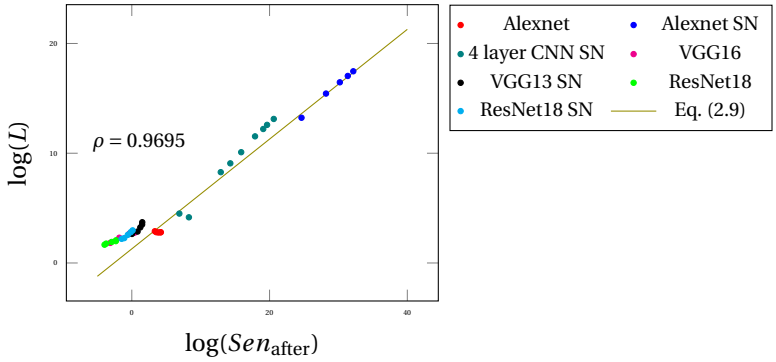


(d) Sensitivity before training

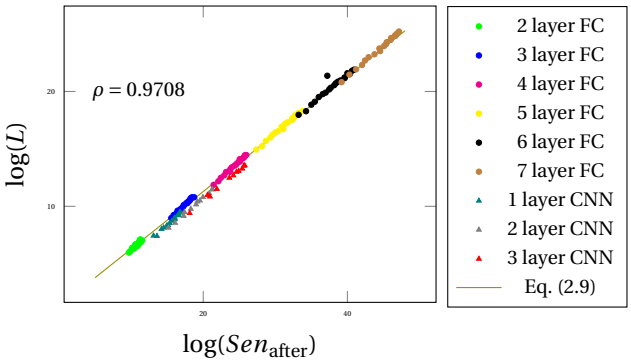


(e) 2-layer convolutional neural networks trained with or without regularization techniques

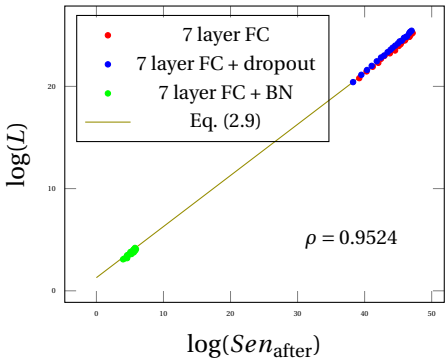
Figure 2.9: Test loss Sen versus sensitivity S for networks trained on 6000 samples of the MNIST training dataset. Each point in each color indicates a network with a different width and the sensitivity and test loss are averaged over multiple runs.



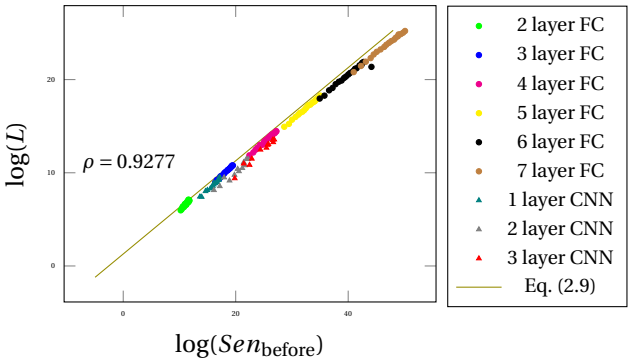
(a) Replicate of Fig. 2.1 for CIFAR-100



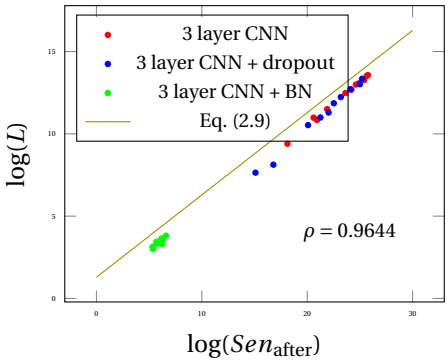
(b) Sensitivity after training vs test loss



(c) Effect of regularization on fully-connected neural networks



(d) Sensitivity before training vs test loss

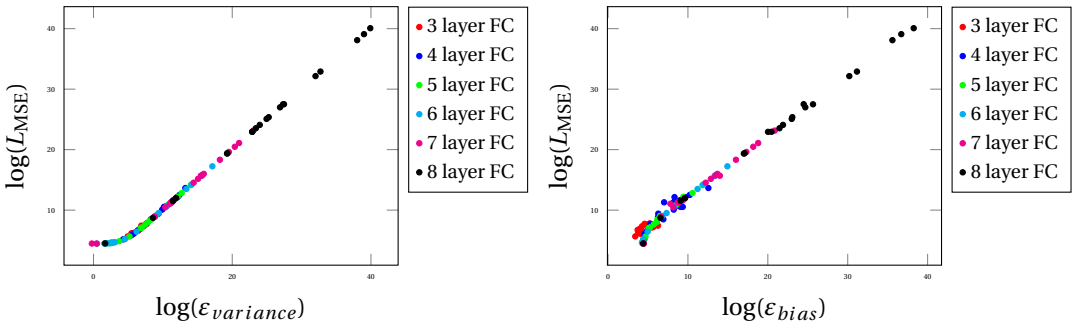


(e) Effect of regularization on convolutional neural networks

Figure 2.10: Test loss L versus sensitivity Sen for networks trained on 1000 samples of the CIFAR-100 training dataset. Each point indicates a network with a different width and the sensitivity and test loss are averaged over 10 runs.

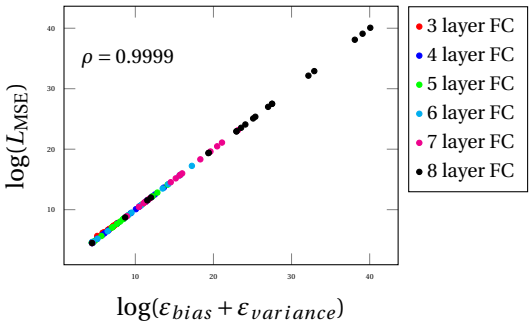
	before calibration	after calibration
ρ between L and Sen	0.958	0.841
ρ between L and classification error	-0.797	0.137
ρ between Sen and classification error	-0.757	0.087

Table 2.3: Comparison of Pearson's correlation coefficient ρ between cross-entropy loss L, sensitivity Sen and classification error for 4-layer CNN and 4-layer FC trained on a subset of the MNIST dataset before and after applying temperature scaling [Guo et al. 2017], which is a network calibration method.

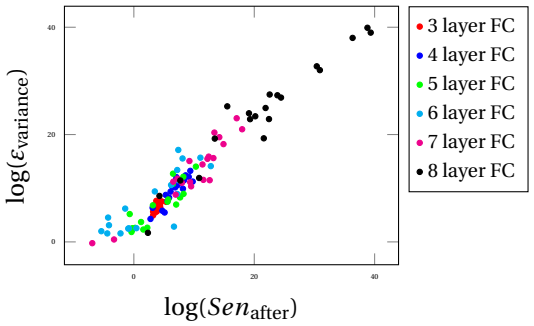


(a) Test loss versus variance

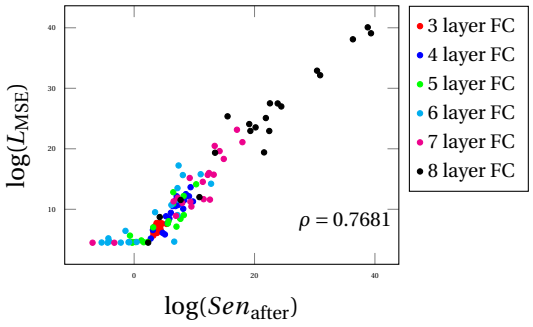
(b) Test loss versus bias



(c) Test loss versus sum of bias and variance

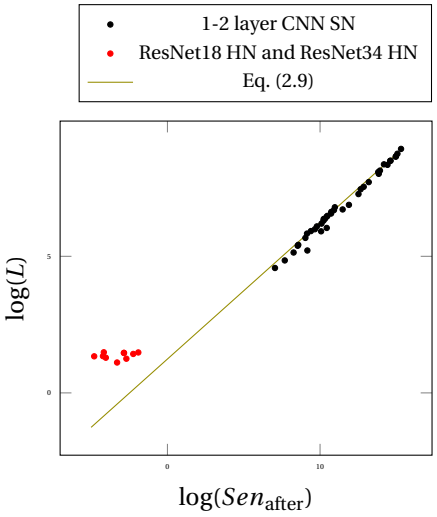


(d) Variance versus sensitivity Sen_{after}

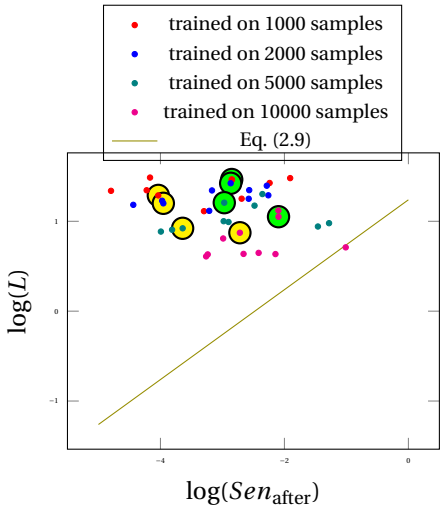


(e) Test loss versus sensitivity Sen_{after}

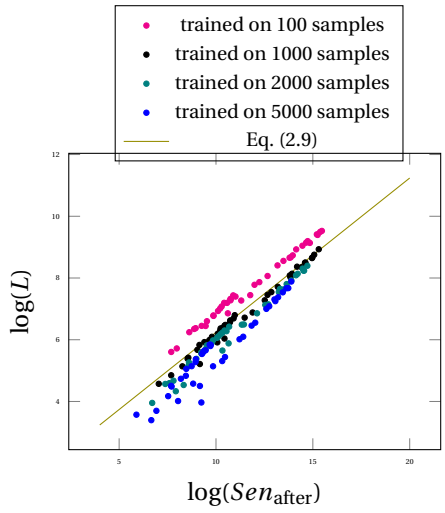
Figure 2.11: Test loss L versus variance $\varepsilon_{\text{variance}}$, bias $\varepsilon_{\text{bias}}$ and sensitivity Sen for a regression task using the MSE loss. The fully-connected neural networks are trained and evaluated on the Boston house price dataset. Each point represents a network with a different width and its coordinates are averaged over multiple runs.



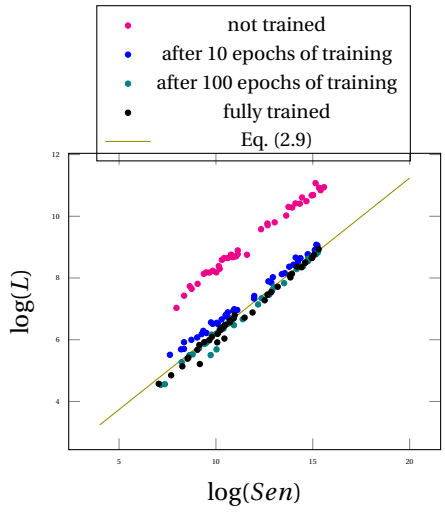
(a) Networks trained on 1000 samples of the training dataset



(b) ResNet18 and ResNet34, the yellow marks are ResNet18 with s=1, and the green marks are ResNet34 with s=0.5

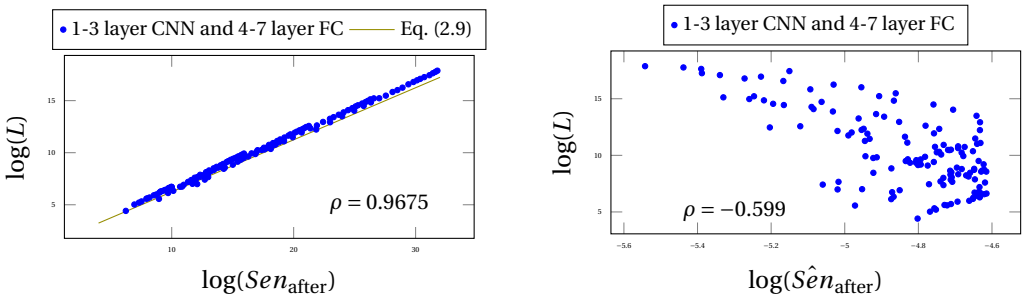


(c) 1-2 layer CNN



(d) 1-2 layer CNN

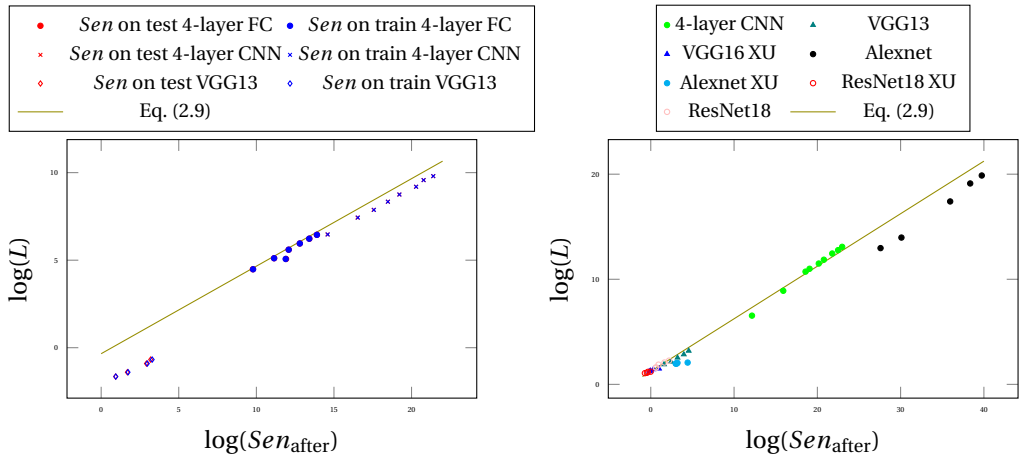
Figure 2.12: Test loss L versus sensitivity Sen for networks at different stages of training and trained on different numbers of training samples. Each point indicates an average over multiple runs of a network with a different width and depth. (b) is the zoom in of (a) on the bottom left, and we add the results of the same networks trained on a different number of samples. In (b) the network parameters are initially drawn from a normal distribution by using the He technique. (c) and (d) are the zoom in of (a) on the top right, and we add the results for the same networks trained with different number of training samples in (c) and at different stages of training in (d). In (c) and (d) the network parameters are drawn from the standard normal distribution. In all the figures the red and black points are the same experiments (1-2 layer CNNs trained on 1000 samples for the black points and ResNet18 and ResNet34 trained on 1000 samples for the red points). In (b) we observe how adding number of training samples, results in closer values to Eq. (2.9). In (d) with some abuse of notation, L is computed at different stages of training.



(a) Test loss versus sensitivity before softmax

(b) Test loss versus sensitivity after softmax

Figure 2.13: Test loss L versus sensitivity before and after the softmax layer for 1-3 layer CNNs and 4-7 layer FCs. The networks are trained on a subset of the CIFAR-10 training set. Sen_{after} is computed after training and *before* the softmax layer and follows Eq. (2.2); $\hat{Sen}_{\text{after}}$ is computed after training and *after* the softmax layer, i.e., $\hat{Sen}_{\text{after}} = \mathbb{E}_{\theta^*}\left[\text{Var}_{x,\varepsilon_x}\left[\frac{1}{K}\sum_{k=1}^{K} f^k_{\theta^*}(x+\varepsilon_x) - f^k_{\theta^*}(x)\right]\right]$. By expanding $\hat{Sen}_{\text{after}}$ up to the first order, it is approximated by the product of $\sigma^2_{\varepsilon_x}$ and the Frobenius norm of the Jacobian J of the output.



(a) MNIST dataset

(b) CIFAR-10 dataset

Figure 2.14: **(a)** Test loss versus sensitivity computed on the training set and the testing set for networks that are trained on 6000 samples of the MNIST training set. The Pearson correlation between Sen computed on the training set and Sen computed on the testing set is $\rho = 0.9999$ and in the figure these two values meet each other at the exact same points. **(b)** Test loss versus sensitivity computed on the training set for networks that are trained on 1000 samples of the CIFAR-10 training set. We observe that the strong match between the empirical results and Eq. (2.9) also holds for the sensitivity metric Sen when it is computed on the training dataset. It is interesting to note that the y-axis is computed over the testing dataset, whereas the x-axis is computed without using the testing set, suggesting Sen as a metric that does not require sacrificing the training samples for a validation set. In both figures the network parameters are initially drawn from the standard normal distribution unless otherwise stated as XU (Xavier technique with the uniform distribution).

3 Disparity Between Batches

3.1 A Signal for Early Stopping

In this chapter[1], we propose a metric for evaluating the generalization ability of deep neural networks trained with mini-batch gradient descent. Our metric, called *gradient disparity*, is the ℓ_2 norm distance between the gradient vectors of two mini-batches drawn from the training set. It is derived from a probabilistic upper bound on the difference between the classification errors over a given mini-batch, when the network is trained on this mini-batch and when the network is trained on another mini-batch of points sampled from the same dataset. We empirically show that gradient disparity is a very promising early-stopping criterion (i) when data is limited, as it uses all the samples for training and (ii) when available data has noisy labels, as it signals overfitting better than the validation data. Furthermore, we show in a wide range of experimental settings that gradient disparity is strongly related to the generalization error between the training and test sets, and that it is also very informative about the level of label noise.

3.1.1 Introduction

Early-stopping using a separate validation set is one of the most popular techniques used to avoid under/over fitting deep neural networks trained with iterative methods, such as gradient descent [Prechelt 1998; Yao et al. 2007; Gu et al. 2018]. The optimization is stopped when the performance of the model on a validation set starts to diverge from its performance on the training set. Early stopping requires an accurately labeled validation set, separated from the training set, to act as an unbiased proxy on the unseen test error. Obtaining such a reliable validation set can be expensive in many real-world applications as data collection is a time-consuming process that might require domain expertise. Furthermore, deep learning is becoming popular in applications for which there is simply not enough available data [Roh et al. 2019; Ipeirotis et al. 2010]. Finally, inexperienced label collectors, complex tasks

[1] This chapter is based on [Forouzesh and Thiran 2021].

(e.g., distinguishing a guinea pig from a hamster), and corrupted labels due for instance to adversarial attacks result in datasets that contain noisy labels [Frénay and Verleysen 2013]. Deep neural networks have the unfortunate ability to overfit to such small and/or noisy labeled datasets, an issue that cannot be completely solved by popular regularization techniques [Zhang et al. 2016a]. A signal of overfitting during training is therefore particularly useful, if it does *not* need a separate, accurately labeled validation set, which is the purpose of this chapter.

Let S_1 and S_2 be two mini-batches of points sampled from the available (training) dataset. Suppose that S_1 is selected for an iteration (step) of the mini-batch gradient descent (SGD), at the end of which the parameter vector is updated to θ_1. The average loss over S_1 (denoted by $L_{S_1}(f_{\theta_1})$) is in principle reduced, given a sufficiently small learning rate. The average loss $L_{S_2}(f_{\theta_1})$ over the other mini-batch S_2 is not as likely to be reduced. It is more likely to remain larger than the loss $L_{S_2}(f_{\theta_2})$ computed over S_2, if it was S_2 instead of S_1 that had been selected for this iteration. The difference $\mathscr{R}_2 = L_{S_2}(f_{\theta_1}) - L_{S_2}(f_{\theta_2})$ is the penalty that we pay for choosing S_1 over S_2 (and similarly, $\mathscr{R}_1$ is the penalty that we would pay for choosing S_2 over S_1). $\mathscr{R}_2$ is illustrated in Fig. 3.1 for a hypothetical non-convex loss as a function of a one dimensional parameter θ. The expected penalty measures how much, in an iteration, a model updated on one mini-batch (S_1) is able to generalize on average to another mini-batch (S_2) from the dataset. Hence, we call $\mathscr{R}$ the *generalization penalty*.

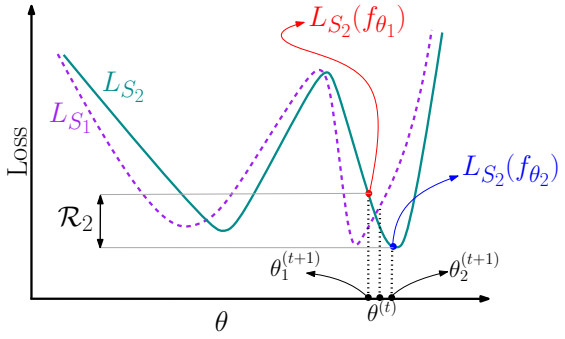


Figure 3.1: An illustration of the penalty term $\mathscr{R}_2$, where the y-axis is the loss, and the x-axis indicates the parameter of the model. L_{S_1} and L_{S_2} are the average losses over mini-batches S_1 and S_2, respectively. $\theta^{(t)}$ is the parameter at iteration t and $\theta_i^{(t+1)}$ is the parameter at iteration $t+1$ if batch S_i was selected for the update step at iteration t, with $i \in \{1,2\}$.

We establish a probabilistic upper bound on the sum of the expected penalties $\mathbb{E}[\mathscr{R}_1] + \mathbb{E}[\mathscr{R}_2]$ by adapting the PAC-Bayesian framework [McAllester 1999a;b; 2003], given a pair of mini-batches S_1 and S_2 sampled from the dataset (Theorem 1). Interestingly, under some mild assumptions, this upper bound is essentially a simple expression driven by $\|g_1 - g_2\|_2$, where g_1 and g_2 are the gradient vectors over the two mini-batches S_1 and S_2, respectively. We call it *gradient disparity*: it measures how much a small gradient step on one mini-batch negatively affects the performance on the other one.

We propose gradient disparity as an effective early stopping criterion, because of its computational tractability that makes it simple to use during the course of training, and because of its strong link with generalization error, as evidenced in the experiments that we run on state-of-the-art configurations. Gradient disparity is particularly well suited when the

Task	Method	Test Loss	Test AUC Score (in percentage)
Abnormal	5-fold CV	$0.284_{\pm 0.016}(0.307_{\pm 0.057})$	$71.016_{\pm 3.66}(87.44_{\pm 1.35})$
	GD	$\mathbf{0.274}_{\pm 0.004}(\mathbf{0.275}_{\pm 0.053})$	$\mathbf{72.67}_{\pm 3.85}(\mathbf{88.12}_{\pm 0.35})$
ACL	5-fold CV	$0.973_{\pm 0.111}(1.246_{\pm 0.142})$	$79.80_{\pm 1.23}(89.32_{\pm 1.47})$
	GD	$\mathbf{0.842}_{\pm 0.101}(\mathbf{1.136}_{\pm 0.121})$	$\mathbf{81.81}_{\pm 1.64}(\mathbf{91.52}_{\pm 0.09})$
Meniscal	5-fold CV	$0.758_{\pm 0.04}(1.163_{\pm 0.127})$	$73.53_{\pm 1.30}(72.14_{\pm 0.74})$
	GD	$\mathbf{0.726}_{\pm 0.019}(\mathbf{1.14}_{\pm 0.323})$	$\mathbf{74.08}_{\pm 0.79}(\mathbf{73.80}_{\pm 0.24})$

Table 3.1: Test loss and area under the receiver operating characteristic curve (AUC score) of the MRNet dataset [Bien et al. 2018] when using 5-fold cross-validation (5-fold CV) and gradient disparity (GD) as early stopping criteria for detecting the presence of abnormally, ACL tears, and meniscal tears from the sagittal plane MRI scans. The corresponding curves during training are shown in Fig. 3.15 (see Section 3.F.3 for more details). The results of early stopping are given, both when the metric (GD or validation loss) has increased for 5 epochs from the beginning of training and between parenthesis when the metric has increased for 5 consecutive epochs. Using GD outperforms 5-fold CV with either choice of the early stopping threshold. The standard deviations are obtained from 5 runs.

available dataset has limited labeled data, because it does not require splitting the available dataset into training and validation sets: all the available data can be used during training, unlike for instance k-fold cross-validation. We observe that using gradient disparity, instead of an unbiased validation set, results in a predictive improvement of at least 1% for classification tasks with limited and very costly available data, such as the MRNet dataset, which is a small size image-classification dataset used for detecting knee injuries (Table 3.1).

Moreover, we find that gradient disparity is a more accurate early stopping criterion than validation loss when the available dataset contains noisy labels. Gradient disparity reflects the label noise level quite well throughout the training process, especially at early stages of training. Finally, we observe that gradient disparity has a strong positive correlation with the test error across experimental settings that differ in training set size, batch size, and network width.

3.1.2 Related Work

The coherent gradient hypothesis [Chatterjee 2020] states that the gradient is stronger in directions where similar examples exist and towards which the parameter update is biased. He and Su [He and Su 2020] study the local elasticity phenomenon, which measures how the prediction over one sample changes, as the network is updated on another sample. Motivated by [He and Su 2020], reference [Deng et al. 2020] proposes generalization upper bounds using locally elastic stability. The generalization penalty introduced in our work measures how the prediction over one sample (batch) changes when the network is updated on the same sample, instead of being updated on another sample.

Finding a practical metric that completely captures the generalization properties of deep neural networks, and in particular indicates the level of label noise and decreases with the size of the training set, is still an active research direction [Dziugaite and Roy 2017; Neyshabur et al. 2017a; Nagarajan and Kolter 2019; Chatterji et al. 2020]. Recently, there have been a few studies that propose similarity between gradients as a generalization metric. The benefit of tracking generalization by measuring the similarity between gradient vectors is its tractability during training, and the dispensable access to unseen data. Sankararaman et al. [2019] propose gradient confusion, which is a bound on the inner product of two gradient vectors, and shows that the larger the gradient confusion is, the slower the convergence is. Gradient interference (when the gradient inner product is negative) has been studied in multi-task learning, reinforcement learning and temporal difference learning [Riemer et al. 2018; Liu et al. 2019; Bengio et al. 2020]. Yin et al. [2017] study the relation between gradient diversity, which measures the dissimilarity between gradient vectors, and the convergence performance of distributed SGD algorithms. Fort et al. [2019] propose a metric called stiffness, which is the cosine similarity between two gradient vectors, and show empirically that it is related to generalization. Fu et al. [2020] study the cosine similarity between two gradient vectors for natural language processing tasks. Reference [Mehta et al. 2020] measures the alignment between the gradient vectors within the same class (denoted by Ω_c) , and studies the relation between Ω_c and generalization as the scale of initialization (the variance of the probability distribution the network parameters are initially drawn from) is increased. These metrics are usually not meant to be used as early stopping criteria, and indeed in Table 3.2 and Table 3.15 in the appendix, we observe that none of them consistently outperforms k-fold cross-validation.

Another interesting line of work is the study of the variance of gradients in deep learning settings. Negrea et al. [2019] derive mutual information generalization error bounds for stochastic gradient Langevin dynamics (SGLD) as a function of the sum (over the iterations) of square gradient incoherences, which is closely related to the variance of gradients. Two-sample gradient incoherences also appear in [Haghifam et al. 2020], which are taken between a training sample and a “ghost" sample that is not used during training and therefore taken from a validation set (unlike gradient disparity). The upper bounds in [Negrea et al. 2019; Haghifam et al. 2020] are cumulative bounds that increase with the number of iterations and are not intended to be used as early stopping criteria. As shown in Section 3.H, gradient disparity can be used as an early stopping criterion not only for SGD with additive noise (such as SGLD), but also other adaptive optimizers. Reference [Qian and Klabjan 2020] shows that the variance of gradients is a decreasing function of the batch size. However, reference [Jastrzebski et al. 2020] hypothesizes that gradient variance counter-intuitively increases with the batch size, by studying the effect of the learning rate on the variance of gradients, which is consistent with our results on convolutional neural networks in Section 3.1.6. References [Jastrzebski et al. 2020; Qian and Klabjan 2020] mention the connection between variance of gradients and generalization as promising future directions. Our study shows that variance of gradients used as an early stopping criterion outperforms k-fold cross-validation (see Table 3.15).

	Min	GD/Var	EB	GSNR	$g_i \cdot g_j$	sign$(g_i \cdot g_j)$	cos$(g_i \cdot g_j)$	Ω_c	OV	k-fold	No ES
TE	13.76	**16.66**	24.63	35.68	37.92	24.63	35.68	29.40	34.36	17.86	25.72
TL	0.75	1.08	**0.86**	1.68	1.82	**0.86**	1.68	1.46	1.65	1.09	0.91

Table 3.2: Test error (TE) and test loss (TL) achieved by using various metrics as early stopping criteria for an AlexNet trained on the MNIST dataset with 50% random labels. See Table 3.15 in the appendix for further details and experiments.

Liu et al. [2020a] propose a relation between gradient signal-to-noise ratio (SNR), called GSNR, and the one-step generalization error, with the assumption that both the training and test sets are large. Mahsereci et al. [2017] also study gradient SNR and propose an early stopping criterion called evidence-based criterion (EB) that eliminates the need for a held-out validation set. Reference [Liu et al. 2008] proposes an early stopping criterion based on the signal-to-noise ratio figure, which is further studied in [Piotrowski and Napiorkowski 2013], a study that shows the average test error achieved by standard early stopping is lower than the one obtained by this criterion. Zhang et al. [2021c] empirically show that the variance term in the bias-variance decomposition of the loss function dominates the variations of the test loss, and hence propose optimization variance (OV) as an early stopping criterion.

Summary of Comparison to Related Work In Table 3.2 and Section 3.I, we compare gradient disparity (GD) to EB, GSNR, gradient inner product, sign of the gradient inner product, variance of gradients, cosine similarity, Ω_c, and OV. We observe that the only metrics that consistently outperform k-fold cross-validation as early stopping criteria across various settings (see Table 3.15 in the appendix), and that reflect well the label noise level (see in Fig. 3.26 and Fig. 3.27 that metrics such as EB and sign$(g_i \cdot g_j)$ do not correctly detect the label noise level), are gradient disparity and variance of gradients. The two are analytically very close as discussed in Section 3.I.2. However, we observe that the correlation between gradient disparity and the test loss is in general larger than the correlation between variance of gradients and the test loss (see Table 3.16 in the appendix).

3.1.3 Generalization Penalty

Consider a classification task with input $x \in \mathscr{X} := \mathbb{R}^D$ and ground truth label $y \in \{1, 2, \cdots, K\}$, where K is the number of classes. Let $f_\theta \in \mathscr{F} : \mathscr{X} \to \mathscr{Y} := \mathbb{R}^K$ be a predictor (classifier) parameterized by the parameter vector $\theta \in \mathbb{R}^d$, and $l(\cdot, \cdot)$ be the 0-1 loss function

$$l\left(f_\theta(x), y\right) = \mathbb{1}\left[f_\theta(x)[y] < \max_{j \neq y} f_\theta(x)[j]\right],$$

for all $f_\theta \in \mathscr{F}$ and $(x, y) \in \mathscr{X} \times \{1, 2, \cdots, K\}$. The expected loss and the empirical loss over the training set S of size n are respectively defined as

$$L(f_\theta) = \mathbb{E}_{(x,y)\sim\mathscr{D}}\left[l\left(f_\theta(x), y\right)\right], \tag{3.1}$$

and

$$L_S(f_\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f_\theta(x_i), y_i), \tag{3.2}$$

where $\mathscr{D}$ is the probability distribution of the data points and $S = \{(x_i, y_i)\}^n$ is a collection of n i.i.d. samples drawn from $\mathscr{D}$. Similar to the notation used in [Dziugaite and Roy 2017], distributions on the hypotheses space $\mathscr{F}$ are simply distributions on the underlying parameterization. With some abuse of notation, ∇L_{S_i} refers to the gradient with respect to the surrogate differentiable loss function, which in our experiments is cross entropy[2].

In a mini-batch gradient descent (SGD) setting, let mini-batches S_1 and S_2 have sizes n_1 and n_2, respectively, with $n_1 + n_2 \leq n$. Let $\theta = \theta^{(t)}$ be the parameter vector at the beginning of an iteration t. If S_1 is selected for the next iteration, θ gets updated to $\theta_1 = \theta^{(t+1)}$ with

$$\theta_1 = \theta - \gamma \nabla L_{S_1}\left(f_\theta\right), \tag{3.3}$$

where γ is the learning rate. The generalization penalty $\mathscr{R}_2$ is defined as the gap between the loss over S_2, $L_{S_2}\left(f_{\theta_1}\right)$, and its target value, $L_{S_2}\left(f_{\theta_2}\right)$, at the end of iteration t.

When selecting S_1 for the parameter update, Eq. (3.3) makes a step towards learning the input-output relations of mini-batch S_1. If this negatively affects the performance on mini-batch S_2, $\mathscr{R}_2$ will be large; the model is learning the data structures that are unique to S_1 and that do not appear in S_2. Because S_1 and S_2 are mini-batches of points sampled from the same distribution $\mathscr{D}$, they have data structures in common. If, throughout the learning process, we consistently observe that, in each update step, the model learns structures unique to only one mini-batch, then it is very likely that the model is memorizing the labels instead of learning the common data-structures. This is captured by the generalization penalty $\mathscr{R}$.

We adapt the PAC-Bayesian framework [McAllester 1999a;b] to account for the trajectory of the learning algorithm; For each learning iteration t we define a prior, and two possible posteriors depending on the choice of the batch selection. Let $\theta \sim P$ follow a prior distribution P, which is a $\mathscr{F}_t$-measurable function, where $\mathscr{F}_t$ denotes the filtration of the available information at the beginning of iteration t. Let $f_{\theta_1}, f_{\theta_2}$ be the two learned single predictors, at the end of iteration t, from S_1 and S_2, respectively. In this framework, for $i \in \{1,2\}$, each predictor f_{θ_i} is randomized and becomes f_{ν_i} with $\nu_i = \theta_i + u_i$, where u_i is a random variable whose distribution might depend on S_i. Let Q_i be the distribution of ν_i, which is a distribution over the predictor space $\mathscr{F}$ that depends on S_i via θ_i and possibly u_i. Let $\mathscr{G}_i$ be a σ-field such that $\sigma(S_i) \cup \mathscr{F}_t \subset \mathscr{G}_i$ and such that the posterior distribution Q_i is $\mathscr{G}_i$-measurable for $i \in \{1,2\}$. We further assume that the random variable $\nu_1 \sim Q_1$ is statistically independent from the draw of the mini-batch S_2 and, vice versa, that $\nu_2 \sim Q_2$ is independent from the batch S_1[3], i.e.,

[2] We have also studied networks trained with the mean square error in Section 3.E.3, and we observe that there is a strong positive correlation between the test error/loss and gradient disparity for this choice of the surrogate loss function as well (see Fig. 3.11).

[3] Mini-batches S_1 and S_2 are drawn without replacement, and the random selection of indices of mini-batches

$\mathscr{G}_1 \perp\!\!\!\perp \sigma(S_2)$ and $\mathscr{G}_2 \perp\!\!\!\perp \sigma(S_1)$.

Theorem 1. *For any $\delta \in (0,1]$, with probability at least $1-\delta$ over the sampling of sets S_1 and S_2, the sum of the expected penalties conditional on S_1, and S_2, respectively, satisfies*

$$\mathbb{E}[\mathscr{R}_1] + \mathbb{E}[\mathscr{R}_2] \leq \sqrt{\frac{2KL(Q_2||Q_1) + 2\ln\frac{2n_2}{\delta}}{n_2 - 2}} + \sqrt{\frac{2KL(Q_1||Q_2) + 2\ln\frac{2n_1}{\delta}}{n_1 - 2}}. \tag{3.4}$$

In this chapter, the goal is to get a signal of overfitting that indicates at the beginning of each iteration t whether to stop or to continue training. This signal should track the performance of the model at the end of iteration t by investigating its evolution over all the possible outcomes of the batch sampling process during this iteration. For simplicity, we consider two possible outcomes: either mini-batch S_1 or mini-batch S_2 is chosen for this iteration (we later in the next section extend to more pairs of batches). If we were to use bounds such as the ones in [McAllester 2003; Neyshabur et al. 2017b] for one iteration at a time, the generalization error at the end of that iteration can be bounded by a function of either $\mathrm{KL}(Q_1||P)$ or $\mathrm{KL}(Q_2||P)$, depending on the selected batch. Therefore, as each of the two batches is equally likely to be sampled, we should track $\mathrm{KL}(Q_1||P)$ and $\mathrm{KL}(Q_2||P)$ for a signal of overfitting at the end of the iteration, which requires in turn access to the three distributions P, Q_1 and Q_2. In contrast, the upper bound on the generalization penalty given in Theorem 1 only requires the two distributions Q_1 and Q_2, which is a first step towards a simpler metric since, loosely speaking, the symmetry between the random choices for S_1 and S_2 should carry over these two distributions, leading us to assume the random perturbations u_1 and u_2 to be identically distributed. If furthermore we assume them to be Gaussian, then we show in the next section that $\mathrm{KL}(Q_2||Q_1)$ and $\mathrm{KL}(Q_1||Q_2)$ are equal and boil down to a very tractable generalization metric, which we call gradient disparity.

3.1.4 Gradient Disparity

In Section 3.1.3, the randomness modeled by the additional perturbation u_i, conditioned on the current mini-batch S_i, comes from (i) the parameter vector at the beginning of the iteration θ, which itself comes from the random parameter initialization and the stochasticity of the parameter updates until that iteration, and (ii) the gradient vector ∇L_{S_i} (simply denoted by g_i), which may also be random because of the possible additional randomness in the network structure due for instance to dropout [Srivastava et al. 2014]. A common assumption made in the literature is that the random perturbation u_i follows a normal distribution [Bellido and Fiesler 1993; Neyshabur et al. 2017b]. The upper bound in Theorem 1 takes a particularly simple form if we assume that for $i \in \{1,2\}$, u_i are zero mean i.i.d. normal variables ($u_i \sim \mathcal{N}(0,\sigma^2 I)$), and that θ_i is fixed, as in the setting of [Dziugaite and Roy 2017].

As $\theta_i = \theta - \gamma g_i$ for $i \in \{1,2\}$, the KL-divergence between $Q_1 = \mathcal{N}(\theta_1, \sigma^2 I)$ and $Q_2 = \mathcal{N}(\theta_2, \sigma^2 I)$

S_1 and S_2 is independent from the dataset S. Hence, similarly to [Negrea et al. 2019; Dziugaite et al. 2020], we have $\sigma(S_1) \perp\!\!\!\perp \sigma(S_2)$.

(Lemma 1 in Section 3.B) is simply

$$\mathrm{KL}(Q_1||Q_2) = \frac{1}{2}\frac{\gamma^2}{\sigma^2}\left\|g_1 - g_2\right\|_2^2 = \mathrm{KL}(Q_2||Q_1), \tag{3.5}$$

which shows that, keeping a constant step size γ and assuming the same variance for the random perturbations σ^2 in all the steps of the training, the bound in Theorem 1 is driven by $\left\|g_1 - g_2\right\|_2$. This indicates that the smaller the ℓ_2 distance between gradient vectors is, the lower the upper bound on the generalization penalty is, and therefore the closer the performance of a model trained on one batch is to a model trained on another batch.

For two mini-batches of points S_i and S_j, with respective gradient vectors g_i and g_j, we define the *gradient disparity* (GD) between S_i and S_j as

$$\mathfrak{D}_{i,j} = \left\|g_i - g_j\right\|_2. \tag{3.6}$$

To compute $\mathfrak{D}_{i,j}$, a first option is to sample S_i from the training set and S_j from the held-out validation set, which we refer to as the “train-val" setting, following [Fort et al. 2019]. The generalization penalty $\mathscr{R}_j$ in this setting measures how much, during the course of an iteration, a model updated on a training set is able to generalize to a validation set, making the resulting (“train-val”) gradient disparity $\mathfrak{D}_{i,j}$ a natural candidate for tracking overfitting. But it requires access to a validation set to sample S_j, which we want to avoid. The second option is to sample both S_i and S_j from the training set, as proposed in this chapter, to yield now a value of $\mathfrak{D}_{i,j}$ that we could call “train-train" gradient disparity (GD) by analogy. Importantly, we observe a strong positive correlation between the two types of gradient disparities ($\rho = 0.957$) in Fig. 3.2. Therefore, we can expect that both of them do (almost) equally well in detecting overfitting, with the advantage that the latter does not require to set data aside, contrary to the former. We will therefore consider GD when both batches are sampled from the training set and evaluate it in this chapter.

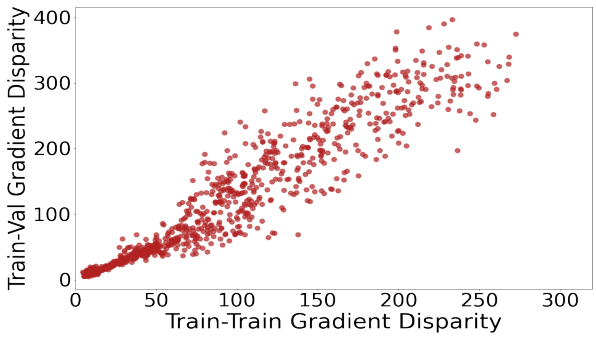


Figure 3.2: “Train-val" gradient disparity versus “train-train" gradient disparity for 220 experimental settings that vary in architecture, dataset, training set size, label noise level and initial random seed. Pearson’s correlation coefficient is $\rho = 0.957$.

To track the upper bound of the generalization penalty for more pairs of mini-batches, we can compute an average gradient disparity over B mini-batches, which requires all the B gradient vectors at each iteration, which is computationally expensive if B is large. We approximate it by computing GD over only a much smaller subset of the mini-batches, of size $s \ll B$,

$$\overline{\mathfrak{D}} = \sum_{i=1}^{s} \sum_{j=1, j\neq i}^{s} \frac{\mathfrak{D}_{i,j}}{s(s-1)}. \tag{3.7}$$

In our experiments, $s = 5$; we observe that such a small subset is already sufficient (see Section 3.E.2 for an experimental comparison of different values of s).

Consider two training iterations t_1 and t_2 where $t_1 \ll t_2$. At earlier stages of the training (iteration t_1), the parameter vector ($\theta^{(t_1)}$) is likely to be located in a steep region of the training loss landscape, where the gradient vector of training batches, g_i, and the training loss $L_{S_i}(f_{\theta^{(t_1)}})$ take large values. At later stages of training (iteration t_2), the parameter vector ($\theta^{(t_2)}$) is more likely in a flatter region of the training loss landscape where g_i and $L_{S_i}(f_{\theta^{(t_2)}})$ take small values. To compensate for this scale mismatch when comparing the distance between gradient vectors at different stages of training, we re-scale the loss values within each batch before computing $\overline{\mathfrak{D}}$ (see Section 3.E.1 for more details). Note that this re-scaling is only done for the purpose of using GD as a metric, and therefore does not have any effect on the training process itself.

We focus on the vanilla SGD optimizer. In Section 3.H, we extend the analysis to other stochastic optimization algorithms: SGD with momentum, Adagrad, Adadelta, and Adam. In all these optimizers, we observe that GD (Eq. (3.6)) appears in $\mathrm{KL}(Q_1||Q_2)$ with other factors that depend on a decaying average of past gradient vectors. Experimental results support the use of GD as an early stopping metric also for these popular optimizers (see Fig. 3.25 in Section 3.H). For vanilla SGD optimizer, we also provide an alternative and simpler derivation leading to gradient disparity from the linearization of the loss function in Section 3.D.

3.1.5 Early Stopping

In the presence of *large* amounts of *reliable* data, it is affordable to split the available dataset into a training and a validation set and to perform early stopping by evaluating the performance of the model on the held-out validation set. However, if the dataset is *limited*, this approach makes an inefficient use of the data because the model never learns the information that is still present in the validation set. Moreover, if the dataset is *noisy*, held-out validation might poorly estimate the performance of the model as the validation set might contain a high percentage of noisy samples. To avoid these issues, k-fold cross-validation [Stone 1974] is a solution that makes an efficient usage of the available data while providing an unbiased estimate of the performance, at the expense of a high computational overhead and of a possibly underestimated variance [Bengio and Grandvalet 2004]. While each of its k rounds is itself a setting with a held-out validation set, k-fold cross-validation (as opposed to held-out validation) would be therefore advantageous to use in the presence of limited and/or noisy

Setting	Method	Test loss	Test accuracy
CIFAR-10, VGG-13	5-fold CV	$1.846_{\pm 0.016}$	$35.982_{\pm 0.393}$
	GD	$\mathbf{1.793}_{\pm 0.016}$	$\mathbf{36.96}_{\pm 0.861}$
MNIST, AlexNet	5-fold CV	$1.123_{\pm 0.25}$	$62.62_{\pm 6.36}$
	GD	$\mathbf{0.656}_{\pm 0.080}$	$\mathbf{79.12}_{\pm 3.04}$

Table 3.3: Test loss and accuracy when using gradient disparity (GD) and k-fold cross-validation (CV) (k =5) as early stopping criteria when the available dataset is limited: (top) VGG-13 trained on 1.28 k samples of the CIFAR-10 dataset, and (bottom) AlexNet trained on 256 samples of the MNIST dataset. The corresponding curves during training are presented in Fig. 3.13. The results below are obtained by stopping the optimization when the metric (either validation loss or GD) has increased for 5 epochs from the beginning of training.

data. It extracts more information from the dataset as it uses all the data samples for both training and validation, and it is less dependent on how the data is split into training and validation sets.

The baseline to beat is therefore k-fold cross-validation (CV). We compare gradient disparity to CV in the two target settings: (i) when the available dataset is limited and (ii) when the available dataset has corrupted labels. Medical applications are one of the practical examples of setting (i), where datasets are costly because they require the collection of patient data, and the medical staff's expertise to label the data. An example of such an application is the MRNet dataset [Bien et al. 2018], which contains a limited number of MRI scans to study the presence of abnormally, ACL tears and meniscal tears in knee injuries. This dataset is by nature limited and we use the entire available data for both early stopping methods GD and k-fold CV. In addition, to further simulate setting (i), we use small subsets of three image classification benchmark datasets: MNIST, CIFAR-10 and CIFAR-100. Performing early stopping in the presence of label noise (setting (ii)) is also practically very important, because it has been empirically observed that deep neural networks trained on noisy datasets overfit to noisy labeled samples at later stages of training. A good early stopping signal can therefore prevent such an overfitting [Li et al. 2020c; Song et al. 2020a; Xia et al. 2021]. To simulate setting (ii), we use a corrupted version of these image classification benchmark datasets, where for a fraction of the samples (the amount of noise), we choose the labels at random.

(i) We observe that using gradient disparity instead of a validation loss in k-fold CV results in an improvement of more than 1% (on average over all three tasks) in the test AUC score of the MRNet dataset, and therefore adds a correct detection for more than one patient for each task (see Table 3.1). Furthermore, we observe that gradient disparity performs better than k-fold CV as an early stopping criterion for image-classification benchmark datasets as well (see Table 3.3). A plausible explanation for the better peformance of GD over k-fold CV is that, although CV uses the entire set of samples over the k rounds for both training and validation, it trains the model only on a $(1-1/k)$ portion of the dataset in each individual

round. In contrast, GD allows to train the model over the entire dataset in a single run, which therefore results in a better performance on the final unseen (test) data when data is limited. For more experimental results refer to Table 3.13 and Fig. 3.13 and Fig. 3.15 in Section 3.F.

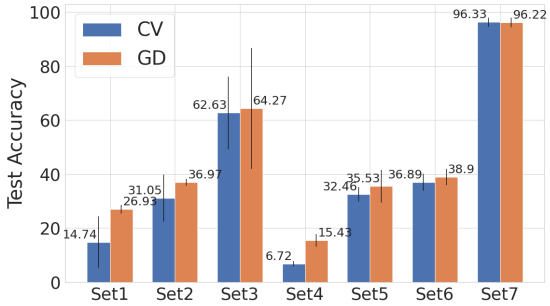
(ii) We observe that gradient disparity performs better than k-fold cross-validation as an early stopping criterion when data is noisy (see Table 3.4). When the labels of the available data are noisy, the validation set is no longer a reliable estimate of the test set. Nevertheless, and although it is computed over the noisy training set, gradient disparity reflects the performance on the test set quite well[4] For more experimental results refer to Table 3.14 and Fig. 3.14 in Section 3.F.

Quite surprisingly, we observe that GD performs better in terms of accuracy than an extension of k-fold CV, which we call k^+-fold CV, which uses the entire dataset for training with the early stopping signal found by k-fold CV (see Table 3.4, where $k = 10$ for these settings). More precisely, k^+-fold CV is done in 3 steps: (1) perform k-fold CV, (2) compute the stopping epoch by tracking the validation loss found in step (1), and (3) retrain the model on the entire dataset and stop at the epoch obtained in step (2). k^+-fold CV uses therefore $k+1$ rounds because of step (3), thus one more round than k-fold CV, but unlike k-fold CV (and similarly to GD), k^+-fold CV produces models that are trained on the entire dataset. It is therefore interesting to note that using GD still outperforms k^+-fold CV in terms of accuracy (although not in terms of loss).

Setting	Method	Test loss	Test accuracy
CIFAR-100, ResNet-18	10-fold CV	$5.023_{\pm 0.083}$	$1.59_{\pm 0.15}$ (top-5: $6.47_{\pm 0.52}$)
	GD	$\mathbf{4.463}_{\pm 0.038}$	$\mathbf{3.68}_{\pm 0.52}$ (top-5: $\mathbf{15.22}_{\pm 1.24}$)
	10^+-fold CV	$4.964_{\pm 0.057}$	$1.68_{\pm 0.24}$ (top-5: $7.05_{\pm 0.71}$)
MNIST, AlexNet	10-fold CV	$0.656_{\pm 0.034}$	$97.28_{\pm 0.20}$
	GD	$0.654_{\pm 0.031}$	$\mathbf{97.32}_{\pm 0.27}$
	10^+-fold CV	$\mathbf{0.639}_{\pm 0.029}$	$97.31_{\pm 0.15}$

Table 3.4: Test loss and accuracy when using gradient disparity (GD) and k-fold cross-validation (CV) (k =10) as early stopping criteria when the available dataset is noisy: 50% of the available data has random labels. The corresponding curves during training are shown in Fig. 3.14. The results above are obtained by stopping the optimization when the metric (either validation loss or GD) has increased for 5 epochs from the beginning of training. The last row in each setting, which we call 10^+-fold CV, refers to the test loss and accuracy reached at the epoch suggested by 10-fold CV, for a network trained on the entire set. Notice that for the CIFAR-100 experiments (the top rows), for computational reasons, the models are trained on only 1.28 k samples of the dataset which explains the very low test accuracy for this experiment. However, for the MNIST experiments (the bottom rows), the models are trained on the entire dataset, and we observe rather high test accuracies.

[4] See for example Fig. 3.14 (left column) where the validation loss fails to estimate the test loss, but where GD (Fig. 3.14 (middle left column)) does signal overfitting correctly.



(a)

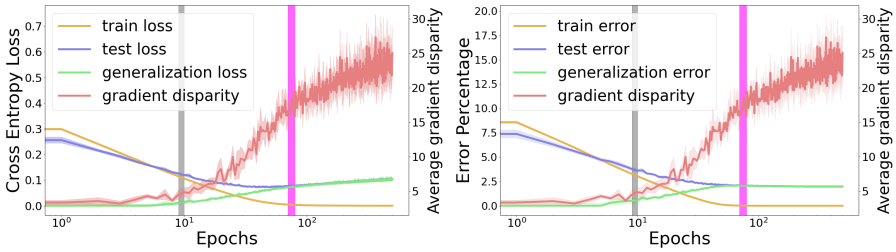
Figure 3.3: Test Accuracy achieved by using GD and k-fold CV as early stopping methods in 7 experimental settings (indicated in the x-axis by Set1-7). The result is averaged over 20 choices of the early stopping threshold. The complete set of results are reported in Table 3.10 and Table 3.11. For the CIFAR-100 experiments (Set1, 4, and 5) the top-5 accuracy is reported.

The metrics used as early stopping criteria, whether they are the validation loss or gradient disparity, are measured on signals that are subject to random fluctuations. As a result, they rely on a pre-defined threshold p (sometimes called *patience* by practitioners) that sets the number of iterations during which the metric increases before the algorithm is stopped. We use two popular thresholds: (t1) the first one is to stop the algorithm when the metric (GD or validation loss) has increased for $p = 5$ (possibly non consecutive) epochs from the beginning of training, and (t2) the second is the same as (t1) but the $p = 5$ epochs must be consecutive. Both GD and k-fold CV might be sensitive to the choice of (t1) or (t2), or even to the value of p itself. It is therefore important to study the sensitivity of an early stopping metric to the choice of the threshold p, which is done in Section 3.F.1 for both GD and k-fold CV for ten different values of $p \in \{1, \cdots, 10\}$ and the two thresholds (t1) and (t2). We observe that GD always gives similar or higher test accuracy than k-fold CV for all 20 possible thresholds (see Fig. 3.3). More importantly, GD is much more robust to the choice of the early stopping threshold (see Table 3.5).

Method	Sensitivity of the Test Accuracy	Sensitivity of the Test Loss
GD	**0.916**	**0.886**
CV	1.613	1.019

Table 3.5: Sensitivity of each method to the choice of the early stopping threshold. The sensitivity is computed from the reported values of Table 3.10 and Table 3.11 according to Eq. (3.16) in the appendix.

When data is abundant and clean, the validation loss is affordable and trustworthy to use as an early stopping signal. GD does also correctly signal overfitting in this case (see for example Fig. 3.4). However, when data is limited and/or noisy (which is also when early stopping is particularly important), we observe that the validation loss is costly and unreliable to use as an early stopping signal. In contrast, in these settings, GD does not cost a separate



(a) Loss vs gradient disparity

(b) Error vs gradient disparity

Figure 3.4: The cross entropy loss, error percentage, and average gradient disparity during training for a 4-layer fully connected neural network with 500 hidden units trained on the entire MNIST dataset with 0% label noise. The parameter initialization is the He initialization with normal distribution. Pearson's correlation coefficient ρ between $\overline{\mathfrak{D}}$ and generalization loss/error over all the training iterations are $\rho_{\overline{\mathfrak{D}},\text{gen loss}} = 0.967$ and $\rho_{\overline{\mathfrak{D}},\text{gen error}} = 0.734$. The gray vertical bar indicates when GD increases for 5 epochs from the beginning of training. The magenta vertical bar indicates when GD increases for 5 *consecutive* epochs. We observe that the gray bar signals when overfitting is starting, which is when the training and testing curves are starting to diverge. The magenta bar would be a good stopping time, because if we train beyond this point, although the test error remains the same, the test loss would increase, which would result in overconfidence on wrong predictions.

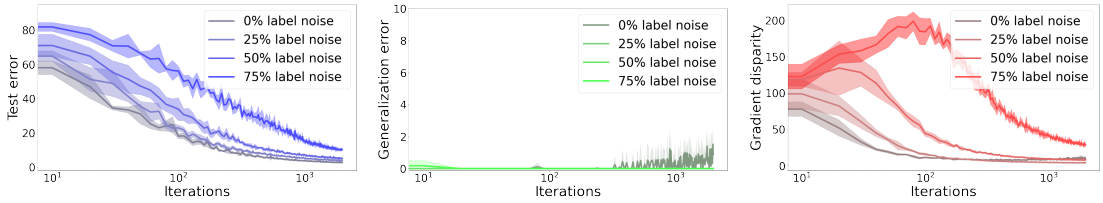
held-out validation set and is a reliable signal of overfitting even in the presence of label noise. In practice, the label noise level of a given dataset is in general not known a priori and we do not know whether the size of the dataset is large enough to afford sacrificing a subset for validation. We often do not know whether we are in the former setting, with abundant and clean data, or in the later setting, with limited and/or noisy data. It is therefore important to have a good early stopping criterion that works for both settings. Unlike the validation loss, GD is such a signal.

3.1.6 Discussion and Final Remarks

We propose gradient disparity (GD), as a simple to compute early stopping criterion that is particularly well-suited when the dataset is limited and/or noisy. Beyond indicating the early stopping time, GD is well aligned with factors that contribute to improve or degrade the generalization performance of a model, which have an often strikingly similar effect on the value of GD as well. We briefly discuss in this section some of these observations that further validate the use of GD as an effective early stopping criterion; more details are provided in the appendix.

Label Noise Level. We observe that GD reflects well the label noise level throughout the training process, even at early stages of training, where the generalization gap fails to do so (see Fig. 3.5, and Fig. 3.17, Fig. 3.21, and Fig. 3.24 in Section 3.G).

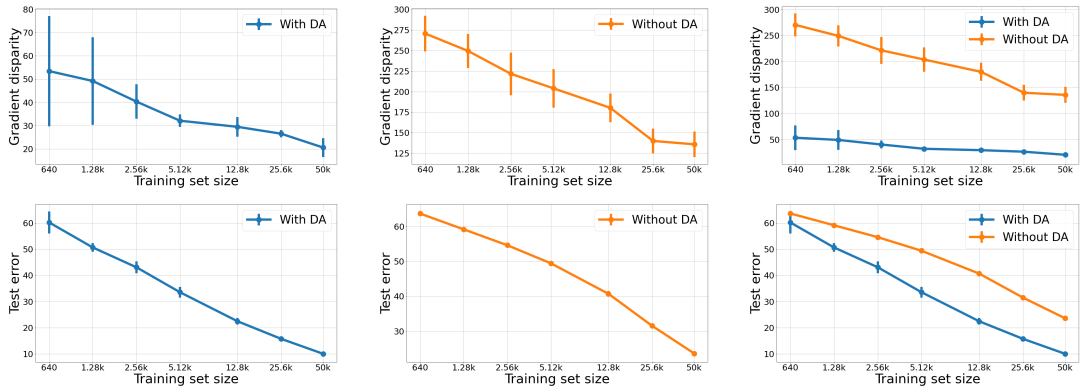
Training Set Size. We observe that GD, similarly to the test error, decreases with training



(a) Test error (b) Generalization error (c) $\overline{\mathfrak{D}}$

Figure 3.5: The error percentage and $\overline{\mathfrak{D}}$ during training for an AlexNet trained on a subset of 12.8 k points of the MNIST training dataset with different amounts of label noise. Pearson's correlation coefficient between gradient disparity and test error (TE)/test loss (TL) over all iterations and label noise levels are $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.861$ and $\rho_{\overline{\mathfrak{D}},\mathrm{TL}} = 0.802$. The generalization error (gap) is the difference between the train and test errors and for this experiment fails to account for the label noise level, unlike gradient disparity.

set size, unlike many previous metrics as shown by [Neyshabur et al. 2017a; Nagarajan and Kolter 2019]. Moreover, we observe that applying data augmentation decreases the values of both GD and the test error (see Fig. 3.6 and Fig. 3.22 in Section 3.G).



(a) $\rho_{\overline{\mathscr{D}},\mathrm{TE}} = 0.979$ (b) $\rho_{\overline{\mathscr{D}},\mathrm{TE}} = 0.984$ (c) $\rho_{\overline{\mathscr{D}},\mathrm{TE}} = 0.619$

Figure 3.6: Test error (TE) (bottom row) and gradient disparity ($\overline{\mathfrak{D}}$) (top row) for a ResNet-18 trained on the CIFAR-10 dataset with different training set sizes. (a) Results with data augmentation (DA) . (b) Results without using any data augmentation technique. (c) Combined results of (a) and (b). We observe a strong positive correlation between $\overline{\mathfrak{D}}$ and TE regardless of using data augmentation. We also observe that using data augmentation decreases the values of both gradient disparity and the test error.

Batch Size. We observe that both the test error and GD increase with batch size. This observation is counter-intuitive because one might expect that gradient vectors get more similar when they are averaged over a larger batch. GD matches the ranking of test errors for different networks, trained with different batch sizes, as long as the batch sizes are not too large (see Fig. 3.7 and Fig. 3.23 in Section 3.G).

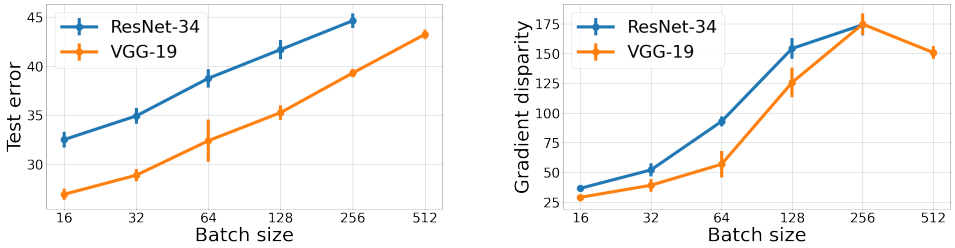


Figure 3.7: Test error and gradient disparity for networks that are trained with different batch sizes trained on 12.8 k points of the CIFAR-10 dataset. The training is stopped when the training loss is below 0.01. The correlation between normalized gradient disparity and test error is $\rho_{\tilde{\mathfrak{D}},\mathrm{TE}} = 0.893$.

Width. We observe that both the test error and GD (normalized with respect to the number of parameters) decrease with the network width for ResNet, VGG and fully connected neural networks (see Fig. 3.8 and Fig. 3.20 in Section 3.G).

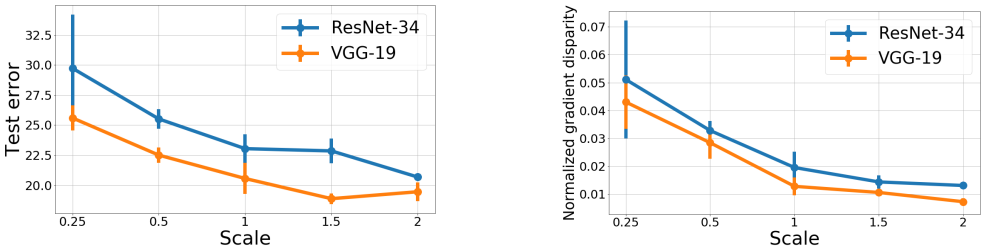


Figure 3.8: Test error and normalized gradient disparity for networks trained on the CIFAR-10 dataset with different number of channels and hidden units for convolutional neural networks (CNN) (scale = 1 recovers the original configurations). The correlation between normalized gradient disparity and test loss $\rho_{\tilde{\mathfrak{D}},\mathrm{TL}}$ and between normalized gradient disparity and test error $\rho_{\tilde{\mathfrak{D}},\mathrm{TE}}$ are $\rho_{\tilde{\mathfrak{D}},\mathrm{TL}} = 0.970$ and $\rho_{\tilde{\mathfrak{D}},\mathrm{TE}} = 0.939$, respectively.

Gradient disparity belongs to the same class of metrics based on the similarity between two gradient vectors [Sankararaman et al. 2019; Fort et al. 2019; Fu et al. 2020; Mehta et al. 2020; Jastrzebski et al. 2020]. A common drawback of all these metrics is that they are not informative when the gradient vectors are very small. In practice however, we observe (see for instance Fig. 3.18 in the appendix) that the time at which the test and training losses start to diverge, which is the time when overfitting kicks in, does not only coincide with the time at which gradient disparity increases, but also occurs much before the training loss becomes infinitesimal. This drawback is therefore unlikely to cause a problem for gradient disparity when it is used as an early stopping criterion. Nevertheless, as a future direction, it would be interesting to explore this further especially for scenarios such as epoch-wise double-descent [Heckel and Yilmaz 2020].

3.2 Time-series Applications

This section aims to address the challenge of cross-validation in time-series applications, where the standard approach of random-shuffling over the data may result in imbalanced subsets that are dependent on each other. We propose to investigate alternative methods to cross-validation in this context. Specifically, we will explore the potential benefits of using gradient disparity, which was introduced in Section 3.1, as an early stopping criterion that does not require a held-out validation set. Our study aims to provide insights into the efficacy of gradient disparity as a replacement for cross-validation in time-series applications.

3.2.1 Introduction

The aim of neural network training is to achieve optimal generalization performance, but neural networks are prone to overfitting, where the error on unseen observations increases despite decreasing training error [Geman et al. 1992]. To avoid overfitting in deep neural networks trained with iterative methods like gradient descent, early stopping using a separate validation set is a popular technique [Gu et al. 2018; Prechelt 1998]. Early stopping involves stopping optimization when overfitting is detected, typically by observing an increase in the validation set error, estimated by the average error on a validation set not used for training. Choosing a suitable performance estimation method for neural networks depends on the nature of the data used. Cross-validation is commonly used when the data is assumed to be independent and identically distributed, due to its efficient use of data [Arlot and Celisse 2010]. However, cross-validation may not be suitable for dependent data, such as time series, as the validation set is often drawn randomly, which can lead to evaluating past observations in the validation set using future observations in the training set.

A possible alternative to cross-validation for time series data is gradient disparity, an early stopping criterion introduced in Section 3.1 that does not require a held-out validation set. Recall that gradient disparity is the l_2 norm distance between the gradient vectors of two mini-batches drawn from the training set. In this study, we aim to compare the performance of gradient disparity and cross-validation in evaluating the generalization ability of neural networks. We will evaluate multiple variants of cross-validation adapted for time series data and compare their performance on different datasets.

3.2.2 Early-stopping Methods for Time-series Applications

In this section, we introduce a few cross-validation approaches that we use as baselines to compare with gradient disparity in terms of early stopping methods for time-series applications.

k-fold Cross-Validation In the k-fold cross-validation (CV) technique, the original dataset is first divided randomly into k equal-sized subsets. Out of the k subsets, one subset is set

aside and used as the validation data to test the model, while the remaining $k-1$ subsets are used as training data. This process is repeated k times, with each of the k subsets used exactly once as the validation data. For datasets with independent samples, this cross-validation technique makes the optimal use of the data as all samples are used in both training and validation, hence this was the main baseline technique we used in Section 3.1.

Leave One Block Out Cross-Validation (LOBO CV) In LOBO cross-validation, one or more contiguous blocks of data are held out as the validation set while the rest of the data is used for training. The validation set consists of a single block of data, which is left out of the training set in each iteration. This process is repeated for each block of data in the dataset. The advantage of LOBO cross-validation compared to k-fold CV is that it preserves the temporal or spatial order of the data.

Hv-Blocked Cross-Validation The main idea behind this technique is similar to that of Leave-One-Block-Out (LOBO) cross-validation, but with an additional step of removing dependent values between the validation and training data. In the case of time series data, values that are next to each other can often be highly dependent, and including them in both the validation and training sets may lead to biased evaluations of the model's performance.

Temporal Block Cross-Validation In time series data, the temporal order of observations is crucial, and ignoring it can result in poor performance of the model. In order to preserve the temporal aspect of time series, temporal block cross-validation is used. Temporal block is similar to k-fold cross-validation but instead of randomly partitioning the data, it splits the available sample into k equal-sized subsets. In the first fold, the first subset is used as the training set and the second subset is used as the validation set, with all subsequent subsets being discarded. For the second fold, the third subset is used as the validation set and the first and the second subsets are used as the training set, with all subsequent subsets being discarded. This process is repeated until all the subsets have been used as the validation set once. In each fold, the validation set is contiguous and comes after the training set in the temporal order of the data. This ensures that the validation set is always composed of observations that come after the training set, which is essential for time series data. Although temporal block CV preserves the temporal aspect of time series, it comes at the cost of discarding most of the data for each fold and not being able to use all the available data for training. However, using different parts of the data as the validation set makes temporal block CV more robust than other out-of-sample evaluation methods. Out-of-sample evaluation (OOS) refers to holding out the last part of the time series for validation and using the rest for training, which is done only once over the whole dataset. Temporal block CV is the cross-validation equivalent of OOS.

Gradient Disparity We observed that traditional cross-validation (CV) methods have limitations in preserving the temporal ordering of time series data and require discarding parts of the available data to achieve this. Additionally, these methods assume independence of data which may not be true for time series. To overcome these limitations, we are interested

in studying the potential benefits of gradient disparity (GD) as an early-stopping criterion for time series data. Recall the introduction of GD in Eq. (3.7). In Section 3.1 it was extensively shown that GD is a useful criterion for early stopping. In this study, we are interested in exploring the applicability of GD to time series data, particularly its ability to use all available observations for training without the need for data partitioning. Additionally, we aim to compare the performance of GD to traditional CV methods in signaling overfitting in time series data.

Early-Stopping Threshold After selecting an early stopping metric, the next step is to define a threshold to apply on top of the metric. The two early stopping metrics that are studied in this section are gradient disparity and validation loss, and because both are negatively correlated with the test accuracy, an increase in their value is a sign of overfitting. In this study, we explore four different early stopping thresholds to ensure the generalizability of our results. The default reported values are based on stopping training when there are p consecutive or nonconsecutive increases in the value of the early stopping metric (either the validation loss in CV or the value of gradient disparity) from the previous one. The value of p is referred to as the patience parameter and is set to 5 by default. Another early stopping threshold we found useful in this study, especially when the results had a lot of variations, is the following: Stop training when a defined variable that starts at 0 reaches the value p; the variable would increase by 1 if the early stopping metric increased from the previous one and decreased by 0.5 otherwise.

3.2.3 Experiments

We conducted our experiments using the datasets and data preprocessing approach presented by Lara-Benítez et al. [2021]. Specifically, we worked with the following datasets:

- The CIF 2016 competition dataset [Štěpnička and Burda 2016], which includes 72 monthly time series. Of these, 12 have a 6-month forecasting horizon and the remaining 57 have a 12-month forecasting horizon. The time series consist of bank risk analysis indicators, some of which are real and others that are artificially generated.

- The NN5 competition dataset [NNGC 2008], which comprises 111 time series, each with 735 values. This dataset captures two years of daily cash withdrawals at automatic teller machines (ATMs) in England. The competition required forecasting 56 days ahead.

- The WikiWebTraffic dataset, which was part of a Kaggle competition [Google 2017] aimed at predicting the future web traffic for a set of Wikipedia pages. The dataset contains daily counts of page visits, and the forecasting horizon is set at 59 days.

In order to assess the predictive performance of our models, we utilized the weighted

absolute percentage error (WAPE), which is defined as

$$\text{WAPE}(y, o) = \frac{\text{mean}(|y - o|)}{\text{mean}(y)}, \tag{3.8}$$

where y and o represent the real and predicted values, respectively. WAPE scales the error by dividing it by the mean, allowing for comparison across time series with varying scales. This metric is a variation of the mean absolute percentage error (MAPE), which is a widely used measure of forecast accuracy due to its advantages of being scale-independent and interpretable [Kim and Kim 2016].

Results on the CIF 2016 dataset: When examining the performance of various methods, as shown in Table 3.6, we can observe that training on the complete dataset and using gradient disparity (GD) as an early stopping criterion results in the lowest error, while temporal block cross-validation (CV) was the most effective CV method. The superiority of temporal block CV may be due to its preservation of the temporal order of the time series. 5-Fold and leave-one-block-out (LOBO) CV showed similar outcomes, possibly because they do not take into account temporal ordering but still use all available observations. The least satisfactory outcome was obtained with Hv-Blocked, which may be due to the fact that it discards a large amount of data to maintain the independence of the validation and training sets, resulting in significant loss of information for a small dataset. Note that for this dataset, because of varying lengths of the time series in different samples, we combined all samples and treated them as a single large time-series to perform cross-validation.

Method	WAPE	Epoch
5-fold	0.231	10
LOBO	0.239	9
Hv-blocked	0.259	8
Temporal block	0.212	12
Gradient disparity	**0.181**	8

Table 3.6: Test WAPE results and stopping epoch for different early stopping methods applied to networks trained on the CIF 2016 dataset. We observe that training on the complete dataset using Gradient Disparity yields the lowest error, while temporal block CV is the most effective CV method.

Results on the NN5 and WikiWebTraffic datasets: To emphasize the importance of using gradient disparity with limited data, we selected a subset of the original NN5 and WikiWeb-Traffic datasets containing only 10 time series. Since all samples had the same time-series length, we were able to apply adaptive cross-validation by performing cross-validation on each individual time series. Unlike the CIF dataset, we did not combine all samples to make them a single large time-series. After conducting a comprehensive ablation study on the early stopping threshold for each of the early stopping metrics, we selected the best WAPE result

according to each method and reported the results in Table 3.7 and Table 3.8, for datasets NN5 and WikiWebTraffic, respectively. It can be observed that training on the entire dataset and using gradient disparity as an early stopping criterion results in the best performance.

Method	WAPE	Epoch
5-fold	0.167	41
LOBO	0.166	16
Hv-blocked	0.176	16
Temporal block	0.204	42
Gradient disparity	**0.164**	23

Table 3.7: Test WAPE results and stopping epoch for different early stopping methods applied to networks trained on the NN5 dataset. The optimum early stopping threshold is chosen for each of the early stopping methods. We observe that training on the complete dataset using Gradient Disparity yields the lowest error, while LOBO CV is the most effective CV method.

Method	WAPE	Epoch
5-fold	0.539	15
LOBO	0.558	17
Hv-blocked	0.589	28
Temporal block	0.573	42
Gradient disparity	**0.514**	19

Table 3.8: Test WAPE results and stopping epoch for different early stopping methods applied to networks trained on the WikiWebTraffic dataset. The optimum early stopping threshold is chosen for each of the early stopping methods. We observe that training on the complete dataset using Gradient Disparity yields the lowest error, while LOBO CV is the most effective CV method.

3.2.4 Conclusion

In this project, we explored the use of gradient disparity as an early stopping criterion for time series forecasting, compared to four different cross-validation methods. We tested our approach on the CIF 2016, NN5, and WikiWebTraffic datasets, and evaluated the performance of each approach by measuring the weighted absolute percentage error (WAPE). Our results showed that gradient disparity outperforms other cross-validation methods on the time-series datasets as an early stopping signal. Gradient disparity detects the optimal early stopping iteration and enables the usage of the entire available dataset for training, which decreases the achieved error of the trained model especially when the available time-series dataset is limited in size.

We also experimented with different early stopping thresholds and found that the best threshold for detecting overfitting was when we stopped training when a defined variable reached a value of p, which was increased by 1 if the early stopping metric increased from the previous one and decreased by 0.5 if it decreased from the previous one.

Our study suggests that gradient disparity is a promising approach for early stopping in time series forecasting, as it was able to detect overfitting accurately and outperformed other cross-validation methods on small datasets. For large datasets, gradient disparity performs comparable to some other cross-validation methods. In future work, it would be interesting to explore the use of gradient disparity in other domains and test its robustness on different types of data. Additionally, it would be beneficial to investigate the use of gradient disparity in conjunction with other early stopping methods to see if the combination can lead to further improvements in performance.

Appendices

3.A Organization of the Appendix

This appendix includes sections that are provided here for the sake of completeness and reproducibility. It is structured as follows.

- Section 3.C gives the proof of Theorem 1, which also uses Hoeffding's bound recalled in Section 3.B.
- Section 3.D provides a simple relation between gradient disparity and generalization penalty from linearization.
- A number of details common to all experiments are provided in Section 3.E, which discusses in particular how the loss is re-scaled before computing gradient disparity (Section 3.E.1) and how gradient disparity can also be applied to networks trained with the mean square error (Section 3.E.3).
- A detailed comparison of gradient disparity to k-fold cross validation as early stopping criteria is given in Section 3.F, which includes a study on the robustness to the early stopping threshold in Section 3.F.1 (Table 3.10, Table 3.11 and Table 3.12) and additional experiments on four image classification datasets (Fig. 3.13, Fig. 3.14, and Fig. 3.15 together with Table 3.13, and Table 3.14).
- Additional experiments on benchmark datasets are provided in Section 3.G to study the effect of label noise level, training set size, batch size and network width on the value of gradient disparity. The results, which support the claims made in the main chapter, are displayed in Fig. 3.5 to Fig. 3.24.
- Besides the vanilla SGD algorithm adopted in the main chapter, gradient disparity can be extended to other stochastic optimization algorithms (SGD with momentum, Adagrad, Adadelta, and Adam) as shown in Section 3.H.
- Finally, a detailed comparison with related work is presented in Section 3.I (Table 3.15 and Table 3.16 together with Fig. 3.26 and Fig. 3.27).

3.B Additional Theorem

Hoeffding's bound is used in the proof of Theorem 1, and Lemma 1 is used in Section 3.1.4.

Theorem 2 (Hoeffding's Bound)**.** *Let $Z_1, \cdots, Z_n$ be independent bounded random variables on $[a,b]$ (i.e., $Z_i \in [a,b]$ for all $1 \leq i \leq n$ with $-\infty < a \leq b < \infty$). Then*

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \geq t\right) \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right)$$

and

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}(Z_i - \mathbb{E}[Z_i]) \leq -t\right) \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right)$$

for all $t \geq 0$.

Lemma 1 If $N_1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $N_2 = \mathcal{N}(\mu_2, \Sigma_2)$ are two multivariate normal distributions in $\mathbb{R}^d$, where Σ_1 and Σ_2 are positive definite,

$$\mathrm{KL}(N_1||N_2) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) - d + (\mu_2 - \mu_1)^T\Sigma_2^{-1}(\mu_2 - \mu_1) + \ln\left(\frac{\det\Sigma_2}{\det\Sigma_1}\right)\right).$$

3.C Proof of Theorem 1

Proof. We compute the upper bound in Eq. (3.4) using a similar approach as in [McAllester 2003]. The main challenge in the proof is the definition of a function X_{S_2} of the variables and parameters of the problem, which can then be bounded using similar techniques as in [McAllester 2003]. S_1 is a batch of points (with size n_1) that is randomly drawn from the available set S at the beginning of iteration t, and S_2 is a batch of points (with size n_2) that is randomly drawn from the remaining set $S \setminus S_1$. Hence, S_1 and S_2 are drawn from the set S without replacement ($S_1 \cap S_2 = \emptyset$). Similar to the setting of [Negrea et al. 2019; Dziugaite et al. 2020], as the random selection of indices of S_1 and S_2 is independent from the dataset S, $\sigma(S_1) \perp\!\!\!\perp \sigma(S_2)$, and as a result, $\mathscr{G}_1 \perp\!\!\!\perp \sigma(S_2)$ and $\mathscr{G}_2 \perp\!\!\!\perp \sigma(S_1)$. Recall that ν_i is the random parameter vector at the end of iteration t that depends on S_i, for $i \in \{1,2\}$. For a given sample set S_i, denote the conditional probability distribution of ν_i by Q_{S_i}. For ease of notation, we represent Q_{S_i} by Q_i.

Let us denote

$$\Delta\left(f_{\nu_1}, f_{\nu_2}\right) \triangleq \left(L_{S_2}(f_{\nu_1}) - L(f_{\nu_1})\right) - \left(L_{S_2}(f_{\nu_2}) - L(f_{\nu_2})\right), \tag{3.9}$$

and

$$X_{S_2} \triangleq \sup_{Q_1,Q_2}\left(\frac{n_2}{2} - 1\right)\mathbb{E}_{\nu_1\sim Q_1}\left[\mathbb{E}_{\nu_2\sim Q_2}\left[\left(\Delta\left(f_{\nu_1}, f_{\nu_2}\right)\right)^2\right]\right] - \mathrm{KL}(Q_2||Q_1). \tag{3.10}$$

Note that X_{S_2} is a random function of the batch S_2. Expanding the KL-divergence, we find that

$$\begin{aligned}
&\left(\frac{n_2}{2}-1\right)\mathbb{E}_{\nu_1\sim Q_1}\left[\mathbb{E}_{\nu_2\sim Q_2}\left[\left(\Delta\left(f_{\nu_1},f_{\nu_2}\right)\right)^2\right]\right]-\mathrm{KL}(Q_2||Q_1)\\
&\quad=\mathbb{E}_{\nu_1\sim Q_1}\left[\left(\frac{n_2}{2}-1\right)\mathbb{E}_{\nu_2\sim Q_2}\left[\left(\Delta\left(f_{\nu_1},f_{\nu_2}\right)\right)^2\right]+\mathbb{E}_{\nu_2\sim Q_2}\left[\ln\frac{Q_1(\nu_2)}{Q_2(\nu_2)}\right]\right]\\
&\quad\le\mathbb{E}_{\nu_1\sim Q_1}\left[\ln\mathbb{E}_{\nu_2\sim Q_2}\left[e^{(\frac{n_2}{2}-1)(\Delta(f_{\nu_1},f_{\nu_2}))^2}\frac{Q_1(\nu_2)}{Q_2(\nu_2)}\right]\right]\\
&\quad=\mathbb{E}_{\nu_1\sim Q_1}\left[\ln\mathbb{E}_{\nu_1'\sim Q_1}\left[e^{(\frac{n_2}{2}-1)\left(\Delta\left(f_{\nu_1},f_{\nu_1'}\right)\right)^2}\right]\right],
\end{aligned}$$

where the inequality above follows from Jensen's inequality as logarithm is a concave function. Therefore, again by applying Jensen's inequality

$$X_{S_2}\le\ln\mathbb{E}_{\nu_1\sim Q_1}\mathbb{E}_{\nu_1'\sim Q_1}\left[e^{(\frac{n_2}{2}-1)\left(\Delta(f_{\nu_1},f_{\nu_1'})\right)^2}\right].$$

Taking expectations over S_2, we have that

$$\begin{aligned}
\mathbb{E}_{S_2}\left[e^{X_{S_2}}\right]&\le\mathbb{E}_{S_2}\mathbb{E}_{\nu_1\sim Q_1}\mathbb{E}_{\nu_1'\sim Q_1}\left[e^{(\frac{n_2}{2}-1)\left(\Delta(f_{\nu_1},f_{\nu_1'})\right)^2}\right]\\
&=\mathbb{E}_{\nu_1\sim Q_1}\mathbb{E}_{\nu_1'\sim Q_1}\mathbb{E}_{S_2}\left[e^{(\frac{n_2}{2}-1)\left(\Delta(f_{\nu_1},f_{\nu_1'})\right)^2}\right],
\end{aligned}\tag{3.11}$$

where the change of order in the expectation follows from the independence of the draw of the set S_2 from $\nu_1\sim Q_1$ and $\nu_1'\sim Q_1$, i.e., Q_1 is $\mathscr{G}_1$-measurable and $\mathscr{G}_1\perp\!\!\!\perp\sigma(S_2)$.

Now let

$$Z_i\triangleq l(f_{\nu_1}(x_i),y_i)-l(f_{\nu_1'}(x_i),y_i),$$

for all $1\le i\le n_2$. Clearly, $Z_i\in[-1,1]$ and because of Eqs. (3.2) and of the definition of Δ in Eq. (3.9),

$$\Delta\left(f_{\nu_1},f_{\nu_1'}\right)=\frac{1}{n_2}\sum_{i=1}^{n_2}\left(Z_i-\mathbb{E}[Z_i]\right).$$

Hoeffding's bound (Theorem 2) implies therefore that for any $t\ge0$,

$$\mathbb{P}_{S_2}\left(|\Delta\left(f_{\nu_1},f_{\nu_1'}\right)|\ge t\right)\le2e^{-\frac{n_2}{2}t^2}.\tag{3.12}$$

Denoting by $p(\Delta)$ the probability density function of $|\Delta\left(f_{\nu_1},f_{\nu_1'}\right)|$, inequality (3.12) implies that for any $t\ge0$,

$$\int_t^\infty p(\Delta)d\Delta\le2e^{-\frac{n_2}{2}t^2}.\tag{3.13}$$

The density $\tilde{p}(\Delta)$ that maximizes $\int_0^\infty e^{(\frac{n_2}{2}-1)\Delta^2}p(\Delta)d\Delta$ (the term in the first expectation of the upper bound of Eq. (3.11)), is the density achieving equality in Eq. (3.13), which is

$\tilde{p}(\Delta) = 2n_2\Delta e^{-\frac{n_2}{2}\Delta^2}$. As a result,

$$\begin{aligned}\mathbb{E}_{S_2}\left[e^{(\frac{n_2}{2}-1)\Delta^2}\right] &\leq \int_0^\infty e^{(\frac{n_2}{2}-1)\Delta^2} 2n_2\Delta e^{-\frac{n_2}{2}\Delta^2} d\Delta \\ &= \int_0^\infty 2n_2\Delta e^{-\Delta^2} d\Delta = n_2\end{aligned}$$

and consequently, inequality (3.11) becomes

$$\mathbb{E}_{S_2}\left[e^{X_{S_2}}\right] \leq n_2.$$

Applying Markov's inequality on X_{S_2}, we have therefore that for any $0 < \delta \leq 1$,

$$\mathbb{P}_{S_2}\left[X_{S_2} \geq \ln\frac{2n_2}{\delta}\right] = \mathbb{P}_{S_2}\left[e^{X_{S_2}} \geq \frac{2n_2}{\delta}\right] \leq \frac{\delta}{2n_2}\mathbb{E}_{S_2}\left[e^{X_{S_2}}\right] \leq \frac{\delta}{2}.$$

Replacing X_{S_2} by its expression defined in Eq. (3.10), the previous inequality shows that with probability at least $1-\delta/2$

$$\left(\frac{n_2}{2}-1\right)\mathbb{E}_{v_1\sim Q_1}\mathbb{E}_{v_2\sim Q_2}\left[\left(\Delta(f_{v_1}, f_{v_2})\right)^2\right] - \mathrm{KL}(Q_2||Q_1) \leq \ln\frac{2n_2}{\delta}.$$

Using Jensen's inequality and the convexity of $\left(\Delta(f_{v_1}, f_{v_2})\right)^2$, and assuming that $n_2 > 2$, we therefore have that with probability at least $1-\delta/2$,

$$\begin{aligned}\left(\mathbb{E}_{v_1\sim Q_1}\mathbb{E}_{v_2\sim Q_2}[\Delta\left(f_{v_1}, f_{v_2}\right)]\right)^2 &\leq \mathbb{E}_{v_1\sim Q_1}\mathbb{E}_{v_2\sim Q_2}\left[\left(\Delta\left(f_{v_1}, f_{v_2}\right)\right)^2\right] \\ &\leq \frac{\mathrm{KL}(Q_2||Q_1) + \ln\frac{2n_2}{\delta}}{\frac{n_2}{2}-1}.\end{aligned}$$

Replacing $\Delta(f_{v_1}, f_{v_2})$ by its expression Eq. (3.9) in the above inequality, yields that with probability at least $1-\delta/2$ over the choice of the sample set S_2,

$$\mathbb{E}_{v_1\sim Q_1}\left[L_{S_2}(f_{v_1}) - L(f_{v_1})\right] \leq \mathbb{E}_{v_2\sim Q_2}\left[L_{S_2}(f_{v_2}) - L(f_{v_2})\right] + \sqrt{\frac{2\mathrm{KL}(Q_2||Q_1) + 2\ln\frac{2n_2}{\delta}}{n_2-2}}. \tag{3.14}$$

Similar computations with S_1 and S_2 switched, and considering that $n_1 > 2$, yields that with probability at least $1-\delta/2$ over the choice of the sample set S_1,

$$\mathbb{E}_{v_2\sim Q_2}\left[L_{S_1}(f_{v_2}) - L(f_{v_2})\right] \leq \mathbb{E}_{v_1\sim Q_1}\left[L_{S_1}(f_{v_1}) - L(f_{v_1})\right] + \sqrt{\frac{2\mathrm{KL}(Q_1||Q_2) + 2\ln\frac{2n_1}{\delta}}{n_1-2}}. \tag{3.15}$$

The events in Eq. (3.14) and Eq. (3.15) jointly hold with probability at least $1-\delta$ over the choice of the sample sets S_1 and S_2 (using the union bound and De Morgan's law), and by adding the

two inequalities we therefore have

$$\mathbb{E}_{v_1\sim Q_1}\left[L_{S_2}(f_{v_1})\right]+\mathbb{E}_{v_2\sim Q_2}\left[L_{S_1}(f_{v_2})\right]\leq \mathbb{E}_{v_2\sim Q_2}\left[L_{S_2}(f_{v_2})\right]+\mathbb{E}_{v_1\sim Q_1}\left[L_{S_1}(f_{v_1})\right] + \sqrt{\frac{2\mathrm{KL}(Q_2||Q_1)+2\ln\frac{2n_2}{\delta}}{n_2-2}}+\sqrt{\frac{2\mathrm{KL}(Q_1||Q_2)+2\ln\frac{2n_1}{\delta}}{n_1-2}},$$

which concludes the proof.

□

3.D A Simple Connection Between Generalization Penalty and Gradient Disparity

In this section, we present an alternative and much simpler connection between the notions of generalization penalty and gradient disparity than the one presented in Section 3.1.3 and Section 3.1.4. Recall that each update step of the mini-batch gradient descent is written as $\theta_i=\theta-\gamma g_i$ for $i\in\{1,2\}$. By applying a first order Taylor expansion over the loss, we have

$$L_{S_1}(f_{\theta_1})=L_{S_1}(f_{\theta-\gamma g_1})\approx L_{S_1}(f_\theta)-\gamma g_1\cdot g_1.$$

The generalization penalties $\mathscr{R}_1$ and $\mathscr{R}_2$ would therefore be

$$\mathscr{R}_1=L_{S_1}(f_{\theta_2})-L_{S_1}(f_{\theta_1})\approx\gamma g_1\cdot(g_1-g_2),$$

and

$$\mathscr{R}_2=L_{S_2}(f_{\theta_1})-L_{S_2}(f_{\theta_2})\approx\gamma g_2\cdot(g_2-g_1),$$

respectively. Consequently,

$$\mathscr{R}_1+\mathscr{R}_2\approx\gamma\left\|g_1-g_2\right\|_2^2.$$

This derivation requires the loss function to be (approximately) linear near parameter vectors θ_1 and θ_2, which does not necessarily hold. Therefore, in the main chapter we only focus on the connection between generalization penalty and gradient disparity via Theorem 1, which does not require such an assumption. Nevertheless, it is interesting to note that this simple derivation recovers the connection between generalization penalty and gradient disparity.

3.E Common Experimental Details

The training objective in our experiments is to minimize the cross-entropy loss, and both the cross entropy and the error percentage are displayed in the figures. The training error is computed using Eq. (3.2) over the training set. The empirical test error also follows Eq. (3.2) but

it is computed over the test set. The generalization loss (respectively, error) is the difference between the test and the training cross entropy losses (resp., classification errors). The batch size in our experiments is 128 unless otherwise stated. The SGD learning rate is $\gamma = 0.01$ and no momentum is used (unless otherwise stated). All the experiments took at most few hours on one Nvidia Titan X Maxwell GPU. All the reported values throughout the chapter are an average over at least 5 runs.

To present results throughout the training, in the x-axis of figures, both epoch and iteration are used: an epoch is the time spent to pass through the entire dataset, and an iteration is the time spent to pass through one batch of the dataset. Thus, each epoch has B iterations, where B is the number of batches. The convolutional neural network configurations we use are: AlexNet [Krizhevsky et al. 2012], VGG [Simonyan and Zisserman 2014] and ResNet [He et al. 2016a]. In those experiments with varying width, we use a scaling factor to change both the number of channels and the number of hidden units in convolutional and fully connected layers, respectively. The default configuration is with scaling factor = 1. For the experiments with data augmentation (Fig. 3.6) we use random crop with padding = 4 and random horizontal flip with probability = 0.5.

In experiments with a random labeled training set, we modify the dataset similarly to [Chatterjee 2020]. For a fraction of the training samples, which is the amount of noise (0%, 25%, 50%, 75%, 100%), we choose the labels at random. For a classification dataset with a number K of classes, if the label noise is 25%, then on average $75\% + 25\% * 1/K$ of the training points still have the correct label.

3.E.1 Re-scaling the Loss

Let us track the evolution of gradient disparity (Eq. (3.6)) during training. As training progresses, the training losses of all the batches start to decrease when they get selected for the parameter update. Therefore, the value of gradient disparity might decrease, not necessarily because the distance between the two gradient vectors is decreasing, but because the value of each gradient vector is itself decreasing. To avoid this, a re-scaling or normalization is needed to compare gradient disparity at different stages of training.

If we perform a re-scaling with respect to the gradient vectors, then the gradient disparity between two batches S_1 and S_2 would be $\left\| g_1/\text{std}(g_1) - g_2/\text{std}(g_2) \right\|_2$, where $\text{std}(g_i)$ is the standard deviation of the gradients within batch S_i for $i \in \{1,2\}$. However, such a re-scaling would also absorb the variations of g_1 and g_2 with respect to each other. That is, if after an iteration, g_1 is scaled by a factor $\alpha < 1$, while g_2 remains unchanged, this re-scaling would leave the gradient disparity unchanged, although the performance of the network has improved only on S_1 and not on S_2, which might be a signal of overfitting.

[5] Note that in this figure, both the gradient disparity re-scaled and the generalization loss are increasing from the very first epoch. If we would use gradient disparity as an early stopping criterion, optimization would stop at epoch 5 and we would have a 0.36 drop in the test loss value, compared to the loss reached when the model

We therefore propose to normalize the loss values instead, before computing gradient disparity, so that the initial losses of two different iterations would have the same scale. We can normalize the loss values by

$$L_{S_j} = \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{l_i - \mathrm{Min}_i\,(l_i)}{\mathrm{Max}_i\,(l_i) - \mathrm{Min}_i\,(l_i)},$$

where with some abuse of notation, l_i is the cross entropy loss for the data point i in the batch S_j. However, this normalization might be sensitive to outliers, making the bulk of data end up in a very narrow range within 0 and 1, and degrading in turn the accuracy of the signal of overfitting. Re-scaling is usually less sensitive to outliers in comparison with normalization, it leads to loss values that are given by

$$L_{S_j} = \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{l_i}{\mathrm{std}_i\,(l_i)}.$$

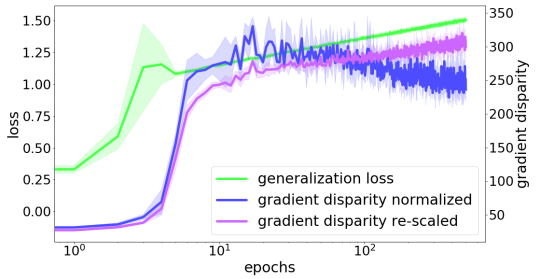


Figure 3.9: Normalizing versus re-scaling loss before computing average gradient disparity $\overline{\mathfrak{D}}$ for a VGG-11 trained on 12.8 k points of the CIFAR-10 dataset[5]. For this experiment, Pearson's correlation coefficient between gradient disparity re-scaled (our chosen metric) and the test loss is 0.91. If we would have instead normalized the loss values the correlation would be 0.88. If we would have re-scaled with respect to the gradients (instead of the loss), the correlation would be 0.79.

We experimentally compare these two ways of computing gradient disparity in Fig. 3.9. Both the re-scaled and normalized losses might get unbounded, if within each batch the loss values are very close to each other. However, in our experiments, we do not observe gradient disparity becoming unbounded either way. We observe that the correlation between gradient disparity and the test loss is the highest if we re-scale the loss values before computing gradient disparity. This is therefore how we compute gradient disparity in all experiments presented in the chapter. Note that this re-scaling does not affect the training algorithm, since it is only used to compute the gradient disparity metric (we do not perform loss re-scaling before `opt.step()`).

achieves 0 training loss.

3.E.2 The Hyper-parameter s

In this section, we briefly study the choice of the size s of the subset of batches to compute the average gradient disparity

$$\overline{\mathfrak{D}} = \frac{1}{s(s-1)} \sum_{i=1}^{s} \sum_{j=1, j\neq i}^{s} \mathfrak{D}_{i,j}.$$

Fig. 3.10 shows the average gradient disparity when averaged over s number of batches[6]. When $s = 2$, gradient disparity is the ℓ_2 norm distance of the gradients of two randomly selected batches and has a quite high variance. Although with higher values of s the results have lower variance, computing it with a large value of s is more computationally expensive (refer to Section 3.F for more details). Therefore, we find the choice of $s = 5$ to be sufficient enough to track overfitting; in all the experiments reported in this chapter, we use $s = 5$.

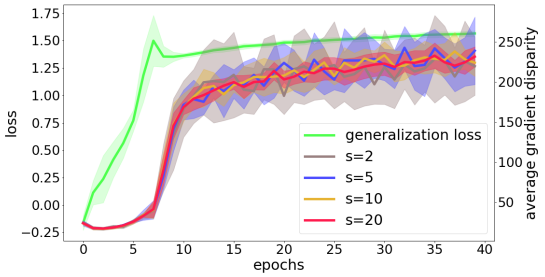


Figure 3.10: Average gradient disparity for different averaging parameter s for a ResNet-18 that has been trained on 12.8k points of the CIFAR-10 dataset.

3.E.3 The Surrogate Loss Function

Cross entropy has been shown to be better suited for computer-vision classification tasks, compared to mean square error [Kline and Berardi 2005; Hui and Belkin 2020]. Hence, we choose the cross entropy criterion for all our experiments to avoid possible pitfalls of the mean square error, such as not tracking the confidence of the predictor.

[Soudry et al. 2018] argues that when using cross entropy, as training proceeds, the magnitude of the network parameters increases. This can potentially affect the value of gradient disparity. Therefore, we compute the magnitude of the network parameters over iterations in various settings. We observe that this increase is very low both at the end of the training and, more importantly, at the time when gradient disparity signals overfitting (denoted by GD epoch in Table 3.9). Therefore, it is unlikely that the increase in the magnitude of the network parameters affects the value of gradient disparity.

Furthermore, we examine gradient disparity for models trained on the mean square error, instead of the cross entropy criterion. We observe a high correlation between gradient disparity and test error/loss (Fig. 3.11), which is consistent with the results obtained using the cross entropy criterion. The applicability of gradient disparity as a generalization metric is therefore not limited to settings with the cross entropy criterion.

[6] In the setting of Fig. 3.10, if we use gradient disparity as an early stopping criterion, optimization would stop at epoch 9 and we would have a 0.28 drop in the test loss value compared to the loss reached when the model achieves 0 training loss.

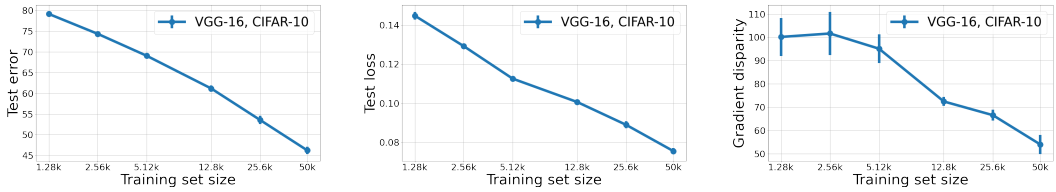


Figure 3.11: Test error (TE), test loss (TL), and gradient disparity ($\overline{\mathfrak{D}}$) for VGG-16 trained with different training set sizes to minimize the *mean square error* criterion on the CIFAR-10 dataset. The Pearson correlation coefficient between TE and $\overline{\mathfrak{D}}$ and between TL and $\overline{\mathfrak{D}}$ are $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.976$ and $\rho_{\overline{\mathfrak{D}},\mathrm{TL}} = 0.943$, respectively.

Setting	at epoch 0	at GD epoch	at epoch 200
(1)	1	1.00034	1.00123
(2)	1	1.00019	1.00980
(3)	1	1.00107	1.00127
(4)	1	1.00222	1.00233

Table 3.9: The ratio of the magnitude of the network parameter vector at epoch t to the magnitude of the network parameter vector at epoch 0, for $t \in \{0, \mathrm{GD}, 200\}$, where GD stands for the epoch when gradient disparity signals to stop the training. Setting (1): AlexNet, MNIST, (2): AlexNet, MNIST, 50% random, (3): VGG-16, CIFAR-10, and (4): VGG-16, CIFAR-10, 50% random.

3.F k-fold Cross-Validation

k-fold cross-validation (CV) splits the available dataset into k sets, training on $k-1$ of them and validating on the remaining one. This is repeated k times so that every set is used once as the validation set. Each experiment out of k folds can itself be viewed as a setting where the available data is split into a training and a validation set. Early stopping can then be done by evaluating the network performance on the validation set and stopping the training as soon as there is an increase in the value of the validation loss. However, in practice, the validation loss curve is not necessarily smooth (as can be clearly observed in our experiments throughout the chapter), and therefore as discussed in [Prechelt 1998; Lodwich et al. 2009], there is no obvious early stopping rule (threshold) to obtain the minimum value of the generalization error.

3.F.1 Early Stopping Threshold

In this chapter, we adapt two different early stopping thresholds: (t1) stop training when there are $p = 5$ (consecutive or nonconsecutive) increases in the value of the early stopping metric (the hyper-parameter p is commonly referred to as the "patience" parameter among practitioners), which is indicated by the gray vertical bars in Fig. 3.13 and Fig. 3.14, and (t2) stop training when there are p consecutive increases in the value of the early stopping metric, which is indicated by the magenta vertical bars in Fig. 3.13 and Fig. 3.14. When there are either low variations or a sharp increase in the value of the metric, the two coincide (for instance, in Fig. 3.13 (b) (middle left)). For k-fold CV, the early stopping metric is the validation loss, and for our proposed method, the early stopping criterion is gradient disparity (GD).

Which exact patience parameter to choose as an early stopping threshold, or whether it should include non-consecutive increases or not, are indeed interesting questions [Prechelt 1998], which do not have a definite answer to date even for k-fold CV. Table 3.10 and Table 3.11 give the results obtained by 20 different early stopping thresholds for both k-fold CV (shown on the left tables) and GD (shown on the right tables). We also give the best, mean and standard deviation of test loss and test accuracy across all thresholds. In the following two paragraphs, we summarize the findings of these two tables.

Performance In Fig. 3.12, we observe that the test accuracy (averaged over 20 thresholds) is higher when using GD as an early stopping criterion, than k-fold CV. Note that beforehand we do not have access to the test set to choose the best possible threshold, hence in Fig. 3.12 the average test accuracy is reported over all thresholds. However, even if we did have the test set to choose the best threshold, we can still observe from Table 3.10 and Table 3.11 that GD either performs comparably to CV, or that it significantly outperforms CV.

Sensitivity to Threshold Ideally, we would like to have a robust metric that does not strongly depend on the early stopping threshold. To compute the sensitivity of each method

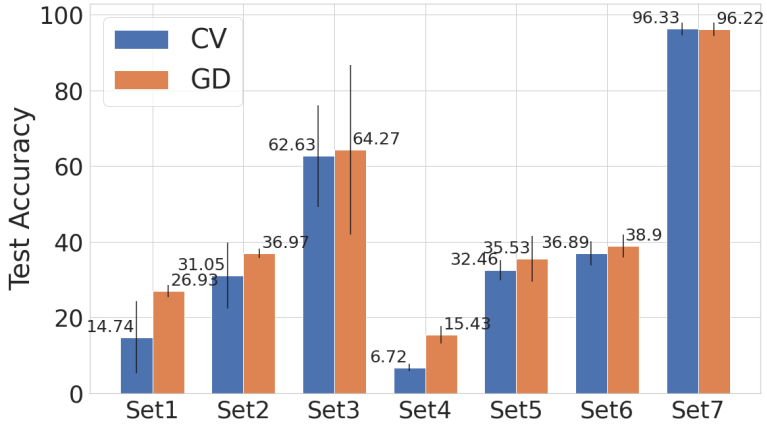


Figure 3.12: Test Accuracy achieved by using GD and k-fold CV as early stopping methods in 7 experimental settings (indicated in the x-axis by Set1-7). The result is averaged over 20 choices of the early stopping threshold. The complete set of results are reported in Table 3.10 and Table 3.11. For the CIFAR-100 experiments (Set1, 4, and 5) the top-5 accuracy is reported.

to the choice of the threshold, we compute

$$\text{Sensitivity to the threshold} = \sum_{i=1}^{7} \text{std}(\text{Set}_i)/\text{Mean}(\text{Set}_i), \tag{3.16}$$

where $\text{Mean}(\text{Set}_i)$ and $\text{std}(\text{Set}_i)$ are the mean and standard deviation of the test accuracy/loss across different thresholds, respectively, of setting i across the early stopping thresholds, which are reported in Table 3.10 and Table 3.11. The lower the sensitivity is, the more the method is robust to the choice of the early stopping threshold. In Table 3.5, we observe that GD is less sensitive and more robust to the choice of the early stopping threshold than k-fold CV, which is another advantage of GD over CV. This can also be observed from our figures: In most of the experiments (more precisely, in 5 out of 7 experiments of Fig. 3.13 and Fig. 3.14), GD is not sensitive to the choice of the threshold (see Fig. 3.13 (a), Fig. 3.13 (b), Fig. 3.14 (a), Fig. 3.13 (b) and Fig. 3.13 (c) on the 2nd column, where the gray and magenta bars almost coincide). In contrast, k-fold CV is more sensitive (see for example the leftmost column of Fig. 3.13 (a) and Fig. 3.13 (b), where the gray and magenta bars are very far away when using k-fold CV). In the other 2 experiments (Fig. 3.13 (c) and Fig. 3.14 (d), both with the MNIST dataset), the thresholds (t1) and (t2) do not coincide for neither GD nor k-fold CV. In Table 3.12, we further study these two settings: we provide the test accuracy for experiments of Fig. 3.13 (c) and Fig. 3.14 (d) (both with the MNIST dataset), for different values of p. We again observe that even if we optimize p for k-fold CV (reported in bold in Table 3.12), GD still outperforms k-fold CV.

Threshold	Loss	ACC	top-5 ACC	Epoch
($t1_1$)	4.63	0.98	5.02	1
($t1_2$)	4.65	0.98	5.0	2
($t1_3$)	4.65	1.19	5.99	3
($t1_4$)	4.25	6.73	21.47	12
($t1_5$)	4.25	6.79	22.19	14
($t1_6$)	4.23	7.46	22.48	17
($t1_7$)	4.25	7.22	22.18	18
($t1_8$)	4.21	7.69	23.50	21
($t1_9$)	4.23	7.94	23.72	22
($t1_{10}$)	4.27	7.58	22.84	24
($t2_1$)	4.63	0.98	5.02	1
($t2_2$)	4.65	0.98	5.0	2
($t2_3$)	4.65	1.19	5.99	3
($t2_4$)	4.12	9.50	26.71	45
($t2_5$)	4.12	9.45	26.32	46
($t2_6$)	4.19	9.51	26.43	314
($t2_7$)	4.20	9.54	26.47	-[b]
($t2_8$)	-	-	-	-
($t2_9$)	-	-	-	-
($t2_{10}$)	-	-	-	-
best	4.11	9.51	26.71	-
mean	4.42	4.64	14.74	38.33
std	0.23	3.70	9.56	84.53
range	4.11 – 4.65	0.98 – 9.51	5.0 – 26.71	1 – 314

(a) CV, CIFAR-100, ResNet-34, limited dataset (Fig. 3.13(a))

Threshold	Loss	ACC	top-5 ACC	Epoch
($t1_1$)	4.27	7.36	23.58	17
($t1_2$)	4.19	7.88	24.28	19
($t1_3$)	4.14	8.85	25.67	22
($t1_4$)	4.13	9.08	25.91	23
($t1_5$)	4.06	9.99	27.84	24
($t1_6$)	4.12	9.32	26.61	25
($t1_7$)	4.07	9.86	27.34	26
($t1_8$)	4.06	10.04	27.89	27
($t1_9$)	4.04	10.43	28.35	28
($t1_{10}$)	4.04	10.41	28.40	29
($t2_1$)	4.27	7.36	23.58	17
($t2_2$)	4.13	9.08	25.91	23
($t2_3$)	4.06	9.99	27.84	24
($t2_4$)	4.12	9.33	26.61	25
($t2_5$)	4.07	9.86	27.34	26
($t2_6$)	4.06	10.04	28.89	27
($t2_7$)	4.04	10.43	28.35	28
($t2_8$)	4.04	10.41	28.40	29
($t2_9$)	4.05	10.29	28.41	30
($t2_{10}$)	4.05	10.39	28.31	31
best	4.04	10.43	28.41	-[a]
mean	**4.1**	**9.52**	**26.93**	25
std	0.07	0.98	1.57	3.89
range	4.04 – 4.27	7.4 – 10.4	23.6 – 28.4	17 – 31

(b) GD, CIFAR-100, ResNet-34, limited dataset (Fig. 3.13(a))

Threshold	Loss	ACC	Epoch
best	1.84	37.03	-
mean	1.97	31.05	12.25
std	0.22	8.80	10.54
range	1.84 – 2.35	15.84 – 37.03	1 – 30

(c) CV, CIFAR-10, VGG-13, limited dataset (Fig. 3.13(b))

Threshold	Loss	ACC	Epoch
best	1.79	38.16	-
mean	**1.80**	**36.97**	6.5
std	0.02	1.27	2.87
range	1.79 – 1.85	33.71 – 38.16	2 – 11

(d) GD, CIFAR-10, VGG-13, limited dataset (Fig. 3.13(b))

Threshold	Loss	ACC	Epoch
best	0.56	81.39	34
mean	**1.09**	62.63	21.71
std	0.36	13.46	9.59
range	0.63 – 1.64	41.15 – 81.39	9 – 41

(e) CV, MNIST, AlexNet, limited dataset (Fig. 3.13(c))

Threshold	Loss	ACC	Epoch
best	0.45	86.39	39
mean	1.13	**64.27**	17.58
std	0.72	22.38	13.91
range	0.45 – 2.18	30.19 – 86.39	1 – 40

(f) GD, MNIST, AlexNet, limited dataset (Fig. 3.13(c))

Table 3.10: The test accuracy, test loss, top-5 accuracy, and stopping epoch obtained by using k-fold cross-validation (CV) (left columns (a), (c), and (e)) and by using gradient disparity (GD) (right columns (b), (d), and (f)) as early stopping criteria for different patience values and for different thresholds (t1) and (t2). ($t1_p$): training is stopped after p increases in the value of the validation loss in k-fold CV, and of GD, respectively. ($t2_p$): training is stopped after p consecutive increases in the value of the validation loss in k-fold CV, and of GD, respectively. For the rest of the experiments we only report the best values, mean and standard deviation (std) over all thresholds. $-^a$: The epoch to have the best test loss does not coincide with the epoch to have the best test accuracy. $-^b$: The metric does not have p consecutive increases during training.

Threshold	Loss	ACC	top-5 ACC	Epoch
best	4.69	2.00	9.38	19
mean	4.94	1.60	6.72	12.3
std	0.1	0.14	0.98	4.47
range	4.69 – 5.03	1.42 – 2.00	5.62 – 9.38	6 – 20

(a) CV, CIFAR-100, ResNet-18, noisy dataset (Fig. 3.14(a))

Threshold	Loss	ACC	top-5 ACC	Epoch
best	4.38	4.38	17.76	-
mean	**4.47**	**3.79**	**15.43**	29.25
std	0.08	0.75	2.31	6.41
range	4.38 – 4.69	2.13 – 4.38	9.74 – 17.76	18 – 41

(b) GD, CIFAR-100, ResNet-18, noisy dataset (Fig. 3.14(a))

Threshold	Loss	ACC	top-5 ACC	Epoch
best	3.87	12.14	37.06	-
mean	**4.16**	10.19	32.46	11.5
std	0.25	1.27	2.72	2.87
range	3.87 – 4.5	8.47 – 12.14	27.80 – 37.06	7 – 16

(c) CV, CIFAR-100, ResNet-34, noisy dataset (Fig. 3.14(b))

Threshold	Loss	ACC	top-5 ACC	Epoch
best	3.82	15.81	40.97	-
mean	4.37	**12.82**	**35.53**	14.75
std	0.25	1.76	6.08	6.59
range	3.82 – 4.59	10.41 – 15.81	23.51 – 40.97	3 – 23

(d) GD, CIFAR-100, ResNet-34, noisy dataset (Fig. 3.14(b))

Threshold	Loss	ACC	Epoch
best	1.69	43.02	2
mean	**2.19**	36.89	6.5
std	0.35	3.21	2.87
range	1.69 – 2.66	32.41 – 43.02	2 – 11

(e) CV, CIFAR-10, VGG-13, noisy dataset (Fig. 3.14(c))

Threshold	Loss	ACC	Epoch
best	1.77	42.45	-
mean	2.35	**38.9**	10.1
std	0.27	2.98	3.59
range	1.77 – 2.59	32.92 – 42.45	4 – 16

(f) GD, CIFAR-10, VGG-13, noisy dataset (Fig. 3.14(c))

Threshold	Loss	ACC	Epoch
best	0.59	97.44	-
mean	**0.63**	**96.33**	23.5
std	0.18	1.68	13.11
range	0.59 – 0.69	92.03 – 97.44	7 – 49

(g) CV, MNIST, AlexNet, noisy dataset (Fig. 3.14(d))

Threshold	Loss	ACC	Epoch
best	0.62	97.49	-
mean	0.65	96.22	20.4
std	0.02	1.81	13.81
range	0.62 – 0.66	92.58 – 97.49	10 – 48

(h) GD, MNIST, AlexNet, noisy dataset (Fig. 3.14(d))

Table 3.11: The test accuracy, test loss, top-5 accuracy, and stopping epoch obtained by using k-fold cross-validation (CV) (left columns (a), (c), (e), and (g)) and by using gradient disparity (GD) (right columns (b), (d), (f), and (h)) as early stopping criteria for different patience values and for different thresholds (t1) and (t2). (t1$_p$): training is stopped after p increases in the value of the validation loss in k-fold CV, and of GD, respectively. (t2$_p$): training is stopped after p consecutive increases in the value of the validation loss in k-fold CV, and of GD, respectively. We report the best values, mean and standard deviation (std) over 20 thresholds.

3.F.2 Image-classification Benchmark Datasets

Limited Data Fig. 3.13 and Table 3.13 show the results for MNIST, CIFAR-10 and CIFAR-100 datasets, where we simulate the limited data scenario by using a small subset of the training set. For the CIFAR-100 experiment (Fig. 3.13 (a) and Table 3.13 (top row), we observe (from the left figure) that the validation loss predicts the test loss pretty well. We observe (from the middle left figure) that gradient disparity also predicts the test loss quite well. However, the main difference between the two settings is that when using cross-validation, $1/k$ of the data is set aside for validation and $1-1/k$ of the data is used for training. Whereas when using gradient disparity, all the data ($1-1/k+1/k=1$) is used for training. Hence, the test loss in the leftmost and middle left figures differ. The difference between the test accuracy (respectively, test loss) obtained in each setting is visible in the rightmost figure (resp., middle right figure). We observe that there is over 3% improvement in the test accuracy when using gradient disparity as an early stopping criterion. This improvement is consistent for the MNIST and CIFAR-10 datasets (Fig. 3.13 (b) and (c) and Table 3.13). We conclude that in the absence of label noise, both k-fold cross-validation and gradient disparity predict the optimal early stopping moment, but the final test loss/error is much lower for the model trained with all the available data (thus, when gradient disparity is used), than the model trained with a $(1-1/k)$ portion of the data (thus when k-fold cross-validation is used). To further test on a dataset that is itself limited, a medical application with limited labeled data is empirically studied later in this section (Section 3.F.3). The same conclusion is made for this dataset.

Noisy Labeled Data The results for datasets with noisy labels are shown in Fig. 3.14 and Table 3.14 for the MNIST, CIFAR-10 and CIFAR-100 datasets. We observe (from Fig. 3.14 (a) (left)) that for the CIFAR-100 experiment, the validation loss does no longer predict the test loss. Nevertheless, although gradient disparity is computed on a training set that contains corrupted samples, it predicts the test loss quite well (Fig. 3.14 (a) (middle left)). There is a 2% improvement in the final test accuracy (for top-5 accuracy there is a 9% improvement) (Table 3.14 (top two rows)) when using gradient disparity instead of a validation set as an early stopping criterion. This is also consistent for other configurations and datasets (Fig. 3.14 and Table 3.14). We conclude that, in the presence of label noise, k-fold cross-validation does no longer predict the test loss and fails as an early stopping criterion, unlike gradient disparity.

Computational Cost Denote the time, in seconds, to compute one gradient vector, to compute the ℓ_2 norm between two gradient vectors, to take the update step for the network parameters , and to evaluate one batch (find its validation loss and error) by t_1, t_2, t_3 and t_4, respectively. Then, one epoch of k-fold cross-validation takes

$$\mathrm{CV}_{\mathrm{epoch}} = k \times \left(\frac{k-1}{k} B(t_1 + t_3) + \frac{B}{k} t_4 \right)$$

seconds, where B is the number of batches. Performing one epoch of training and computing

Method \ Patience	1	5	10	15	20	25
5-fold CV	$41.15_{\pm 5.68}$	$62.62_{\pm 6.36}$	$81.39_{\pm 3.64}$	$80.39_{\pm 2.88}$	$\mathbf{84.84}_{\pm 2.53}$	$83.55_{\pm 2.84}$
GD	$30.19_{\pm 6.21}$	$79.12_{\pm 3.04}$	$84.82_{\pm 2.14}$	$85.35_{\pm 2.09}$	$\mathbf{87.28}_{\pm 1.24}$	$86.69_{\pm 1.31}$

(a) MNIST, AlexNet, limited dataset (Fig. 3.13 (c))

Method \ Patience	1	5	10	15	20	25
10-fold CV	$96.54_{\pm 0.15}$	$97.28_{\pm 0.20}$	$\mathbf{97.35}_{\pm 0.23}$	$97.22_{\pm 0.19}$	$96.60_{\pm 0.33}$	$94.69_{\pm 0.87}$
GD	$97.07_{\pm 0.16}$	$97.32_{\pm 0.15}$	$\mathbf{97.41}_{\pm 0.15}$	$96.57_{\pm 0.64}$	$95.44_{\pm 0.96}$	$92.58_{\pm 0.65}$

(b) MNIST, AlexNet, noisy dataset (Fig. 3.14 (d))

Table 3.12: The test accuracies achieved by using k-fold cross-validation (CV) and by using gradient disparity (GD) as early stopping criteria for different patience values. For a given patience value of p, the training is stopped after p increases in the value of the validation loss in k-fold CV (top rows) and of GD (bottom rows). Throughout the chapter, we have chosen $p = 5$ as the default patience value for all methods without optimizing it even for GD. However, in this table (also in Table 3.10 and Table 3.11), we observe that even if we tune the patience value for k-fold CV and for GD separately (which is indicated in bold), GD still outperforms k-fold CV. Moreover, as we discussed in Section 3.F.1, even if we take an average over patience values and early stopping thresholds (to avoid the need to tune this parameter), GD again outperforms CV (Fig. 3.12).

the gradient disparity takes

$$\mathrm{GD}_{\mathrm{epoch}} = B(t_1 + t_3) + s\left(t_1 + \frac{s-1}{2}t_2\right)$$

seconds. In our experiments, we observe that $t_1 \approx 5.1\,t_2 \approx 100\,t_3 \approx 3.4\,t_4$, hence the approximate time to perform one epoch for each setting is

$$\mathrm{CV}_{\mathrm{epoch}} \approx (k-1)B\,t_1, \qquad \text{and} \qquad \mathrm{GD}_{\mathrm{epoch}} \approx (B+s)\,t_1.$$

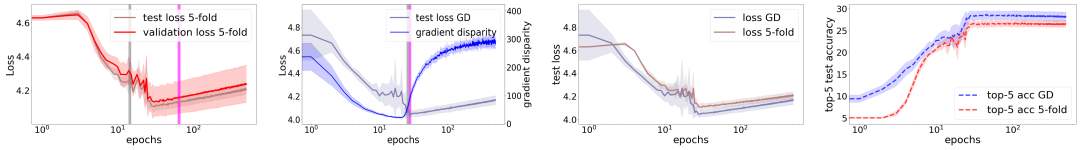
Therefore, as $s < B$, we have $\mathrm{CV}_{\mathrm{epoch}} \gg \mathrm{GD}_{\mathrm{epoch}}$.

3.F.3 MRNet Dataset

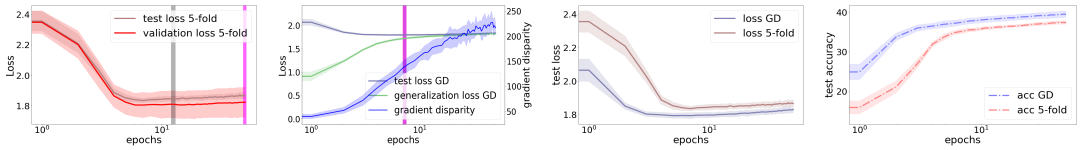
So far, we have shown the improvement of gradient disparity over cross-validation for limited subsets of MNIST, CIFAR-10 and CIFAR-100 datasets. In this sub-section, we give the results for the MRNet dataset [Bien et al. 2018] used for diagnosis of knee injuries, which is by itself limited. The dataset contains 1370 magnetic resonance imaging (MRI) exams to study the presence of abnormality, anterior cruciate ligament (ACL) tears and meniscal tears. The labeled data in the MRNet dataset is therefore very limited. Each MRI scan is a set of S slices of

Setting	Method	Test loss	Test accuracy
CIFAR-100, ResNet-34	5-fold CV	$4.249_{\pm 0.028}$	$6.79_{\pm 0.49}$ (top-5: $22.19_{\pm 0.77}$)
	GD	$\mathbf{4.057}_{\pm 0.043}$	$\mathbf{9.99}_{\pm 0.92}$ (top-5: $\mathbf{27.84}_{\pm 1.30}$)
CIFAR-10, VGG-13	5-fold CV	$1.846_{\pm 0.016}$	$35.982_{\pm 0.393}$
	GD	$\mathbf{1.793}_{\pm 0.016}$	$\mathbf{36.96}_{\pm 0.861}$
MNIST, AlexNet	5-fold CV	$1.123_{\pm 0.25}$	$62.62_{\pm 6.36}$
	GD	$\mathbf{0.656}_{\pm 0.080}$	$\mathbf{79.12}_{\pm 3.04}$

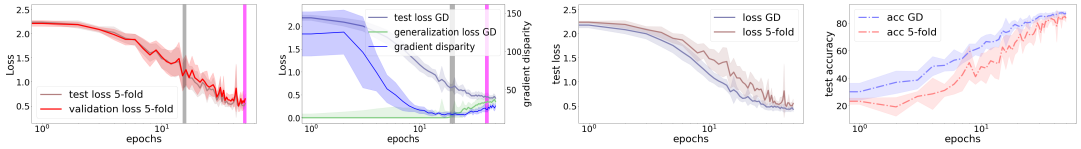
Table 3.13: The loss and accuracy on the test set comparing 5-fold cross-validation and gradient disparity as early stopping criterion when the available dataset is limited. The corresponding curves during training are presented in Fig. 3.13. The results above are obtained by stopping the optimization when the metric (either validation loss or gradient disparity) has increased for five epochs from the beginning of training.



(a) ResNet-34 trained on 1.28 k points of CIFAR-100



(b) VGG-13 trained on 1.28 k points of CIFAR-10

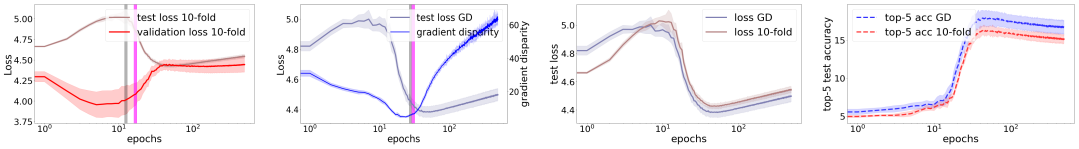


(c) AlexNet trained on 256 points of MNIST

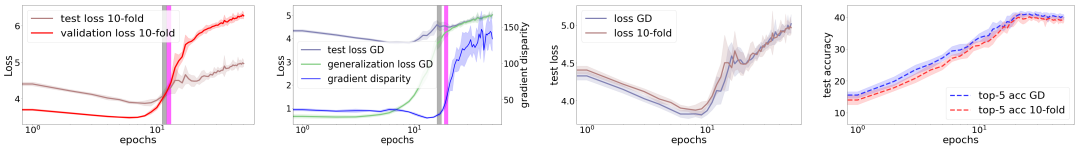
Figure 3.13: Comparing 5-fold cross-validation (CV) with gradient disparity (GD) as an early stopping criterion when the available dataset is limited. (left) Validation loss versus test loss in 5-fold cross-validation. (middle left) Gradient disparity versus test and generalization losses. (middle right and right) Performance on the unseen (test) data for GD versus 5-fold CV. (a) The parameters are initialized by Xavier techniques with uniform distribution. (b, c) The parameters are initialized using He technique with normal distribution. (c) The batch size is 32. The gray and magenta vertical bars indicate the epoch in which the metric (the validation loss or gradient disparity) has increased for 5 epochs from the beginning of training and for 5 consecutive epochs, respectively. In (b) the middle left figure, these two bars meet each other.

Setting	Method	Test loss	Test accuracy
CIFAR-100, ResNet-18	10-fold CV	$5.023_{\pm 0.083}$	$1.59_{\pm 0.15}$ (top-5: $6.47_{\pm 0.52}$)
	GD	$\mathbf{4.463}_{\pm 0.038}$	$\mathbf{3.68}_{\pm 0.52}$ (top-5: $\mathbf{15.22}_{\pm 1.24}$)
	10^+-fold CV	$4.964_{\pm 0.057}$	$1.68_{\pm 0.24}$ (top-5: $7.05_{\pm 0.71}$)
CIFAR-100, ResNet-34	10-fold CV	$4.062_{\pm 0.091}$	$9.62_{\pm 1.08}$ (top-5: $32.06_{\pm 1.47}$)
	GD	$4.592_{\pm 0.179}$	$\mathbf{10.41}_{\pm 1.40}$ (top-5: $\mathbf{36.92}_{\pm 1.20}$)
	10^+-fold CV	$4.134_{\pm 0.185}$	$10.11_{\pm 1.60}$ (top-5: $34.19_{\pm 2.10}$)
CIFAR-10, VGG-13	10-fold CV	$2.126_{\pm 0.063}$	$34.88_{\pm 1.66}$
	GD	$2.519_{\pm 0.062}$	$\mathbf{36.98}_{\pm 0.77}$
	10^+-fold CV	$2.195_{\pm 0.142}$	$35.40_{\pm 3.00}$
MNIST, AlexNet	10-fold CV	$0.656_{\pm 0.034}$	$97.28_{\pm 0.20}$
	GD	$0.654_{\pm 0.031}$	$\mathbf{97.32}_{\pm 0.27}$
	10^+-fold CV	$0.639_{\pm 0.029}$	$97.31_{\pm 0.15}$

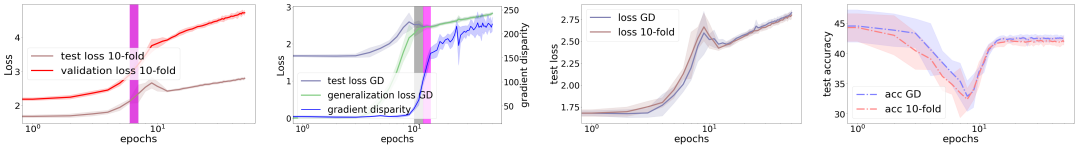
Table 3.14: The loss and accuracy on the test set comparing 10-fold cross-validation and gradient disparity as early stopping criterion when the available dataset is noisy. In all the experiments, 50% of the available data has random labels. The corresponding curves during training are shown in Fig. 3.14. The results below are obtained by stopping the optimization when the metric (either validation loss or gradient disparity) has increased for five epochs from the beginning of training. The last row in each setting, which we call 10^+-fold CV, refers to the test loss and accuracy reached at the epoch suggested by 10-fold CV, for a network trained on the entire set. In all these settings, using GD still results in a higher test accuracy.



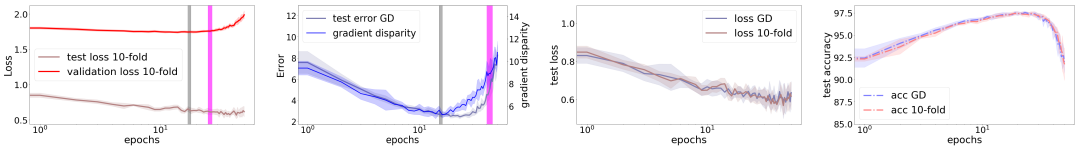
(a) ResNet-18 trained on 1.28 k points of CIFAR-100 dataset with 50% label noise



(b) ResNet-34 trained on the entire CIFAR-100 dataset with 50% label noise



(c) VGG-13 trained on the entire CIFAR-10 dataset with 50% label noise

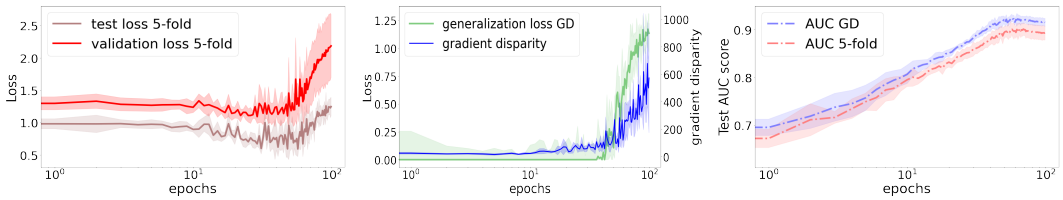


(d) AlexNet trained on the entire MNIST dataset with 50% label noise

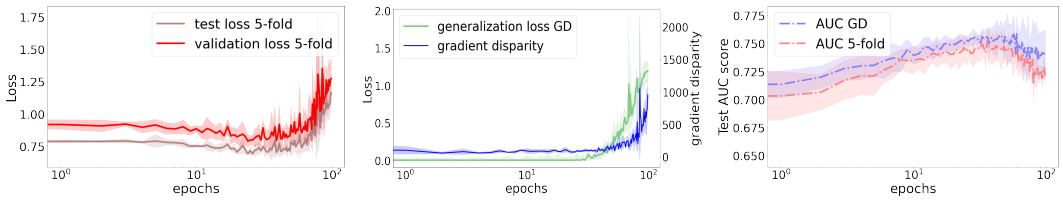
Figure 3.14: Comparing 10-fold cross-validation with gradient disparity as early stopping criteria when the available dataset is noisy. (left) Validation loss versus test loss in 10-fold cross-validation. (middle left) Gradient disparity versus test and generalization losses. (middle right and right) Performance on the unseen (test) data for GD versus 10-fold CV. (a) The parameters are initialized by Xavier techniques with uniform distribution. (b, c, and d) The parameters are initialized using He technique with normal distribution.

images stacked together. Note that, in this dataset, because slice S changes from one patient to another, it is not possible to stack the data into batches, hence the batch size is 1, which may explain the fluctuations of both the validation loss and gradient disparity in this setting. Each patient (case) has three MRI scans: sagittal, coronal and axial. The MRNet dataset is split into training (1130 cases), validation (120 cases) and test sets (120 cases). The test set is not publicly available. We need however to set aside some data to evaluate both gradient disparity and k-fold cross-validation, hence, in our experiments, the validation set becomes the unseen (test) set. To perform cross-validation, we split the set used for training in [Bien et al. 2018] into a first subset used for training in our experiments, and a second subset used as validation set. We use the SGD optimizer with the learning rate 10^{-4} for training the model. Each task in this dataset is a binary classification with an unbalanced set of samples, hence we report the area under the curve of the receiver operating characteristic (AUC score).

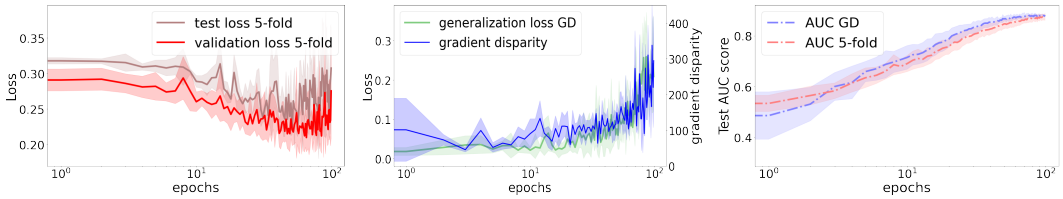
The results for three tasks (detecting ACL tears, meniscal tears and abnormality) are shown in Fig. 3.15 and Table 3.1. We can observe that both the validation loss (despite a small bias) and the gradient disparity predict the generalization loss quite well. Yet, when using gradient disparity, the final test AUC score is higher (Fig. 3.15 (right)). As mentioned above, for this dataset, both the validation loss and gradient disparity vary a lot. Hence, in Table 3.1, we show the results of early stopping, both when the metric has increased for 5 epochs from the beginning of training, and between parenthesis when the metric has increased for 5 consecutive epochs. We conclude that with both approaches, the use of gradient disparity as an early stopping criterion results in more than 1% improvement in the test AUC score. Because the test set used in [Bien et al. 2018] is not publicly available, it is not possible to compare our predictive results with [Bien et al. 2018]. Nevertheless, we can take as a baseline the results presented in the work given at https://github.com/ahmedbesbes/mrnet, which report a test AUC score of 88.5% for the task of detecting ACL tears. We observe in Table 3.1 that stopping training after 5 consecutive increases in gradient disparity leads to 91.52% test AUC score for this task. With further tuning, and combining the predictions found on two other MRI planes of each patient (axial and coronal), our final prediction results could even be improved.



(a) Task: detecting ACL tears



(b) Task: detecting meniscal tears



(c) Task: detecting abnormality

Figure 3.15: Detecting three tasks from the MRNet dataset from the sagittal plane MRI scans. (left) Validation loss versus test loss in 5-fold cross-validation. (middle) Gradient disparity versus generalization loss. (right) Performance comparison on the final unseen data when applying 5-fold CV versus gradient disparity. For the results of applying early stopping refer to Table 3.1.

3.G Additional Experiments

In this section, we provide additional experiments on benchmark image-classification datasets.

3.G.1 MNIST Experiments

Fig. 3.5 shows the test error for networks trained with different amounts of label noise. Interestingly, observe that for this setting the test error for the network trained with 75% label noise remains relatively small, indicating a good resistance of the model against memorization of corrupted samples. As suggested both from the test error (Fig. 3.5 (a)) and gradient disparity (Fig. 3.5 (c)), there is no proper early stopping time for these experiments[7]. The generalization error (Fig. 3.5 (b)) remains close to zero, regardless of the level of label noise, and hence fails to account for label noise. In contrast, gradient disparity is very sensitive to the label noise level in *all* stages of training, even at early stages of training, as desired for a metric measuring generalization.

Fig. 3.16 shows the results for an AlexNet [Krizhevsky et al. 2012] trained on the MNIST dataset[8]. This model generalizes quite well for this dataset. We observe that, throughout the training, the test curves are even below the training curves, which is due to the dropout regularization technique [Srivastava et al. 2014] being applied during training and not during testing. The generalization loss/error is almost zero, until around iteration 1100 (indicated in the figure by the gray vertical bar), which is when overfitting starts and the generalization error becomes non-zero, and when gradient disparity signals to stop training.

Fig. 3.17 shows the results for a 4-layer fully connected neural network trained on the entire MNIST training set. Fig. 3.17 (e) and (f) show the generalization losses. We observe that at the early stages of training, generalization losses do not distinguish between different label noise levels, whereas gradient disparity does so from the beginning (Fig. 3.17 (g) and (h)). At the middle stages of training we can observe that, surprisingly in this setting, the network with 0% label noise has higher generalization loss than the networks trained with 25%, 50% and 75% noise, and this is also captured by gradient disparity. The final gradient disparity values for the networks trained with higher label noise level are also larger. For the network trained with 0% label noise we show the results with more details in Fig. 3.18, and observe again how gradient disparity is well aligned with the generalization loss/error. In this experiment, the early stopping time suggested by gradient disparity is epoch 9, which is the exact same time when the training and test losses/errors start to diverge, and signals therefore the start of overfitting.

[7] According to Table 3.12, for the noisy MNIST dataset, the patience value of $p = 10$ is preferred for both GD and CV. Therefore, in Fig. 3.5 (c), even for the setting with 75% label noise, gradient disparity does not increase for $p = 10$ consecutive iterations, and would therefore not signal overfitting throughout training.

[8] http://yann.lecun.com/exdb/mnist/

3.G.2 CIFAR-10 Experiments

Fig. 3.19 shows the results for a ResNet-18 [He et al. 2016a] trained on the CIFAR-10 dataset[9]. Around iteration 500 (which is indicated by a thick gray vertical bar in the figures), the training and test losses (and errors) start to diverge, and the test loss reaches its minimum. This is indeed when gradient disparity increases and signals overfitting.

To compare models with a different number of parameters using gradient disparity, we need to normalize it. The dimension of a gradient vector is the number d of parameters of the model. Gradient disparity being the ℓ_2-norm of the difference of gradient vectors will thus grow proportionally to $\sqrt{d}$, hence to compare different architectures, we propose to use the normalized gradient disparity $\tilde{\mathfrak{D}} = \overline{\mathfrak{D}}/\sqrt{d}$. We observe in Fig. 3.20 that both the normalized[10] gradient disparity and test error decrease with the network width (the *scale* is a hyper-parameter used to change both the number of channels and hidden units in each configuration).

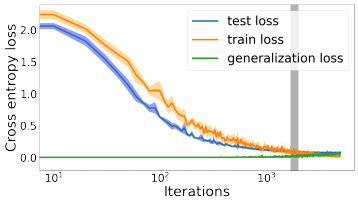
Fig. 3.21 shows the results for a 4-layer fully connected neural network, which is trained on the entire CIFAR-10 training set. We observe that gradient disparity reflects the test error at the early stages of training quite well. In the later stages of training we observe that the ranking of gradient disparity values for different label noise levels matches with the ranking of generalization losses and errors. In all experiments the gradient disparity is indeed very informative about the test error.

The test error decreases with the size of the training set (Fig. 3.6 (bottom)) and a reliable signal of overfitting should therefore reflect this property. Many of the previous metrics fail to do so, as shown by [Neyshabur et al. 2017a; Nagarajan and Kolter 2019]. In contrast, gradient disparity indeed decreases with the training set size, as shown in Fig. 3.6 (top) and Fig. 3.22. In Fig. 3.6, we study the effect of data augmentation [Shorten and Khoshgoftaar 2019], which is one of the popular techniques used to reduce overfitting given limited labeled data. Consistently with the rest of the chapter, we observe a strong positive correlation ($\rho = 0.979$) between the test error and gradient disparity for networks that are trained with data augmentation. Moreover, we observe that applying data augmentation decreases the values of both gradient disparity and the test error.

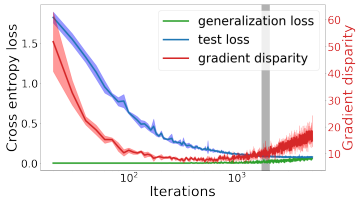
Fig. 3.22 shows the test error and gradient disparity for networks that are trained with different training set sizes. In Fig. 3.23, we observe that, as discussed in Section 3.1.6, gradient disparity, similarly to the test error, increases with the batch size for not too large batch sizes. As expected, when the batch size is very large (512 for the CIFAR-10 experiment and 256 for the CIFAR-100 experiments) gradient disparity starts to decrease, because gradient vectors are averaged over a large batch. Note that even with such large batch sizes, gradient disparity

[9] https://www.cs.toronto.edu/~kriz/cifar.html

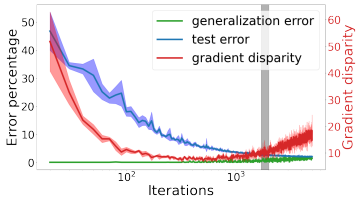
[10] Note that the normalization with respect to the number of parameters is different than the normalization mentioned in Section 3.E.1 which was with respect to the loss values. The value of gradient disparity reported everywhere is the re-scaled gradient disparity; further if comparison between two different architectures is taking place the normalization with respect to dimensionality will also take place.



(a) Cross entropy loss



(b) $\overline{\mathfrak{D}}$ vs. loss



(c) $\overline{\mathfrak{D}}$ vs. error

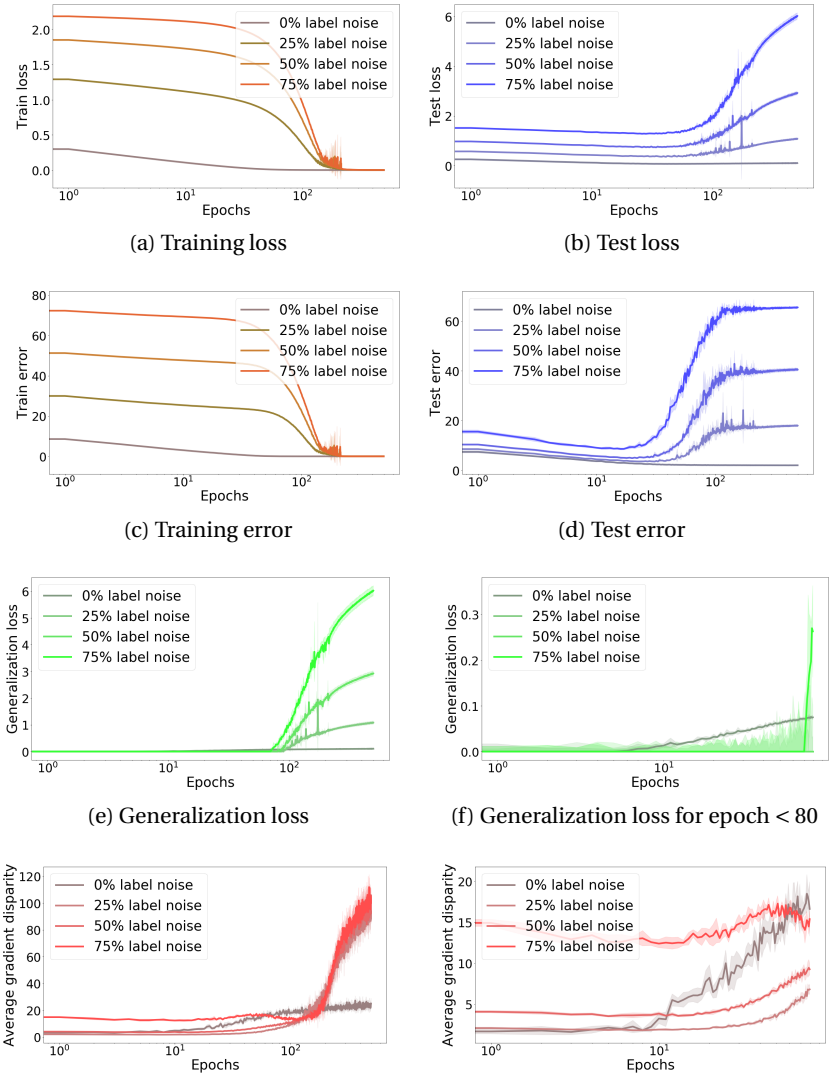
Figure 3.16: The cross entropy loss, the error percentage and the average gradient disparity during training for an AlexNet trained on a subset of 12.8 k points of the MNIST training set (the parameters are initialized according to the He [He et al. 2015] method with Normal distributions). For this experiment, $\rho_{\overline{\mathfrak{D}},\text{gen loss}} = 0.465$ and $\rho_{\overline{\mathfrak{D}},\text{gen error}} = 0.457$. The blue, orange, green, and red curves are the test loss/error, train loss/error, generalization loss/error, and the average gradient disparity $\overline{\mathfrak{D}}$, respectively.

correctly detects the early stopping time, although it can no longer be compared to the value of gradient disparity found with other batch sizes.

3.G.3 CIFAR-100 Experiments

Fig. 3.24 shows the results for a ResNet-18 that is trained on the CIFAR-100 training set[11]. Clearly, the model is not sufficient to learn the complexity of the CIFAR-100 dataset: It has 99% error for the network with 0% label noise, as if it had not learned anything about the dataset and is just making a random guess for classification (because there are 100 classes, random guessing would give 99% error on average). We observe from Fig. 3.24 (f) that as training progresses, the network overfits more, and the generalization error increases. Although the test error is high (above 90%), very surprisingly for this example, the networks with higher label noise level have a lower test loss and error (Fig. 3.24 (b) and (d)). Quite interestingly, gradient disparity (Fig. 3.24 (g)) captures also this surprising trend as well.

[11] https://www.cs.toronto.edu/~kriz/cifar.html



(a) Training loss

(b) Test loss

(c) Training error

(d) Test error

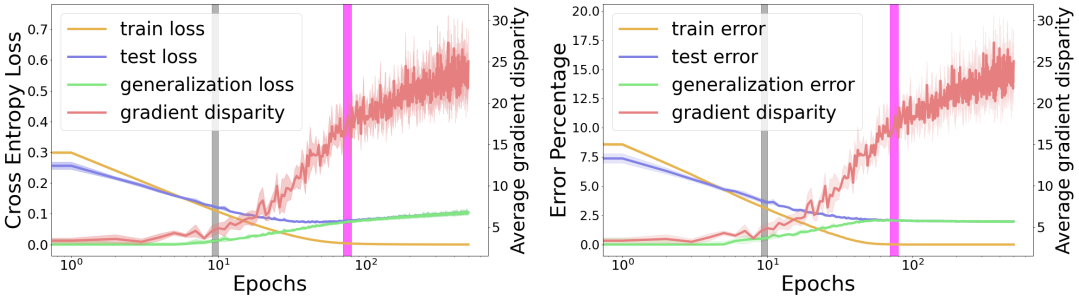
(e) Generalization loss

(f) Generalization loss for epoch < 80

(g) $\overline{\mathfrak{D}}$

(h) $\overline{\mathfrak{D}}$ for epoch < 80

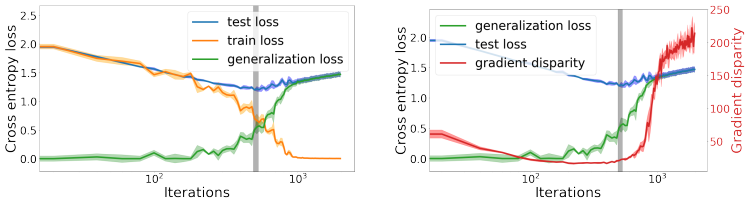
Figure 3.17: The cross entropy loss, error percentage, and average gradient disparity during training with different amounts of randomness in the training labels for a 4-layer fully connected neural network with 500 hidden units trained on the entire MNIST dataset. The parameter initialization is the He initialization with normal distribution.



(a) Loss vs gradient disparity

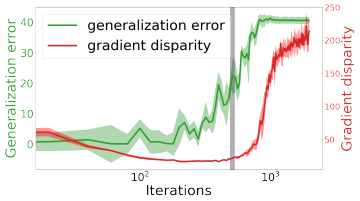
(b) Error vs gradient disparity

Figure 3.18: The cross entropy loss, error percentage, and average gradient disparity during training for a 4-layer fully connected neural network with 500 hidden units trained on the entire MNIST dataset with 0% label noise. The parameter initialization is the He initialization with normal distribution. Pearson's correlation coefficient ρ between $\overline{\mathfrak{D}}$ and generalization loss/error over all the training iterations are $\rho_{\overline{\mathfrak{D}},\text{gen loss}} = 0.967$ and $\rho_{\overline{\mathfrak{D}},\text{gen error}} = 0.734$. The gray vertical bar indicates when GD increases for 5 epochs from the beginning of training. The magenta vertical bar indicates when GD increases for 5 *consecutive* epochs. We observe that the gray bar signals when overfitting is starting, which is when the training and testing curves are starting to diverge. The magenta bar would be a good stopping time, because if we train beyond this point, although the test error remains the same, the test loss would increase, which would result in overconfidence on wrong predictions.



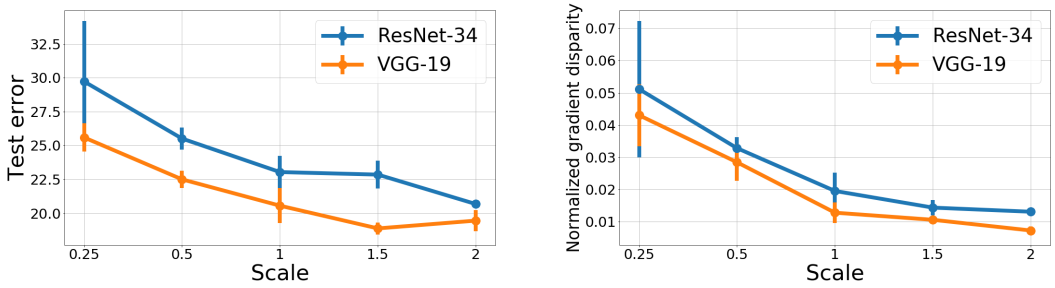
(a) Cross entropy loss

(b) $\overline{\mathfrak{D}}$ vs. loss

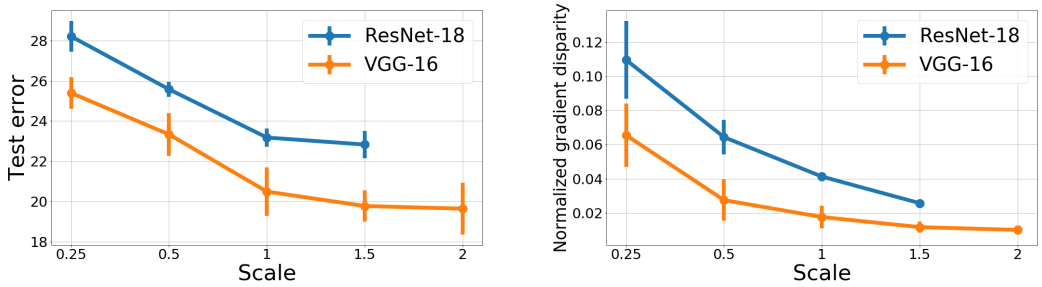


(c) $\overline{\mathfrak{D}}$ vs. error

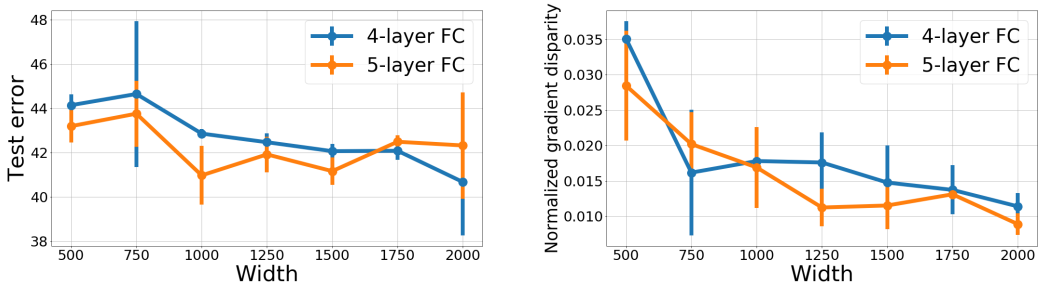
Figure 3.19: The cross entropy loss, the error percentage and the average gradient disparity during training for a ResNet-18 trained on a subset of 12.8 k points of the CIFAR-10 training set (the parameter initialization is Xavier [Glorot and Bengio 2010]). Pearson's correlation coefficient ρ between $\overline{\mathfrak{D}}$ and generalization loss/error over all the training iterations are $\rho_{\overline{\mathfrak{D}},\text{gen loss}} = 0.755$ and $\rho_{\overline{\mathfrak{D}},\text{gen error}} = 0.846$.



(a) $\rho_{\tilde{\mathfrak{D}},\mathrm{TL}} = 0.970$ and $\rho_{\tilde{\mathfrak{D}},\mathrm{TE}} = 0.939$

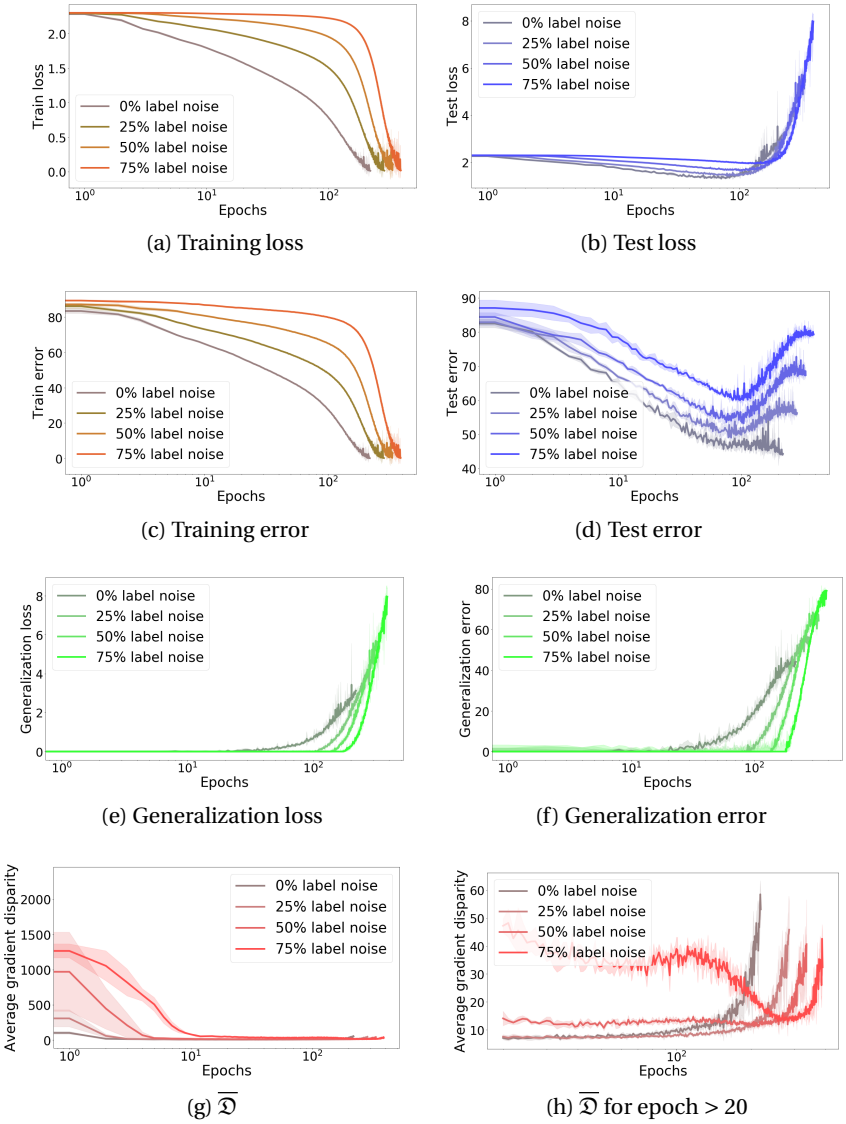


(b) $\rho_{\tilde{\mathscr{D}},\mathrm{TL}} = 0.655, \rho_{\tilde{\mathscr{D}},\mathrm{TE}} = 0.958$



(c) $\rho_{\tilde{\mathscr{D}},\mathrm{TL}} = 0.771, \rho_{\tilde{\mathscr{D}},\mathrm{TE}} = 0.601$

Figure 3.20: Test error and normalized gradient disparity for networks trained on the CIFAR-10 dataset with different number of channels and hidden units for convolutional neural networks (CNN) (scale = 1 recovers the original configurations) and fully connected neural networks (FC). The correlation between normalized gradient disparity and test loss $\rho_{\tilde{\mathfrak{D}},\mathrm{TL}}$ and between normalized gradient disparity and test error $\rho_{\tilde{\mathfrak{D}},\mathrm{TE}}$ are reported in the captions.



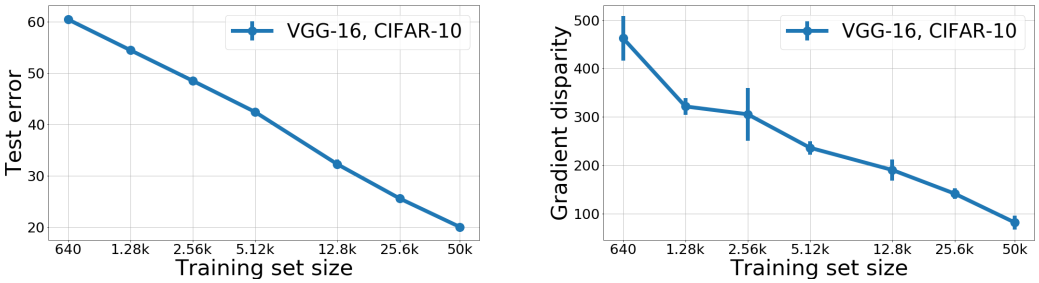
(a) Training loss (b) Test loss

(c) Training error (d) Test error

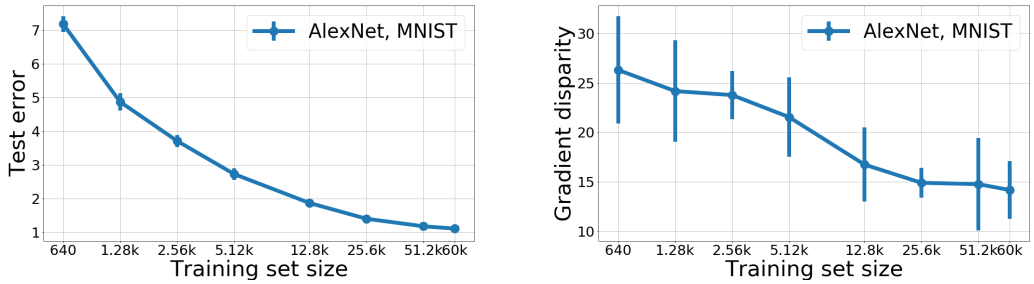
(e) Generalization loss (f) Generalization error

(g) $\overline{\mathfrak{D}}$ (h) $\overline{\mathfrak{D}}$ for epoch > 20

Figure 3.21: The cross entropy loss, error percentage, and average gradient disparity during training with different amounts of randomness in the training labels for a 4-layer fully connected neural network with 500 hidden units trained on the entire CIFAR-10 dataset. The parameter initialization is the Xavier initialization with uniform distribution.

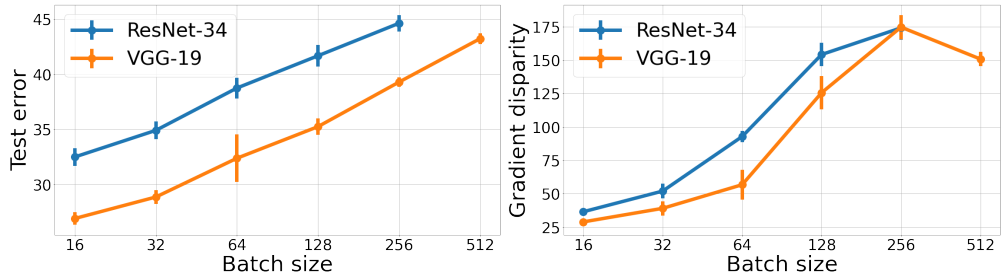


(a) VGG-16, CIFAR-10, $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.972$

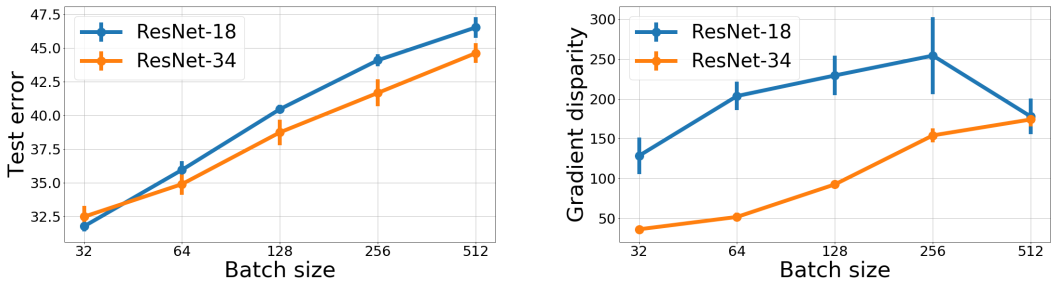


(b) ALexNet, MNIST, $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.929$

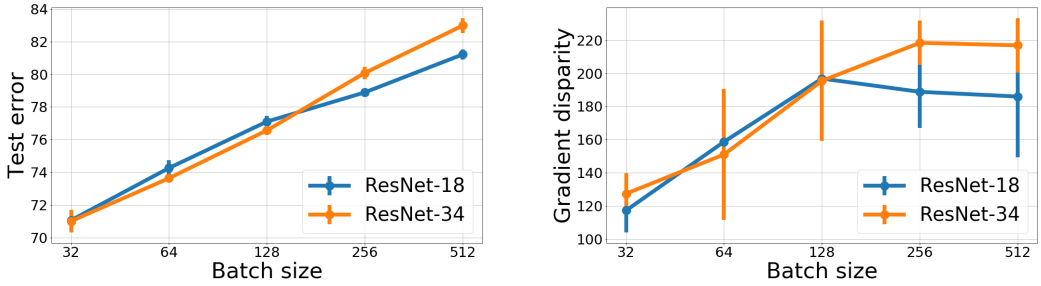
Figure 3.22: Test error and gradient disparity for networks that are trained with different training set sizes. The training is stopped when the training loss is below 0.01.



(a) CIFAR-10, $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.893$

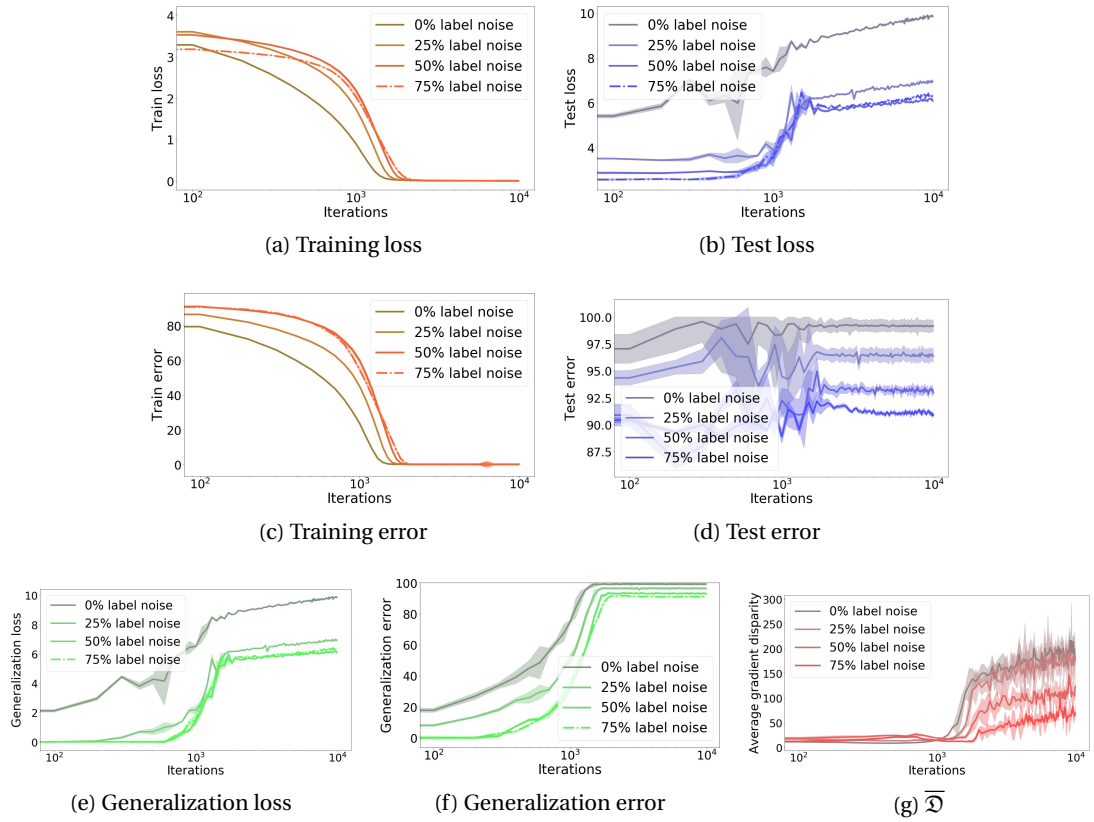


(b) CIFAR-10, $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.631$



(c) CIFAR-100, $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.909$

Figure 3.23: Test error and gradient disparity for networks that are trained with different batch sizes trained on 12.8 k points of the CIFAR-10 and CIFAR-100 datasets. The training is stopped when the training loss is below 0.01.



(a) Training loss (b) Test loss

(c) Training error (d) Test error

(e) Generalization loss (f) Generalization error (g) $\overline{\mathfrak{D}}$

Figure 3.24: The cross entropy loss, error percentage, and average gradient disparity during training with different amounts of randomness in the training labels for a ResNet-18 trained on the CIFAR-100 training set. The parameter initialization is the Xavier initialization.

3.H Beyond SGD

In the following, we discuss how the analysis of Section 3.1.4 can be extended to other optimizers (refer to [Ruder 2016] for an overview on popular optimizers).

3.H.1 SGD with Momentum

The momentum method [Qian 1999] is a variation of SGD which adds a fraction of the update vector of the previous step to the current update vector to accelerate SGD:

$$v^{(t+1)} = \eta v^{(t)} + \gamma g^{(t)},$$
$$\theta^{(t+1)} = \theta^{(t)} - v^{(t+1)},$$

where $g^{(t)}$ is either g_1 or g_2 depending on the selection of the batch S_1 or S_2 for the current update step. As $v^{(t)}$ remains the same for either choice, the KL-divergence between Q_1 and Q_2 for SGD with momentum, is the same as Eq. (3.5).

3.H.2 Adagrad

Adagrad [Duchi et al. 2011] performs update steps with a different learning rate for each individual parameter. By denoting each coordinate of the parameter vector θ by d, one update step of the Adagrad algorithm is

$$\theta_d^{(t+1)} = \theta_d^{(t)} - \frac{\gamma}{\sqrt{G_{dd}^{(t)} + \epsilon}} g_d^{(t)}, \tag{3.17}$$

where the vector $g^{(t)}$ is either g_1 or g_2 depending on the selection of the batch for the current update step, and $G_{dd}^{(t)}$ is the accumulative squared norm of the gradients up until iteration t. Hence, for Adagrad, Eq. (3.5) is replaced by

$$\begin{aligned} \mathrm{KL}(Q_1||Q_2) &= \frac{1}{2}\frac{\gamma^2}{\sigma^2} \left\| \frac{1}{G^{(t)} + \epsilon} \odot \left(g_1 - g_2\right) \right\|_2^2 \\ &\leq \frac{1}{2}\frac{\gamma^2}{\sigma^2} \left\| \frac{1}{G^{(t)} + \epsilon} \right\|_2^2 \left\| g_1 - g_2 \right\|_2^2, \end{aligned} \tag{3.18}$$

where $\odot$ denotes the element-wise product of two vectors, where division is also taken element-wise and where ϵ is a small positive constant that avoids a possible division by 0. To compare the upper bound in Theorem 1 from one iteration to the next one (as needed to determine the early stopping moment in Section 3.1.5), gradient disparity is not the only factor in Eq. (3.18) that evolves over time. Indeed $G^{(t)}$ is an increasing function of t. However, after a few iterations when the gradients become small, this value becomes approximately constant (the initial gradient values dominate the sum in $G^{(t)}$). Then the right hand side of Eq. (3.18) varies mostly as a function of gradient disparity, and therefore gradient disparity approximately tracks down the generalization penalty upper bound.

3.H.3 Adadelta and RmsProp

Adadelta [Zeiler 2012] is an extension of Adagrad, which computes a decaying average of the past gradient vectors instead of the accumulative squared norm of the gradients during the previous update steps. $G_{dd}^{(t)}$ in Eq. (3.17) is then replaced by $v_d^{(t+1)}$ where $v_d^{(t+1)} = \eta v_d^{(t)} + (1-\eta)(g_d^{(t)})^2$. As training proceeds, the gradient magnitude decreases. Also, η is usually close to 1. Therefore, the dominant term in $v_d^{(t+1)}$ becomes $\eta v_d^{(t)}$. Then, if we approximate $v_1^{(t+1)} = \eta v^{(t)} + (1-\eta)\left(g_1\right)^2 \approx \eta v^{(t)} + (1-\eta)\left(g_2\right)^2 = v_2^{(t+1)}$ (squares are done element-wise), then for Adadelta we have

$$\mathrm{KL}(Q_1||Q_2) \leq \frac{1}{2}\frac{\gamma^2}{\sigma^2}\left\|\frac{1}{v^{(t+1)}+\epsilon}\right\|_2^2 \left\|g_1 - g_2\right\|_2^2, \tag{3.19}$$

where again the division is done element-wise. The denominator in Eq. (3.19) is smaller than the denominator in Eq. (3.18). In both equations, the first non-constant factor in the upper bound of $\mathrm{KL}(Q_1||Q_2)$ decreases as a function of t, and therefore an increase in the value of $\mathrm{KL}(Q_1||Q_2)$ should be accounted for by an increase in the value of gradient disparity. Moreover, as training proceeds, gradient magnitudes decrease and the first factor on the upper bound of Eq. (3.18) and Eq. (3.19) becomes closer to a constant. Therefore, an upper bound on the generalization penalties can be tracked by gradient disparity. The update rule of RmsProp[12] is very similar to Adadelta, and the same conclusions can be made.

3.H.4 Adam

Adam [Kingma and Ba 2014] combines Adadelta and momentum by storing an exponentially decaying average of the previous gradients and squared gradients:

$$\begin{aligned}
m^{(t+1)} &= \beta_1 m^{(t)} + (1-\beta_1) g^{(t)}, \qquad v^{(t+1)} = \beta_2 v^{(t)} + (1-\beta_2)\left(g^{(t)}\right)^2, \\
\hat{m}^{(t+1)} &= \frac{m^{(t+1)}}{1-\left(\beta_1\right)^t}, \qquad \hat{v}^{(t+1)} = \frac{v^{(t+1)}}{1-\left(\beta_2\right)^t}, \\
\theta^{(t+1)} &= \theta^{(t)} - \frac{\gamma}{\sqrt{\hat{v}^{(t+1)}}+\epsilon}\hat{m}^{(t+1)}.
\end{aligned}$$

All the operations in the above equations are done element-wise. As β_2 is usually very close to 1 (around 0.999), and as squared gradient vectors at the current update step are much smaller than the accumulated values during the previous steps, we approximate: $v_1^{(t+1)} = \beta_2 v^{(t)} + (1-\beta_2)\left(g_1\right)^2 \approx \beta_2 v^{(t)} + (1-\beta_2)\left(g_2\right)^2 = v_2^{(t+1)}$ (squares are done element-wise). Hence, Eq. (3.5) becomes

$$\mathrm{KL}(Q_1||Q_2) \leq \frac{1}{2}\frac{\gamma^2}{\sigma^2}\frac{1-\beta_1}{1-(\beta_1)^t}\left\|\frac{1}{\sqrt{\hat{v}^{(t+1)}}+\epsilon}\right\|_2^2 \left\|g_1 - g_2\right\|_2^2. \tag{3.20}$$

The first non-constant factor in the equation above decreases with t (because $\beta_1 < 1$). However, it is not clear how the second factor varies as training proceeds. Therefore, unlike previous

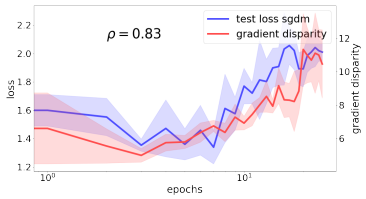
[12] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

optimizers, it is more hazardous to claim that the factors other than gradient disparity in Eq. (3.20) become constant as training proceeds. Hence, tracking only gradient disparity for the Adam optimizer may be insufficient. This is empirically investigated in the next sub-section.

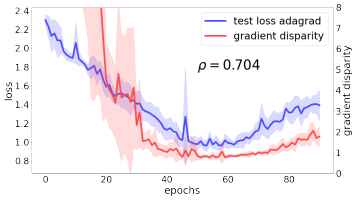
3.H.5 Experiments

Fig. 3.25 shows gradient disparity and the test loss curves during the course of training for adaptive optimizers. The epoch in which the fifth increase in the value of the test loss and gradient disparity has happened is shown in the caption of each experiment. We observe that the two suggested epochs for stopping the optimization (the one suggested by gradient disparity (GD) and the other one suggested by test loss) are extremely close to each other, except in Fig. 3.25 (c) where the fifth epoch with an increase in the value of gradient disparity is much later than the epoch with the fifth increase in the value of test loss. However, in this experiment, there is a 23% improvement in the test accuracy if the optimization is stopped according to GD compared to test loss, due to many variations of the test loss compared to gradient disparity.

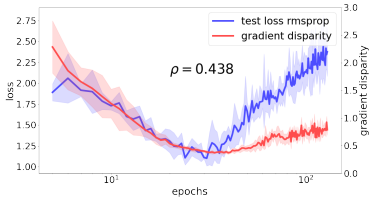
As an early stopping criterion, the increase in the value of gradient disparity coincides with the increase in the test loss in all our experiments presented in Fig. 3.25. In Fig. 3.25 (h), for the Adam optimizer, we observe that after around 20 epochs, the value of gradient disparity starts to decrease, whereas the test loss continues to increase. This mismatch between test loss and gradient disparity might result from other factors that appear in Eq. (3.20). Nevertheless, even in this experiment, the increase in the test loss and gradient disparity coincide, and hence gradient disparity can correctly detect early stopping time. These experiments are a first indication that gradient disparity can be used as an early stopping criterion for optimizers other than SGD.



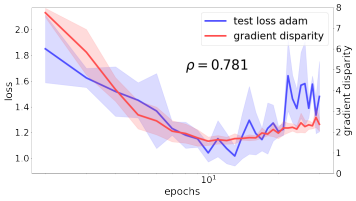
(a) SGD with Momentum, test loss epoch: 11, GD epoch: 10



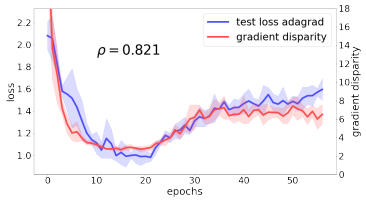
(b) Adagrad, test loss epoch: 19, GD epoch: 18



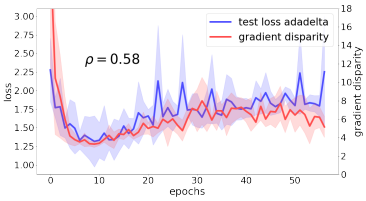
(c) RmsProp, test loss epoch: 15 (err: 54%), GD epoch: 36 (err: **31%**)



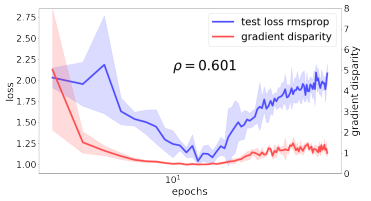
(d) Adam, test loss epoch: 19, GD epoch: 20



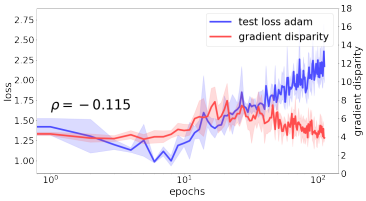
(e) Adagrad, test loss epoch: 21, GD epoch: 21



(f) Adadelta, test loss epoch: 12, GD epoch: 15



(g) RmsProp, test loss epoch: 20, GD epoch: 18



(h) Adam, test loss epoch: 12, GD epoch: 10

Figure 3.25: (a-d) VGG-19 configuration trained on 12.8 k training points of CIFAR-10 dataset. (e-h) VGG-11 configuration trained on 12.8 k points of the CIFAR-10 dataset. The training is stopped when the training loss gets below 0.01. The presented results are an average over 5 runs. The captions below each figure give the epoch number where test loss and gradient disparity have respectively been increased for 5 epochs from the beginning of training.

	Min	GD/Var	EB	GSNR	$g_i \cdot g_j$	sign($g_i \cdot g_j$)	cos($g_i \cdot g_j$)	Ω_c	OV	k-fold	No ES
TE	4.84	**4.84**	**4.84**	12.82	22.30	12.82	18.31	8.30	11.79	4.84	4.96
TL	0.18	**0.18**	**0.18**	0.46	0.82	0.46	0.69	0.32	0.38	0.18	0.22

(a) MNIST, AlexNet

	Min	GD/Var	EB	GSNR	$g_i \cdot g_j$	sign($g_i \cdot g_j$)	cos($g_i \cdot g_j$)	Ω_c	OV	k-fold	No ES
TE	13.76	**16.66**	24.63	35.68	37.92	24.63	35.68	29.40	34.36	17.86	25.72
TL	0.75	1.08	**0.86**	1.68	1.82	**0.86**	1.68	1.46	1.65	1.09	0.91

(b) MNIST, AlexNet, 50% random

	Min	GD/Var	EB	GSNR	$g_i \cdot g_j$	sign($g_i \cdot g_j$)	cos($g_i \cdot g_j$)	Ω_c	OV	k-fold	No ES
TE	45.54	**45.95**	61.76	70.46	70.46	55.84	67.09	67.37	70.61	51.64	64.19
TL	1.32	**1.45**	1.68	1.92	1.92	1.52	1.83	1.85	1.92	1.49	1.98

(c) CIFAR-10, ResNet-18

	Min	GD/Var	EB	GSNR	$g_i \cdot g_j$	sign($g_i \cdot g_j$)	cos($g_i \cdot g_j$)	Ω_c	OV	k-fold	No ES
TE	59.77	71.97	73.17	77.08	75.91	**65.80**	75.43	77.71	76.65	72.56	75.96
TL	1.75	2.00	2.03	2.12	2.13	**1.93**	2.07	2.13	2.10	2.02	2.30

(d) CIFAR-10, ResNet-18, 50% random

Table 3.15: Test error (TE) and test loss (TL) achieved by using various metrics as early stopping criteria. On the leftmost column, the minimum values of TE and TL over all the iterations are reported (which is not accessible during training). The results of 5-fold cross validation are reported on the right, which serve as a baseline. For each experiment, we have underlined those metrics that result in a better performance than 5-fold cross-validation. We observe that gradient disparity (GD) and variance of gradients (Var) consistently outperform k-fold cross-validation, unlike other metrics. On the rightmost column (No ES) we report the results without performing early stopping (ES) (training is continued until the training loss is below 0.01).

3.I Comparison to Related Work

In Table 3.15, we compare gradient disparity (GD) to a number of metrics that were proposed either directly as an early stopping criterion, or as a generalization metric. For those metrics that were not originally proposed as early stopping criteria, we choose a similar method for early stopping as the one we use for gradient disparity. We consider two datasets (MNIST and CIFAR-10), and two levels of label noise (0% and 50%). Here is a list of the metrics that we compute in each setting (see Section 3.1.2 of the main chapter where we introduce each metric):

1. Gradient disparity (GD) (ours): we report the error and loss values at the time when the value of GD increases for the 5th time (from the beginning of the training).

2. The EB-criterion [Mahsereci et al. 2017]: we report the error and loss values when EB becomes positive.

3. Gradient signal to noise ratio (GSNR) [Liu et al. 2020a]: we report the error and loss values when the value of GSNR decreases for the 5th time (from the beginning of the training).

4. Gradient inner product, $g_i \cdot g_j$ [Fort et al. 2019]: we report the error and loss values when the value of $g_i \cdot g_j$ decreases for the 5th time (from the beginning of the training).

5. Sign of the gradient inner product, $\text{sign}(g_i \cdot g_j)$ [Fort et al. 2019]: we report the error and loss values when the value of $\text{sign}(g_i \cdot g_j)$ decreases for the 5th time (from the beginning of the training).

6. Cosine similarity between gradient vectors, $\cos(g_i \cdot g_j)$ [Fort et al. 2019]: we report the error and loss values when the value of $\cos(g_i \cdot g_j)$ decreases for the 5th time (from the beginning of the training).

7. Variance of gradients (Var) [Negrea et al. 2019]: we report the error and loss values when the value of Var increases for the 5th time (from the beginning of the training). Variance is computed over the same number of batches used to compute gradient disparity, in order to compare metrics given the same computational budget.

8. Average gradient alignment within the class Ω_c [Mehta et al. 2020]: we report the error and loss values when the value of Ω_c decreases for the 5th time (from the beginning of the training).

9. Optimization variance (OV) [Zhang et al. 2021c]: we report the error and loss values when the value of OV increases for the 5th time (from the beginning of the training).

On the leftmost column of Table 3.15, we report the minimum values of the test error and the test loss over all the iterations, which may not necessarily coincide. For instance, in setting (c), the test error is minimized at iteration 196, whereas the test loss is minimized at iteration 126. On the rightmost column of Table 3.15 we report the values of the test error and loss when no early stopping is applied and the training is continued until training loss value is below 0.01. Next to this, we report the values of the test error and the test loss when using 5-fold cross-validation, which serves as a baseline. We have underlined the metrics that outperform k-fold cross validation. We observe that the only metrics that consistently outperform k-fold CV are GD and Variance of gradients.

The EB-criterion, $\text{sign}(g_i \cdot g_j)$, and $\cos(g_i \cdot g_j)$ are metrics that perform quite well as early stopping criteria, although not as well as GD and Var. In Section 3.I.1, we observe that these metrics are not informative of the label noise level, contrary to gradient disparity.

It is interesting to observe that gradient disparity and variance of gradients produce the exact same results when used as early stopping criteria (Table 3.15). Moreover, these two are the only metrics that consistently outperform k-fold cross-validation. However, in Section 3.I.2, we observe that the correlation between gradient disparity and the test loss is in general larger than the correlation between variance of gradients and the test loss.

3.I.1 Capturing Label Noise Level

In this section, we show in particular three metrics that even though perform relatively well as early stopping criteria, fail to account for the level of label noise, contrary to gradient disparity.

- The sign of the gradient inner product, $\text{sign}(g_i \cdot g_j)$, should be inversely related to the test loss; it should decrease when overfitting increases. However, we observe that the value of $\text{sign}(g_i \cdot g_j)$ is larger for the setting with the higher label noise level; it incorrectly detects the setting with the higher label noise level as the setting with the better generalization performance (see Fig. 3.26).
- The EB-criterion should be larger for settings with more overfitting. In most stages of training, the EB-criterion does not distinguish between settings with different label noise levels, contrary to gradient disparity (see Fig. 3.26). At the end of the training, the EB-criterion even mistakenly signals the setting with the higher label noise level as the setting with the better generalization performance.
- The cosine similarity between gradient vectors, $\cos(g_i \cdot g_j)$, should decrease when overfitting increases and therefore with the level of label noise in the training data. But $\cos(g_i \cdot g_j)$ does not appear to be sensitive to the label noise level, and in some cases (Fig. 3.27 (a)) it even increases with the noise level. Gradient disparity is much more informative of the label noise level compared to cosine similarity and the correlation between gradient disparity and the test error is larger than the correlation between cosine similarity and the test accuracy (see Fig. 3.27).

3.I.2 Gradient Disparity versus Variance of Gradients

It has been shown that generalization is related to gradient alignment experimentally in [Fort et al. 2019], and to variance of gradients theoretically in [Negrea et al. 2019]. Gradient disparity can be viewed as bringing the two together. Indeed, one can check that $\mathbb{E}\left[\mathfrak{D}_{i,j}^2\right] = 2\sigma_g^2 + 2\mu_g^T\mu_g - 2\mathbb{E}\left[g_i^T g_j\right]$, given that $\mu_g = \mathbb{E}[g_i] = \mathbb{E}[g_j]$ and $\sigma_g^2 = \text{tr}\left(\text{Cov}\left[g_i\right]\right) = \text{tr}\left(\text{Cov}\left[g_j\right]\right)$. This shows that gradient variance σ_g^2 and gradient alignment $g_i^T g_j$ both appear as components of gradient disparity. We conjecture that the dominant term in gradient disparity is the variance of gradients, hence as early stopping criteria these two metrics almost always signal overfitting simultaneously. This is indeed what our experiments show; we show that variance of gradients is also a very promising early stopping criterion (Table 3.15). However, because of the additional term in gradient disparity (the gradients inner product), gradient disparity emphasizes the alignment or misalignment of the gradient vectors. This could be the reason why gradient disparity in general outperforms variance of gradients in tracking the value of the generalization loss; the positive correlation between gradient disparity and the test loss is often larger than the positive correlation between variance of gradients and the test loss (Table 3.16).

Setting	$\rho_{\overline{\mathfrak{D}},\mathrm{TL}}$	$\rho_{\mathrm{Var},\mathrm{TL}}$
AlexNet, MNIST	**0.433**	0.169
AlexNet, MNIST, 50% random labels	**0.535**	0.161
VGG-16, CIFAR-10	0.190	**0.324**
VGG-16, CIFAR-10, 50% random labels	**0.634**	0.623
VGG-19, CIFAR-10	**0.685**	0.508
VGG-19, CIFAR-10, 50% random labels	**0.748**	0.735
ResNet-18, CIFAR-10	**0.975**	0.958
ResNet-18, CIFAR-10, 50% random labels	**0.471**	0.457

Table 3.16: Pearson's correlation coefficient between gradient disparity ($\overline{\mathfrak{D}}$) and test loss (TL) over the training iterations is compared to the correlation between variance of gradients (Var) and test loss.

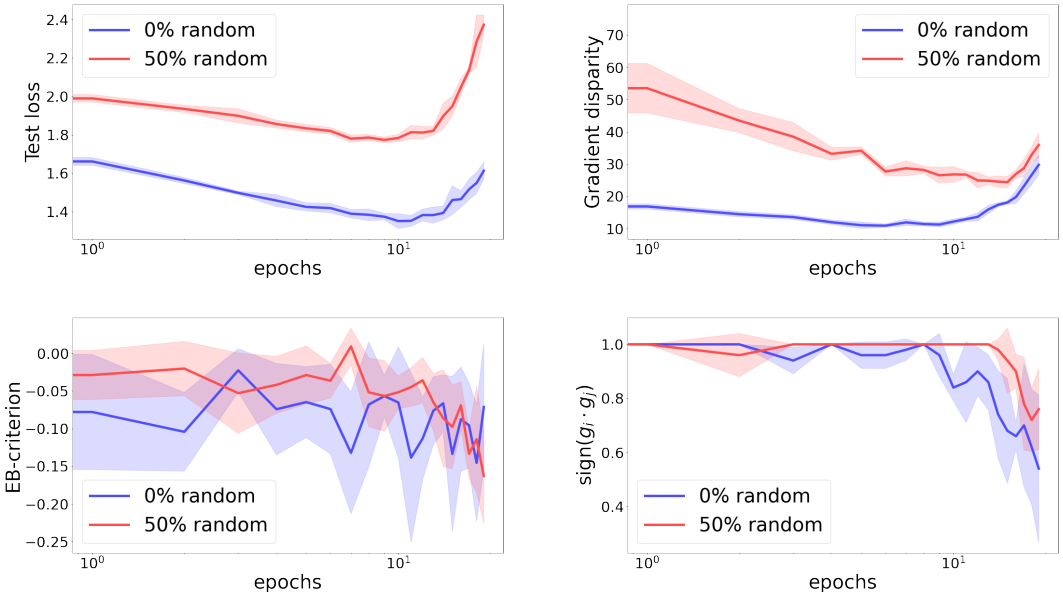
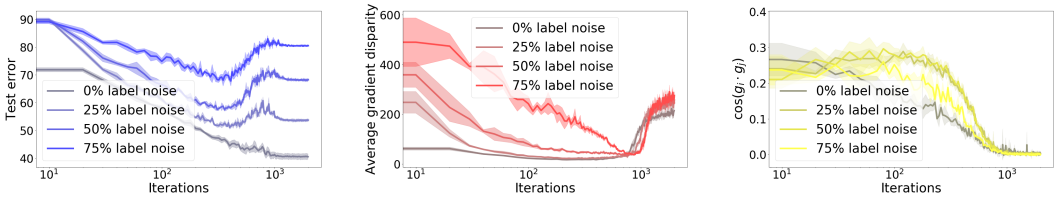
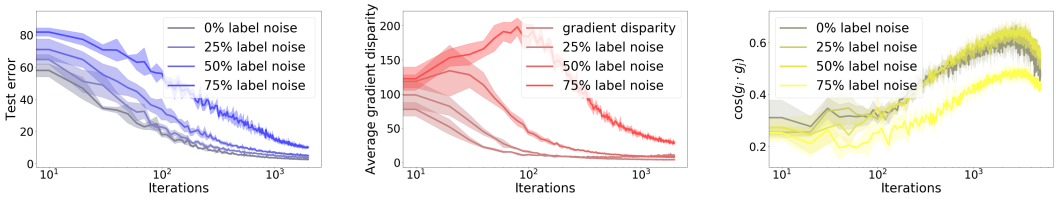


Figure 3.26: Test loss, gradient disparity, EB-criterion [Mahsereci et al. 2017], and sign($g_i \cdot g_j$) for a ResNet-18 trained on the CIFAR-10 dataset, with 0% and 50% random labels. Gradient disparity, contrary to EB-criterion and sign($g_i \cdot g_j$), clearly distinguishes the setting with correct labels from the setting with random labels.



(a) CIFAR-10, ResNet-18



(b) MNIST, AlexNet

Figure 3.27: The test error (TE), average gradient disparity ($\overline{\mathfrak{D}}$), and cosine similarity ($\cos(g_i \cdot g_j)$) during training with different amounts of randomness in the training labels for two sets of experiments. (a) ResNet-18 trained on 12.8k points of the CIFAR-10 training set. The Pearson correlation coefficient between test accuracy (TA) and cosine similarity (cos) over all levels of randomness and over all the iterations is $\rho_{\cos,\mathrm{TA}} = -0.0088$, whereas the correlation between test error/generalization error and gradient disparity is $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.2029$ and $\rho_{\overline{\mathfrak{D}},\mathrm{GE}} = 0.5268$, respectively. (b) AlexNet configuration trained on 12.8k points of the MNIST dataset. The correlation between the test accuracy and cosine similarity is $\rho_{\cos,\mathrm{TA}} = 0.7521$, which is positive and relatively high for this experiment. Yet, it is still lower than the correlation between test error and gradient disparity which is $\rho_{\overline{\mathfrak{D}},\mathrm{TE}} = 0.8019$.

4 Leveraging Unlabeled Data to Track Memorization

Deep neural networks may easily memorize noisy labels present in real-world data, which degrades their ability to generalize. It is therefore important to track and evaluate the robustness of models against noisy label memorization. In this chapter[1], we propose a metric, called *susceptibility*, to gauge such memorization for neural networks. Susceptibility is simple and easy to compute during training. Moreover, it does not require access to ground-truth labels and it only uses unlabeled data. We empirically show the effectiveness of our metric in tracking memorization on various architectures and datasets and provide theoretical insights into the design of the susceptibility metric. Finally, we show through extensive experiments on datasets with synthetic and real-world label noise that one can utilize susceptibility and the overall training accuracy to distinguish models that maintain a low memorization on the training set and generalize well to unseen clean data.

4.1 Introduction

Deep neural networks are prone to memorizing noisy labels in the training set, which are inevitable in many real-world applications [Frénay and Verleysen 2013; Zhang et al. 2016b; Arpit et al. 2017; Song et al. 2022; Nigam et al. 2020; Han et al. 2020; Zhang et al. 2021a; Wei et al. 2021]. Given a new dataset that contains clean and noisy labels, one refers to the subset of the dataset with correct labels (respectively, with incorrect labels due to noise), as the clean (respectively, noisy) subset. When neural networks are trained on such a dataset, it is important to find the sweet spot from no fitting at all to fitting every sample. Indeed, fitting the clean subset improves the generalization performance of the model (measured by the classification accuracy on unseen clean data), but fitting the noisy subset, referred to as "memorization"[2], degrades its generalization performance. New methods have been introduced to address this issue (for example, robust architectures [Xiao et al. 2015; Li et al. 2020b], robust objective functions [Li et al. 2019a; Ziyin et al. 2020], regularization techniques [Zhang et al. 2017;

[1] This chapter is based on [Forouzesh et al. 2023].

[2] Fitting samples that have incorrect random labels is done by memorizing the assigned label for each particular sample. Hence, we refer to it as memorization, in a similar spirit as Feldman and Zhang [2020].

Pereyra et al. 2017; Chen et al. 2019; Harutyunyan et al. 2020], and sample selection methods [Nguyen et al. 2019]), but their effectiveness cannot be assessed without oracle access to the ground-truth labels to distinguish the clean and the noisy subsets, or without a clean test set.

Our goal in this chapter is to track memorization during training without any access to ground-truth labels. To do so, we sample a subset of the input data and label it uniformly at random from the set of all possible labels. The samples can be taken from unlabeled data, which is often easily accessible, or from the available training set with labels removed. This new held-out randomly-labeled set is created for evaluation purposes only, and does not affect the original training process.

First, we compare how different models fit the held-out randomly-labeled set after multiple steps of training on it. We observe empirically that models that have better accuracy on unseen clean test data show more resistance towards memorizing the randomly-labeled set. This resistance is captured by the number of steps required to fit the held-out randomly-labeled set. In addition, through our theoretical convergence analysis on this set, we show that models with high/low test accuracy are resistant/susceptible to memorization, respectively.

Building on this result, we then propose an easy-to-compute metric that we call *susceptibility to noisy labels*, which is the difference in the objective function of a *single* mini-batch from the held-out randomly-labeled set, before and after taking an optimization step on it. At each step during training, the larger this difference is, the more the model is affected by (and is therefore susceptible to) the noisy labels in the mini-batch. Fig. 4.1 (bottom/middle left) provides an illustration of the susceptibility metric. We observe a strong correlation between the susceptibility and the memorization within the training set, which is measured by the fit on the noisy subset. We then show how one can utilize this metric and the overall training accuracy to distinguish models with a high test accuracy across a variety of state-of-the-art deep learning models, including DenseNet [Huang et al. 2017], EfficientNet [Tan and Le 2019], and ResNet [He et al. 2016a] architectures, and various datasets including synthetic and real-world label noise (Clothing-1M, Animal-10N, CIFAR-10N, Tiny ImageNet, CIFAR-100, CIFAR-10, MNIST, Fashion-MNIST, and SVHN), see Fig. 4.1 (right).

Our main contributions and takeaways are summarized below:

- We empirically observe and theoretically show that models with a high test accuracy are resistant to memorizing a randomly-labeled held-out set (Section 4.2 and Section 4.5).
- We propose the susceptibility metric, which is computed on a randomly-labeled subset of the available data. Our extensive experiments show that susceptibility closely tracks memorization of the noisy subset of the training set (Section 4.3).
- We observe that models which are trainable and resistant to memorization, i.e., having a high training accuracy and a low susceptibility, have high test accuracies. We leverage

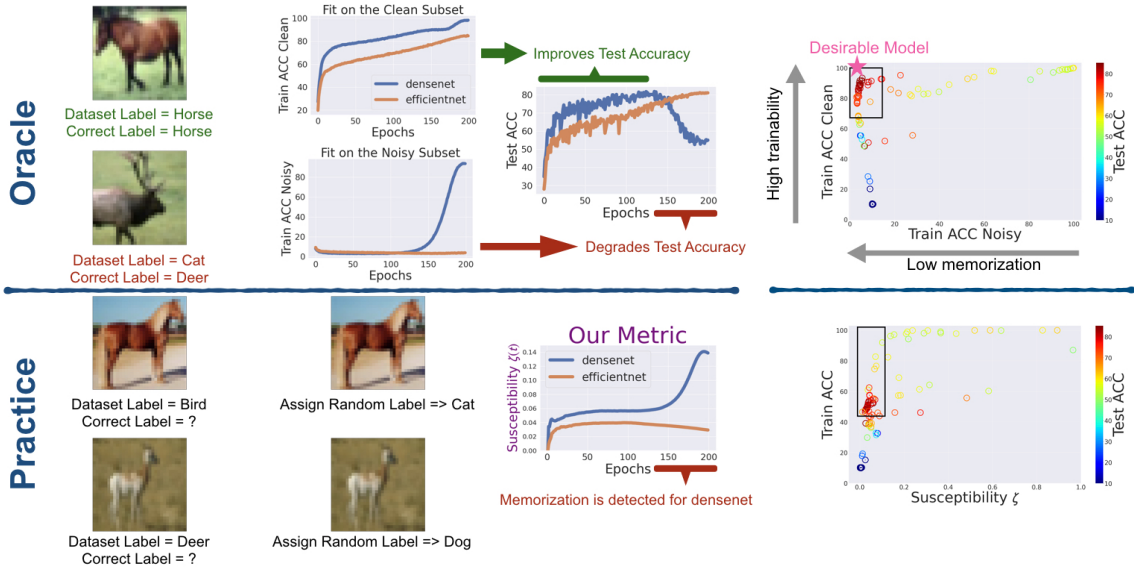


Figure 4.1: Models trained on CIFAR-10 with 50% label noise. **Top (Oracle access to ground-truth labels):** We observe that the fit on the clean subset of the training set (shown in the top left) and the fit on the noisy subset (located below the fit on the clean subset) affect the predictive performance (measured by the classification accuracy) on unseen clean test data differently. Fitting the clean (resp., noisy) subset improves (resp., degrades) test accuracy, as shown by the green (resp., red) arrow. With oracle access to the ground-truth label, one can therefore select models with a high fit on the clean subset and a low fit on the noisy subset, as it is done in the top right to find desirable models. **Bottom (Our approach in practice):** In practice however, the ground-truth label, and hence the fit on the clean and noisy subsets are not available. In this chapter, we propose the metric called susceptibility ζ to track the fit on the noisy subset of the training set. Susceptibility is computed using a mini-batch of data that is assigned with random labels independently from the dataset labels. We observe a strong correlation between susceptibility and memorization. Moreover, the susceptibility metric together with the training accuracy on the entire set, is used to recover models with low "memorization" (low fit on the noisy subset) and high "trainability" (high fit on the clean subset) without any ground-truth label oracle access. The average test accuracy of models in the top-left rectangle of the right figures are $77.93_{\pm 4.68}$% and $76.15_{\pm 6.32}$% for the oracle and our approach, respectively. Hence, our method successfully recovers desirable models.

this observation to propose a model-selection method in the presence of noisy labels (Section 4.4).

- We show through extensive experiments that our results hold for real-world label noise, and are persistent for various datasets, architectures, hyper-parameters, label noise levels, and label noise types (Section 4.6 and Section 4.7).

Related work Garg et al. [2021a] show that for models trained on a mixture of clean and noisy data, a low accuracy on the noisy subset of the training set and a high accuracy on the clean subset of the training set guarantee a low generalization error. With oracle access to the ground-truth labels to compute these accuracies, one can therefore predict which models perform better on unseen clean data. This is done in Fig. 4.1 (Oracle part). However, in practice,

there is no access to ground-truth labels. Our work therefore complements the results of [Garg et al. 2021a] by providing a practical approach (see Fig. 4.1 (Practice part)). Moreover, both our work and [Zhang et al. 2019c] emphasize that a desirable model differentiates between fitting clean and noisy samples. Zhang et al. [2019c] showcase this intuition by studying the drop in the training accuracy of models trained when label noise is injected in the dataset. We do it by studying the resistance/susceptibility of models to noisy labels of a held-out set. Zhang et al. [2019c] only study the training accuracy drop for settings where the available training set is clean. However, when the training set is itself noisy, which is the setting of interest in our work, we observe that this training accuracy drop does not predict memorization, unlike our susceptibility metric (see Fig. 4.8 (left) in Section 4.A). Furthermore, even though the metric proposed in Lu and He [2022] is rather effective as an early stopping criterion, it is not able to track memorization, contrary to susceptibility (see Fig. 4.8 (middle)). For a thorough comparison to other related work refer to Section 4.A.

4.2 Good Models are Resistant to Memorization

Consider models $f_\Theta(\mathbf{x})$ with parameter matrix Θ trained on a dataset $S = \{(\mathbf{x}_i, y_i)\}_1^n$, which is a collection of n input-output pairs that are drawn from a data distribution $\mathscr{D}$ over $\mathscr{X} \times \mathscr{Y}$ in a multi-class classification setting. We raise the following question: How much are these models resistant/susceptible to memorization of a new low-label-quality (noisy) dataset when they are trained on it? Intuitively, we expect a model with a high accuracy on correct labels to stay persistent on its predictions and hence to be *resistant* to the memorization of this new noisy dataset. We use the number of steps that it requires to fit the noisy dataset, as a measure for this resistance. The larger this number, the stronger the resistance (hence, the lower the susceptibility). In summary, we conjecture that models with high test accuracy on unseen clean data are more resistant to memorization of noisy labels, and hence take longer to fit a randomly-labeled held-out set.

To mimic a noisy dataset, we create a set with random labels. More precisely, we define a randomly-labeled held-out set of samples $\widetilde{S} = \{(\widetilde{\mathbf{x}}_i, \widetilde{y}_i)\}_1^{\widetilde{n}}$, which is generated from $\widetilde{n}$ unlabeled samples $\{\widetilde{\mathbf{x}}_i\}_1^{\widetilde{n}} \sim \mathscr{X}$ that are either accessible, or are a subset of $\{\mathbf{x}_i\}_1^n$. The outputs $\widetilde{y}_i$ of dataset $\widetilde{S}$ are drawn independently (both from each other and from labels in S) and uniformly at random from the set of possible classes. Therefore, to fit $\widetilde{S}$, the model needs to memorize the labels. We track the fit (memorization) of a model on $\widetilde{S}$ after $\widetilde{k}$ optimization steps on the noisy dataset $\widetilde{S}$.

We perform the following empirical test. In Fig. 4.2, we compare two deep neural networks. The first one has a high classification accuracy on unseen clean test data, whereas the other one has a low test accuracy. We then train both these models on a single sample with an incorrect label ($\widetilde{S}$ contains a single incorrectly-labeled sample). We observe that the model with a high test accuracy takes a long time (in terms of the number of training steps) to fit this sample; hence this model is *resistant* to memorization as the number of steps $\widetilde{k}$ required

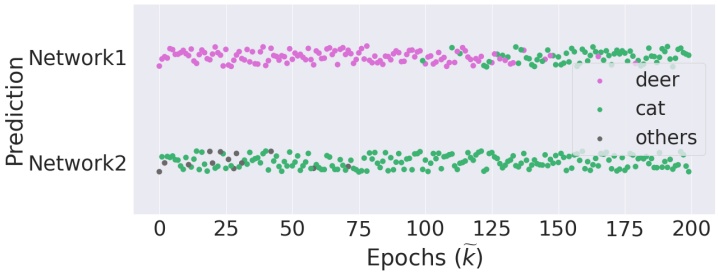


Figure 4.2: The evolution of output prediction of two neural networks during training on a single sample with a wrong randomly-assigned label (the assigned label is "cat" while the ground-truth label is "deer") versus the number of epochs (steps) $\widetilde{k}$. Network1 is a GoogLeNet [Szegedy et al. 2015] (pre-)trained on CIFAR-10 dataset with clean labels and has initial test accuracy of 95.36%. Network2 is a GoogLeNet (pre-)trained on CIFAR-10 dataset with 50% label noise level and has initial test accuracy of 58.35%. Network2 has a lower test accuracy compared to Network1, and we observe that it memorizes this new sample after only $\widetilde{k} = 2$ steps. In contrast, Network1 persists in predicting the correct output for this sample for longer and memorizes the new sample only after $\widetilde{k} = 99$ steps. More examples and an ablation study on the effect of calibration are given in Fig. 4.9 and Fig. 4.11 in Section 4.D, respectively.

to fit $\widetilde{S}$ is large. In contrast, the model with a low test accuracy memorizes this sample after taking only a few steps; hence this model is *susceptible* to memorization as $\widetilde{k}$ required to fit $\widetilde{S}$ is small[3]. This observation reinforces our intuition, that *good models*, measured by a high test accuracy, *are more resistant to memorization of noisy labels*.

Next, we further validate this observation theoretically by performing a convergence analysis on the held-out set $\widetilde{S}$, where the input samples $\{\widetilde{\mathbf{x}}_i\}_1^{\widetilde{n}}$ of $\widetilde{S}$ are the same as the inputs of dataset S (and hence $\widetilde{n} = n$). We consider the binary-classification setting. The output vector $\widetilde{\mathbf{y}} = \left(\widetilde{y}_1, \cdots, \widetilde{y}_n\right)^T$ is a vector of independent random labels that take values uniformly in $\{-1, 1\}$. Therefore, creating this dataset $\widetilde{S}$ does not require extra information, and in particular no access to the data distribution $\mathscr{D}$. Consider the network $f_\Theta(\mathbf{x})$ as a two-layer neural network with ReLU non-linearity (denoted by $\sigma(x) = \max\{x, 0\}$) and m hidden-units:

$$f_\Theta(\mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma\left(\theta_r^T \mathbf{x}\right),$$

where $\mathbf{x} \in \mathbb{R}^D$ is the input vector, $\Theta = (\theta_1, \cdots, \theta_m) \in \mathbb{R}^{D \times m}$, $\mathbf{a} = (a_1, \cdots, a_m) \in \mathbb{R}^m$ are the weight matrix of the first layer, and the weight vector of the second layer, respectively. For simplicity, the outputs associated with the input vectors $\{\mathbf{x}_i\}_1^n$ are denoted by vector $\mathbf{f}_\Theta \in \mathbb{R}^n$ instead of $\{f_\Theta(\mathbf{x}_i)\}_1^n$. The first layer weights are initialized as $\theta_r(0) \sim \mathcal{N}\left(\mathbf{0}, \kappa^2 \mathbf{I}\right), \quad \forall r \in [m]$, where $0 < \kappa \leq 1$ is the magnitude of initialization and $\mathcal{N}$ denotes the normal distribution. a_rs are independent random variables taking values uniformly in $\{-1, 1\}$, and are considered to be fixed throughout training.

[3] Interestingly, we observe that the situation is different for a correctly-labeled sample. Fig. 4.10 in Section 4.D shows that models with a higher test accuracy typically fit an unseen correctly-labeled sample faster than models with a lower test accuracy.

We define the objective function on datasets S and $\widetilde{S}$, respectively, as

$$\Phi(\Theta) = \frac{1}{2}\left\|\mathbf{f}_\Theta - \mathbf{y}\right\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}\left(f_\Theta(\mathbf{x}_i) - y_i\right)^2, \qquad \widetilde{\Phi}(\Theta) = \frac{1}{2}\sum_{i=1}^{n}\left(f_\Theta(\mathbf{x}_i) - \widetilde{y}_i\right)^2. \tag{4.1}$$

Label noise level (LNL) of S (and accordingly of the label vector $\mathbf{y}$) is the ratio n_1/n, where n_1 is the number of samples in S that have labels that are independently drawn uniformly in $\{-1,1\}$, and the remaining $n-n_1$ samples in S have the ground-truth label. Hence, the two extremes are S with LNL = 0, which is the clean dataset, and S with LNL = 1, which is a dataset with entirely random labels.

To study the convergence of different models on $\widetilde{S}$, we compute the objective function $\widetilde{\Phi}(\Theta)$ after $\widetilde{k}$ steps of training on the dataset $\widetilde{S}$. Therefore, the overall training/evaluation procedure is as follows; for $r \in [m]$, the first layer weights of the neural network are updated according to gradient descent:

$$\theta_r(t+1) - \theta_r(t) = -\gamma\frac{\partial\overline{\Phi}\left(\Theta(t)\right)}{\partial\theta_r},$$

where for $0 \le t < k$: $\overline{\Phi} = \Phi$, and for $k \le t < k+\widetilde{k}$: $\overline{\Phi} = \widetilde{\Phi}$, and γ is the learning rate. Note that we refer to $\Phi(\Theta(t))$ as $\Phi(t)$ and to $\widetilde{\Phi}(\Theta(t))$ as $\widetilde{\Phi}(t)$.

When a model fits the set $\widetilde{S}$, the value of $\widetilde{\Phi}$ becomes small, say below some threshold ε. The resistance of a model to memorization (fit) on the set $\widetilde{S}$ is then measured by the number of steps $\widetilde{k}^*(\varepsilon)$ such that $\widetilde{\Phi}(k+\widetilde{k}) \le \varepsilon$ for $\widetilde{k} > \widetilde{k}^*(\varepsilon)$. The larger (respectively, smaller) $\widetilde{k}^*$ is, the more the model is *resistant* (resp, susceptible) to this memorization. We can reason on the link between a good model and memorization from the following proposition.

Proposition 1 (Informal)**.** *The objective function $\widetilde{\Phi}$ at step $k+\widetilde{k}$ is a decreasing function of the label noise level (LNL) in S, and of the number of steps $\widetilde{k}$.*

Proposition 1 yields that models trained on S with a low LNL (which are models with a high test accuracy on clean data) push $\widetilde{\Phi}(k+\widetilde{k})$ to become large. The number of steps $\widetilde{k}^*(\varepsilon)$ should therefore be large for $\widetilde{\Phi}$ to become less than ε: these models resist the memorization of $\widetilde{S}$. Conversely, models trained on S with a high LNL (which are models with a low test accuracy), allow for $\widetilde{k}^*(\varepsilon)$ to be small and give up rapidly to the memorization of $\widetilde{S}$. These conclusions match the empirical observation in Fig. 4.2, and therefore further support the intuition that *good models are resistant to memorization.*

Refer to Section 4.5 for the formal theoretical results that produce Proposition 1.

4.3 Evaluating Resistance to Memorization

In Section 4.2, we showed that desirable models in terms of high classification accuracy on unseen clean data are resistant to the memorization of a randomly-labeled set. In this section, we describe a simple and computationally efficient metric to measure this resistance. To evaluate it efficiently at each step of the forward pass, we propose to take a *single* step on *multiple* randomly labeled samples, instead of taking *multiple* steps on a *single* randomly labeled sample (as done in Fig. 4.2). To make the metric even more computationally efficient, instead of the entire randomly labeled set $\widetilde{S}$ (as in our theoretical developments), we only take a single mini-batch of $\widetilde{S}$. For simplicity and with some abuse of notation, we still refer to this single mini-batch as $\widetilde{S}$.

To track the prediction of the model over multiple samples, we compare the objective function ($\widetilde{\Phi}$) on $\widetilde{S}$ before and after taking a single optimization step on it. The learning rate, and its schedule, have a direct effect on this single optimization step. For certain learning rate schedules, the learning rate might become close to zero (e.g., at the end of training), and hence the magnitude of this single step would become too small to be informative. To avoid this, and to account for the entire history of the optimization trajectory[4], we compute the average difference over the learning trajectory (a moving average). We propose the following metric, which we call *susceptibility to noisy labels*, $\zeta(t)$. At step t of training,

$$\zeta(t) = \frac{1}{t}\sum_{\tau=1}^{t}\left[\widetilde{\Phi}\left(\Theta(\tau)\right) - \widetilde{\Phi}\left(\widetilde{\Theta}\left(\tau+1\right)\right)\right], \quad (4.2)$$

Algorithm 1: Computes the susceptibility to noisy labels ζ

Input: Dataset S, Number of Epochs T
Sample a mini-batch $\widetilde{S}$ from S; Replace its labels with random labels
Initialize network $f_{\Theta(0)}$
Initialize $\zeta(0) = 0$
for $t = 1, \cdots, T$ **do**
  Update $f_{\Theta(t)}$ from $f_{\Theta(t-1)}$ using dataset S
  Compute $\widetilde{\Phi}(\Theta(t))$
  Update $f_{\widetilde{\Theta}(t+1)}$ from $f_{\Theta(t)}$ using dataset $\widetilde{S}$
  Compute $\widetilde{\Phi}(\widetilde{\Theta}(t+1))$
  Compute $\zeta(t) = \frac{1}{t}\Big[(t-1)\zeta(t-1) + \widetilde{\Phi}(\Theta(t)) - \widetilde{\Phi}(\widetilde{\Theta}(t+1))\Big]$
end for
Return ζ.

where $\Theta(\tau)$, $\widetilde{\Theta}(\tau+1)$ are the model parameters before and after a single update step on $\widetilde{S}$. Algorithm 1 describes the computation of ζ. The lower $\zeta(t)$ is, the less the model changes its predictions for $\widetilde{S}$ after a single step of training on it, and thus the less it memorizes the randomly-labeled samples. We say therefore that a model is resistant (to the memorization of randomly-labeled samples) when the susceptibility to noisy labels $\zeta(t)$ is low.

Susceptibility ζ tracks memorization for different models The classification accuracy of a model on a set is defined as the ratio of the number of samples where the label predicted

[4] The importance of the entire learning trajectory including its early phase is emphasized in prior work such as [Jastrzebski et al. 2020].

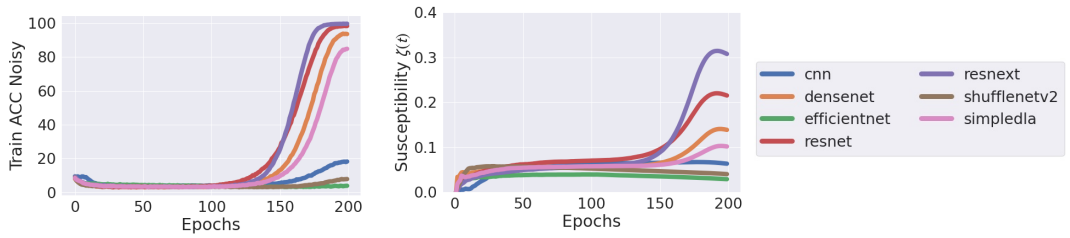
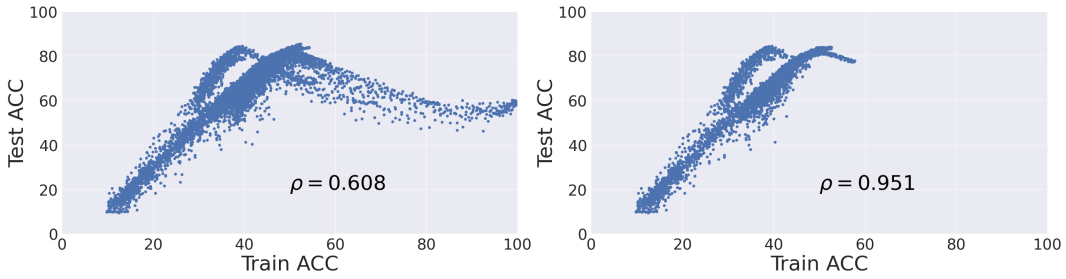


Figure 4.3: Accuracy on the noisy subset of the training set versus susceptibility $\zeta(t)$ (Eq. (4.2)) for deep convolutional neural networks (ranging from a 5−layer cnn to more sophisticated structures such as EfficientNet [Tan and Le 2019]) trained on CIFAR-10 with 50% label noise. We observe a strong Pearson correlation coefficient between Train ACC Noisy and $\zeta(t)$: $\rho = 0.884$.

by the model matches the label on the dataset, to the total number of samples. We refer to the subset for which the dataset label is different from the ground-truth label as the noisy subset of the training set. Its identification requires access to the ground-truth label. Recall that we refer to memorization within the training set as the fit on the noisy subset, which is measured by the accuracy on it. We call this accuracy "Train ACC Noisy" in short. In Fig. 4.3, we observe a strong positive correlation between the memorization of the held-out randomly labeled set, which is tracked by $\zeta(t)$ (Eq. (4.2)) and the memorization of noisy labels within the training set, tracked by Train ACC Noisy. Susceptibility ζ, which is computed on a mini-batch with labels independent from the training set, can therefore predict the resistance to memorization of the noisy subset of the training set. In Fig. 4.30, we show the robustness of susceptibility ζ to the mini-batch size, and to the choice of the particular mini-batch.

Susceptibility ζ tracks memorization for a single model It is customary when investigating different capabilities of a model, to keep checkpoints/snapshots during training and decide which one to use based on the desired property, see for example [Zhang et al. 2019a; Chatterji et al. 2019; Neyshabur et al. 2020; Andreassen et al. 2021; Baldock et al. 2021]. With access to ground-truth labels, one can track the fit on the noisy subset of the training set to find the model checkpoint with the least memorization, which is not necessarily the end checkpoint, and hence an early-stopped version of the model. However, the ground-truth label is not accessible in practice, and therefore the signals presented in Fig. 4.3 (left) are absent. Moreover, as discussed in Fig. 4.6, the training accuracy on the entire training set is also not able to recover these signals. Using susceptibility ζ, we observe that these signals can be recovered without any ground-truth label access, as shown in Fig. 4.3 (right). Therefore, ζ can be used to find the model checkpoint with the least memorization. For example, in Fig. 4.3, for the ResNet model, susceptibility ζ suggests to select model checkpoints before it sharply increases, which exactly matches with the sharp increase in the fit on the noisy subset. On the other hand, for the EfficientNet model, susceptibility suggests to select the end checkpoint, which is also consistent with the selection according to the fit on the noisy subset.



(a) Test accuracy versus train accuracy for all models at all epochs

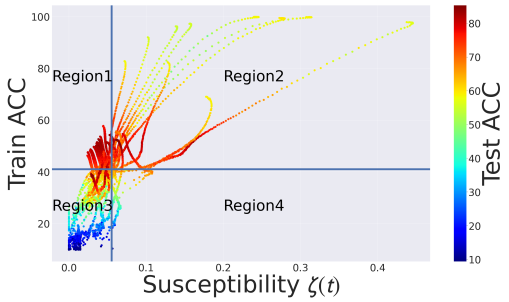
(b) Test accuracy versus train accuracy for models that have $\zeta(t) \leq 0.05$

Figure 4.4: The effect of filtering the models based on the susceptibility metric $\zeta(t)$ for models trained on CIFAR-10 with 50% label noise. We observe that by removing models with a high value of $\zeta(t)$, the correlation between the training accuracy (accessible to us in practice) and the test accuracy (not accessible) increases a lot, and we observe that among the selected models, a higher training accuracy implies a higher test accuracy. Refer to Fig. 4.13, Fig. 4.15, Fig. 4.18, Fig. 4.21, Fig. 4.23 for similar results on 5 other datasets. We use a threshold such that half of the models have $\zeta <$ threshold and hence remain after filtration.

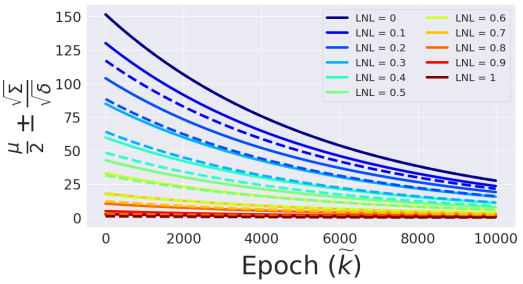
4.4 Good Models are Resistant and Trainable

When networks are trained on a dataset that contains clean and noisy labels, the fit on the clean and noisy subsets affect the generalization performance of the model differently. Therefore, as we observe in Fig. 4.4a for models trained on a noisy dataset (details are deferred to Section 4.B), the correlation between the training accuracy (accuracy on the entire training set) and the test accuracy (accuracy on the unseen clean test set) is low. Interestingly however, we observe in Fig. 4.4b that if we remove models with a high memorization on the noisy subset, as indicated by a large susceptibility to noisy labels $\zeta(t)$, then the correlation between the training and the test accuracies increases significantly (from $\rho = 0.608$ to $\rho = 0.951$), and the remaining models with low values of $\zeta(t)$ have a good generalization performance: a higher training accuracy implies a higher test accuracy. This is especially important in the practical settings where we do not know how noisy the training set is, and yet must reach a high accuracy on clean test data.

After removing models with large values of $\zeta(t)$, we select models with a low label noise memorization. However, consider an extreme case: a model that always outputs a constant is also low at label noise memorization, but does not learn the useful information present in the clean subset of the training set either. Clearly, the models should be “trainable” on top of being resistant to memorization. A “trainable model" is a model that has left the very initial stages of training, and has enough capacity to learn the information present in the clean subset of the dataset to reach a high accuracy on it. The training accuracy on the entire set is a weighted average of the training accuracy on the clean and noisy subsets. Therefore, among the models with a low accuracy on the noisy subset (selected using susceptibility ζ in Fig. 4.4b), we should further restrict the selection to those with a high overall training accuracy, which corresponds to a high accuracy on the clean subset.



(a) For models trained on CIFAR-10 with 50% label noise, we use two thresholds on susceptibility and accuracy to divide these models into 4 regions and corresponding categories as follows. We use the average values of $\zeta(t)$ and of the training accuracy over the available models as the thresholds. **Region 1**: Trainable and resistant, with average test accuracy of **76**%. **Region 2**: Trainable and but not resistant, with average test accuracy of 70.59%. **Region 3**: Not trainable but resistant, with average test accuracy of 52.49%. **Region 4**: Neither trainable nor resistant, with average test accuracy of 58.23%. We observe that the models in Region 1 generalize well on unseen data, as they have a high test accuracy.



(b) The lower (dashed lines) and upper (solid lines) bound terms of Theorem 4 that depend on the label noise level (LNL). They are computed from the eigenvectors and eigenvalues of the Gram-matrix for 1000 samples of the MNIST dataset, computed over 10 random draws computed with hyper-parameters $\gamma = 10^{-6}$, $k = 10000$ and $\delta = 0.05$. We observe that both the lower and the upper bounds in Theorem 4 are a decreasing function of LNL and of the number of epochs $\tilde{k}$. Therefore, we conclude that $\tilde{\Phi}(k+\tilde{k})$ is also a decreasing function of LNL and of $\tilde{k}$ (Proposition 1).

We can use the two metrics, susceptibility to noisy labels $\zeta(t)$ and overall training accuracy, to partition models that are being trained on some noisy dataset in different regions. For simplicity, we use the average values of ζ and training accuracy as thresholds that are used to obtain four different regions. In Section 4.F and Fig. 4.33, we observe that our model selection approach is robust to this threshold choice. According to this partitioning and as shown in Fig. 4.5a the models in each region are: Region 1: Trainable and resistant to memorization (i.e., having a high training accuracy and a low ζ), Region 2: Trainable but not resistant, Region 3: Not trainable but resistant, and Region 4: Neither trainable nor resistant. Note that the colors of each point, which indicate the value of the test accuracy are only for illustration and are not used to find these regions. The average test accuracy is the highest for models in Region 1 (i.e., that are resistant to memorization on top of being trainable). In particular, going from Region 2 to Region 1 increases the average test accuracy of the selected models from 70.59% to 76%, i.e., a 7.66% relative improvement in the test accuracy. This improvement is consistent with other datasets as well: 6.7% for Clothing-1M (Fig. 4.5g), 4.2% for Animal-10N (Fig. 4.5h), 2.06% for CIFAR-10N (Fig. 4.5i), 31.4% for noisy MNIST (Fig. 4.12), 33% for noisy Fashion-MNIST (Fig. 4.14), 33.6% for noisy SVHN (Fig. 4.16), 15.1% for noisy CIFAR-100 (Fig. 4.19), and 8% for noisy Tiny ImageNet (Fig. 4.22) datasets (refer to Section 4.E for detailed results).

Comparison with using a noisy validation set: Given a dataset containing noisy labels, a correctly-labeled validation set is not accessible without ground-truth label access. One can however use a subset of the available dataset and assess models using this noisy validation set. An important advantage of our approach compared to this assessment is the information that

it provides about the memorization within the training set, which is absent from performance on a test set or a noisy validation set. For example, in Fig. 4.28a and Fig. 4.29a we observe models (on the very top right) that have memorized 100% of the noisy subset of the training set, and yet have a high test accuracy. However, some models (on the top middle/left part of the figure), which have the same fit on the clean subset but lower memorization, have lower or similar test accuracy. For more details, refer to Section 4.C.

Moreover, as formally discussed in [Lam and Stork 2003], the number of noisy validation samples that are equivalent to a single clean validation sample depends on the label noise level of the dataset and on the true error rate of the model, which are both unknown in practice, making it difficult to pick an acceptable size for the validation set without compromising the training set. If we use a small validation set (which allows for a large training set size), then we observe a very low correlation between the validation and the test accuracies (as reported in Table 4.1 and Fig. 4.9a in Section 4.C). In contrast, our approach provides much higher correlation to the test accuracy. We also conducted experiments with noisy validation sets with larger sizes and observed that the size of the noisy validation set would need to be increased around ten-fold to reach the same correlation as our approach. Increasing the size of the validation set removes data out of the training set, which may degrade the overall performance, whereas our approach leaves the entire available dataset for training. Overall, in addition to robustness against label noise level and dataset size, our approach brings advantages in offering a gauge on memorization, without needing extra data, while maintaining a very low computational cost.

4.5 Convergence Analysis

In this section, we elaborate on the analysis that leads to (the informal) Proposition 1. A matrix, a vector, and a scalar are denoted respectively by $\mathbf{A}, \mathbf{a}$, and a. The identity matrix is denoted by $\mathbf{I}$. The indicator function of a random event A is denoted by $\mathbb{I}(A)$. The $(i,j)-$th entry of Matrix $\mathbf{A}$ is $\mathbf{A}_{ij}$.

Consider the setting described in Section 4.2 with $\|\mathbf{x}\|_2 = 1$ and $\left|y\right| \leq 1$ for $(\mathbf{x}, y) \sim \mathscr{D} := \mathscr{X} \times \mathscr{Y}$, where $\mathscr{X} = \mathbb{R}^d$ and $\mathscr{Y} = \mathbb{R}$. For input samples $\{\mathbf{x}_i\}_1^n$, the $n \times n$ Gram matrix $\mathbf{H}^\infty$ has entries

$$\mathbf{H}^\infty_{ij} = \mathbb{E}_{\theta \sim \mathscr{N}(\mathbf{0},\mathbf{I})}\left[\mathbf{x}_i^T \mathbf{x}_j \mathbb{I}\{\theta^T \mathbf{x}_i \geq 0, \theta^T \mathbf{x}_j \geq 0\}\right] = \frac{\mathbf{x}_i^T \mathbf{x}_j \left(\pi - \arccos(\mathbf{x}_i^T \mathbf{x}_j)\right)}{2\pi}, \ \forall i, j \in [n], \tag{4.3}$$

and its eigen-decomposition is $\mathbf{H}^\infty = \sum_{i=1}^n \lambda_i \mathbf{v}_i \mathbf{v}_i^T$, where the eigenvectors $\mathbf{v}_i \in \mathbb{R}^n$ are orthonormal and $\lambda_0 = \min\{\lambda_i\}_1^n$. Using the eigenvectors and eigenvalues of $\mathbf{H}^\infty$, we have the following theorem.

Theorem 3. *For* $\kappa = O\left(\frac{\epsilon\sqrt{\delta}\lambda_0}{n^{3/2}}\right)$, $m = \Omega\left(\frac{n^9}{\lambda_0^6 \epsilon^2 \kappa^2 \delta^4}\right)$, *and* $\gamma = O\left(\frac{\lambda_0}{n^2}\right)$, *with probability at least* $1-\delta$ *over the random parameter initialization described in Section 4.2 , we have:*

$$\left\|\mathbf{f}_{\Theta(k+\tilde{k})}-\widetilde{\mathbf{y}}\right\|_2=\sqrt{\sum_{i=1}^{n}\left[\mathbf{v}_i^T\mathbf{y}-\mathbf{v}_i^T\widetilde{\mathbf{y}}-\left(1-\eta\lambda_i\right)^k\mathbf{v}_i^T\mathbf{y}\right]^2\left(1-\gamma\lambda_i\right)^{2\tilde{k}}}\pm\epsilon.$$

The proof is provided in Section 4.J.

Theorem 3 enables us to approximate the objective function on dataset $\widetilde{S}$ (Section 4.2) as follows:

$$\widetilde{\Phi}(k+\tilde{k})\approx\frac{1}{2}\sum_{i=1}^{n}\left[p_i-\widetilde{p}_i-\left(1-\gamma\lambda_i\right)^k p_i\right]^2\left(1-\gamma\lambda_i\right)^{2\tilde{k}}, \tag{4.4}$$

where $p_i=\mathbf{v}_i^T\mathbf{y}$ and $\widetilde{p}_i=\mathbf{v}_i^T\widetilde{\mathbf{y}}$. Let us define

$$\mu=\sum_{i=1}^{n}\mathbb{E}\left[p_i^2\right]\left[1-\left(1-\gamma\lambda_i\right)^k\right]^2\left(1-\gamma\lambda_i\right)^{2\tilde{k}},\quad \Sigma=\mathrm{Var}_{\widetilde{\mathbf{p}},\mathbf{p}}\left[\widetilde{\Phi}(k+\tilde{k})\right]. \tag{4.5}$$

We numerically observe that $\mu/2\pm\sqrt{\Sigma/\delta}$ are both decreasing functions of the label noise level (LNL) of the label vector $\mathbf{y}$, and of $\widetilde{k}$ (see Fig. 4.5b, and Fig. 4.34, Fig. 4.35, Fig. 4.36, Fig. 4.37 in Section 4.K for different values of the learning rate γ, number of steps k, and datasets). The approximation in Eq. (4.4) together with the above observation then yields the following lower and upper bounds for $\widetilde{\Phi}(k+\tilde{k})$.

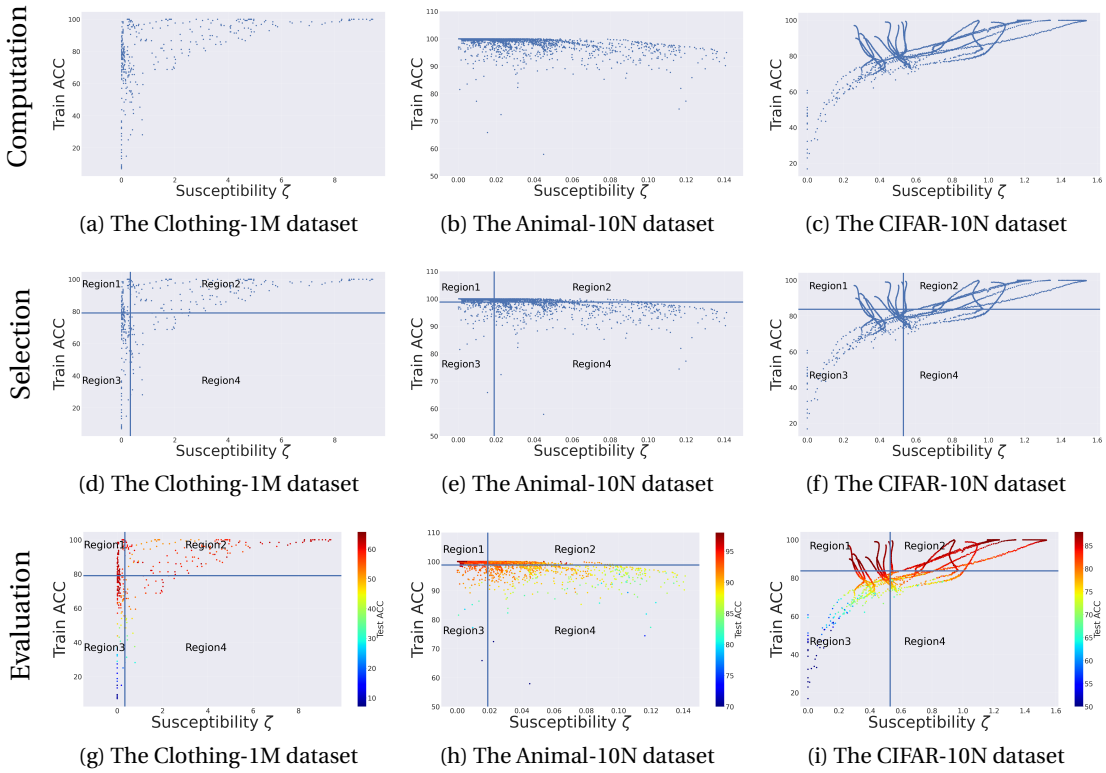
Theorem 4. *With probability at least* $1-\delta$ *over the draw of the random label vector* $\widetilde{\mathbf{y}}$*, given the approximation made in Eq.* (4.4)*,*

$$\frac{\mu}{2}-\sqrt{\frac{\Sigma}{\delta}}\leq\widetilde{\Phi}(k+\tilde{k})-\frac{1}{2}\sum_{i=1}^{n}\left(1-\gamma\lambda_i\right)^{2\tilde{k}}\leq\frac{\mu}{2}+\sqrt{\frac{\Sigma}{\delta}} \tag{4.6}$$

The proof is provided in Section 4.K. Because the term $\sum_{i=1}^{n}\left(1-\gamma\lambda_i\right)^{2\tilde{k}}$ is independent of LNL of the label vector $\mathbf{y}$, we can conclude that $\widetilde{\Phi}(k+\tilde{k})$ is a decreasing function of LNL. Moreover, Since $0<1-\gamma\lambda_i<1$, the term $\sum_{i=1}^{n}\left(1-\gamma\lambda_i\right)^{2\tilde{k}}$ is also a decreasing function of $\widetilde{k}$. Therefore, similarly, we can conclude that $\widetilde{\Phi}(k+\tilde{k})$ is a decreasing function of $\widetilde{k}$. Proposition 1 summarizes this result.

4.6 Experiments on Real-world Datasets with noisy labels

To evaluate the performance of our approach on real-world datasets, we have conducted additional experiments on the Clothing-1M dataset [Xiao et al. 2015], which is a dataset with 1M images of clothes, on the Animal-10N dataset [Song et al. 2019], which is a dataset with 50k images of animals and on the CIFAR-10N dataset [Wei et al. 2022], which is the CIFAR-10 dataset with human-annotated noisy labels obtained from Amazon Mechanical Turk. In the Clothing-1M dataset, the images have been labeled from the texts that accompany them, hence there are both clean and noisy labels in the set, and in the Animal-10N dataset, the



(a) The Clothing-1M dataset (b) The Animal-10N dataset (c) The CIFAR-10N dataset

(d) The Clothing-1M dataset (e) The Animal-10N dataset (f) The CIFAR-10N dataset

(g) The Clothing-1M dataset (h) The Animal-10N dataset (i) The CIFAR-10N dataset

Figure 4.5: For real-world noisy labeled datasets, we show in three steps the efficiency of our model-selection approach. **Top:** Training accuracy and susceptibility ζ are computed for models trained on each dataset. **Middle:** According to the average values of the training accuracy and susceptibility ζ the figure is divided into 4 regions. Our model-selection approach suggests selecting models in Region 1, which are resistant to memorization on top of being trainable. **Bottom:** The test accuracy of the models is visualized using the color of each point (which is only for illustration and is not used to find different regions). We can observe that models in Region 1 have the highest test accuracies in each dataset.

images have been gathered and labeled from search engines. In these datasets, some images have incorrect labels and the ground-truth labels in the training set are not available. Hence in our experiments, we cannot explicitly track memorization as measured by the accuracy on the noisy subset of the training set.

We train different settings on these two datasets with various architectures (including ResNet, AlexNet and VGG) and varying hyper-parameters (refer to Section 4.B for details). We compute the training accuracy and susceptibility ζ during the training process for each setting and visualize the results in Fig. 4.5 (a)-(c).

We divide the models of Fig. 4.5 (a)-(c) into 4 regions, where the boundaries are set to the average value of the training accuracy (horizontal line) and the average value of susceptibility (vertical line): Region 1: Models that are trainable and resistant to memorization, Region 2: Trainable and but not resistant, Region 3: Not trainable but resistant and Region 4: Neither

trainable nor resistant. This is shown in Fig. 4.5 (d)-(f).

Our approach suggests selecting models in Region 1 (low susceptibility, high training accuracy). In order to assess how our approach does in model selection, we can reveal the test accuracy computed on a held-out clean test set in Fig. 4.5 (g)-(i). We observe that the average (± standard deviation) of the test accuracy of models in each region is as follows:

- *Clothing-1M dataset:* Region 1: **61.799**% ± 1.643; Region 2: 57.893% ± 3.562; Region 3: 51.250% ± 17.209; Region 4: 51.415% ± 9.709.
- *Animal-10N dataset:* Region 1: **96.371**% ± 1.649; Region 2: 92.508% ± 2.185; Region 3: 91.179% ± 6.601; Region 4: 89.352% ± 3.142.
- *CIFAR-10N dataset:* Region 1: **87.2**% ± 0.99; Region 2: 85.44% ± 2.52; Region 3: 77.87% ± 8.15; Region 4: 78.45% ± 3.86.

We observe that using our approach we are able to select models with very high test accuracy. In addition, the test accuracies of models in Region 1 have the least amount of standard deviation. Note that our susceptibility metric ζ does not use any information about the label noise level or the label noise type that is present in these datasets. Similarly to the rest of this chapter, random labeling is used for computing ζ. Interestingly, even though within the training sets of these datasets the label noise type is different than random labeling (label noise type is instance-dependent [Xia et al. 2020; Wei et al. 2022]), ζ is still successfully tracking memorization.

Therefore, our approach selects trainable models with low memorization even for datasets with real-world label noise. Observe that selecting models only on the basis of their training accuracy or only on the basis of their susceptibility fails: both are needed. It is interesting to note that in the Clothing-1M dataset, as the dataset is more complex, the range of the performance of different models varies and our approach is able to select "good" models from "bad" models. On the other hand, in the Animal-10N dataset, as the dataset is easier to learn and the estimated label noise level is lower, most models are already performing rather well. Here, our approach is able to select the "best" models from "good" models.

4.7 On the Generality of the Observed Phenomena

In this section, we provide additional experiments that show our empirical results hold across different choices of dataset, training, and architecture design as well as label noise level and form.

Dataset: In addition to the real-world datasets provided in the previous section, our results consistently hold for datasets MNIST, Fashion MNIST, SVHN, CIFAR-100, and Tiny ImageNet datasets; see Fig. 4.12-Fig. 4.23 in Section 4.E.

Learning rate schedule: Our results are not limited to a specific optimization scheme. In our experiments, we apply different learning rate schedules, momentum values, and optimizers (SGD and Adam) (for details see Section 4.B). More specifically, we show in Fig. 4.25 (in Section 4.F) that the strong correlation between memorization and our metric $\zeta(t)$ stays consistent for both learning rate schedulers `cosineannealing` and `exponential`.

Architecture: Results of Section 4.3 and Section 4.4 are obtained from a variety of architecture families, such as DenseNet [Huang et al. 2017], MobileNet [Howard et al. 2017], VGG [Simonyan and Zisserman 2014], and ResNet [He et al. 2016a]. For the complete list of architectures, see Section 4.B. We observe that $\zeta(t)$ does not only detect resistant architecture families (as done for example in Fig. 4.3), but that it is also able to find the best design choice (e.g., width) among configurations that are already resistant, see Fig. 4.24 in Section 4.F.

Low label noise levels: In addition to the real-world datasets with low label noise levels (label noise level of Animal-10N and CIFAR-10N are 8% and 9%, respectively), we studied low label noise levels in datasets with synthetic label noises as well. For models trained on CIFAR-10 and CIFAR-100 datasets with 10% label noise (instead of 50%), we still observe a high correlation between accuracy on the noisy subset and $\zeta(t)$ in Fig. 4.26 and Fig. 4.27 in Section 4.F. Moreover, we observe in Fig. 4.28 and Fig. 4.29 in Section 4.F that the average test accuracy of the selected models using our metric is comparable with the average test accuracy of the selected models with access to the ground-truth label.

Asymmetric label noise: In addition to the real-world datasets with asymmetric label noises (label noise type of the datasets in Section 4.6 is instance-dependent [Xia et al. 2020; Wei et al. 2022]), we studied synthetic asymmetric label noise as well. We have performed experiments with asymmetric label noise as proposed in [Xia et al. 2021]. Using our approach, the average test accuracy of the selected models is 66.307%, whereas the result from oracle is 66.793% (see Fig. 4.32 in Section 4.E).

4.8 Conclusion

We have proposed a simple but surprisingly effective approach to track memorization of the noisy subset of the training set using a single mini-batch of unlabeled data. Our contributions are three-fold. First, we have shown that models with a high test accuracy are resistant to memorization of a held-out randomly-labeled set. Second, we have proposed a metric, susceptibility, to efficiently measure this resistance. Third, we have empirically shown that one can utilize susceptibility and the overall training accuracy to distinguish models (whether they are a single model at various checkpoints, or different models) that maintain a low memorization on the training set and generalize well to unseen clean data while bypassing the need to access the ground-truth label. We have studied model selection in a variety of experimental settings and datasets with label noise, ranging from selecting the "best" models from "good" models (for easy datasets such as Animal-10N) to selecting "good" models from "bad" models (for more complex datasets such as Clothing-1M).

Our theoretical results have direct implications for online settings. When the quality of the labels for new data stream is low, Theorem 3 can directly compare how different models converge on this low-quality data, and help in selecting resistant models, which converge slowly on this data. Our empirical results provide new insights on the way models memorize the noisy subset of the training set: the process is similar to the memorization of a purely randomly-labeled held-out set.

Appendices

4.A Additional Related Work

Memorization Arpit et al. [2017] describe memorization as the behavior shown by neural networks when trained on noisy data. Patel and Sastry [2021] define memorization as the difference in the predictive performance of models trained on clean data and on noisy data distributions, and propose a robust objective function that resists to memorization. The effect of neural network architecture on their robustness to noisy labels is studied in [Li et al. 2021], which measures robustness to noisy labels by the prediction performance of the learned representations on the ground-truth target function. Feldman and Zhang [2020] formally define memorization of a sample for an algorithm as the inability of the model to predict the output of a sample based on the rest of the dataset, so that the only way that the model can fit the sample is by memorizing its label. Our definition of memorization as the fit on the noisy subset of the training set follows the same principle; the fit of a randomly-labeled individual sample is only possible if the totally random label associated with the sample is memorized, and it is impossible to predict using the rest of the dataset.

Learning before Memorization Multiple studies have reported that neural networks learn simple patterns first, and memorize the noisy labeled data later in the training process [Arpit et al. 2017; Gu and Tresp 2019; Krueger et al. 2017; Liu et al. 2020b]. Hence, early stopping might be useful when learning with noisy labels. Parallel to our study, Lu and He [2022] propose early stopping metrics that are computed on the training set. These early stopping metrics are computed by an iterative algorithm (either a GMM or k-Means) on top of the training loss histograms. This adds computational overhead compared to our approach, which is based on a simple metric that can be computed on the fly. Moreover, we can observe in Fig. 4.8 (middle) that the metric proposed in Lu and He [2022], which is the mean difference between distributions obtained from the GMM applied on top of the training losses, does not correlate with memorization on the training set, as opposed to our metric susceptibility. Hence, we can conclude that, even though the mean difference metric might be a rather well early stopping criterion for a single setting, it does not perform well in terms of comparing different settings to one another. Moreover, Rahaman et al. [2019]; Xu et al. [2019a;b] show that neural networks learn lower frequencies in the input space first and higher frequencies later. However, the monotonic behavior of neural networks during the training procedure is recently challenged

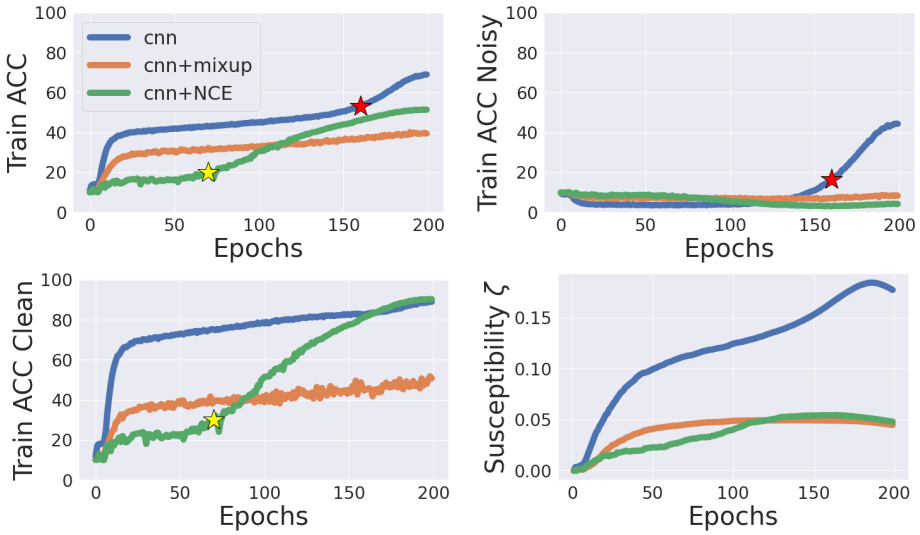


Figure 4.6: Accuracy on the entire, the clean and the noisy subsets of the training set, versus susceptibility in Eq. (4.2) for a 5−layer convolutional neural network trained on CIFAR-10 with 50% label noise without any regularization, with Mixup [Zhang et al. 2017], and with Active Passive loss using the normalized cross entropy together with reverse cross-entropy loss (denoted by NCE) [Ma et al. 2020]. The sharp increases in the overall training accuracy (illustrated by the red and yellow stars) can be caused by an increase in the accuracy of either the clean (the yellow star) or noisy (the red star) subsets. Therefore, this sharp increase could not predict memorization, contrary to the susceptibility ζ, which is strongly correlated with the accuracy on the noisy subset (Pearson correlation between Train ACC Noisy and susceptibility ζ is $\rho = 0.636$).

by the epoch-wise double descent [Zhang et al. 2020a; Nakkiran et al. 2021]. When a certain amount of fit is observed on the entire training set, how much of this fit corresponds to the fit on the clean and noisy subsets, respectively, is still unclear. In this chapter, we propose an approach to track the fit to the clean subset and the fit to the noisy subset (memorization) explicitly.

Training Speed Lyle et al. [2020] show there is a connection between training speed and the marginal likelihood for linear models. Jiang and Gal [2021] also find an explanation for the connection between training speed and generalization, and Ru et al. [2021] use this connection for neural architecture search. However, in Fig. 4.6, we observe that a sharp increase in the training accuracy (a high training speed), does not always indicate an increase in the value of accuracy on the noisy subset. This result suggests that studies that relate training speed with generalization [Lyle et al. 2020; Ru et al. 2021] might not be extended as such to noisy-label training settings. On the other hand, we observe a strong correlation between susceptibility ζ and accuracy on the noisy subset.

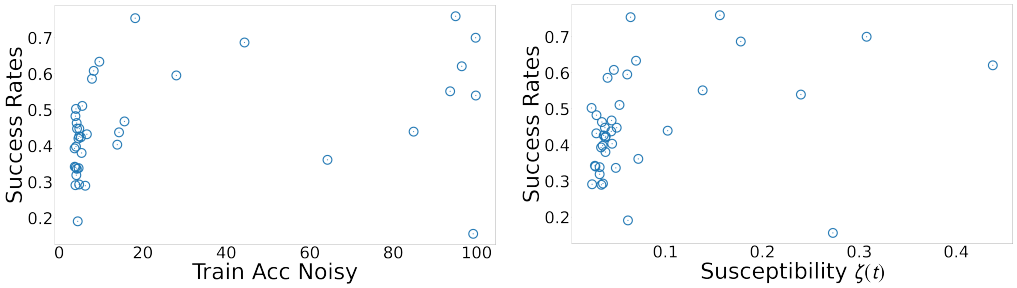
Leveraging Unlabeled/Randomly-labeled Data Unlabeled data has been leveraged previously to predict out-of-distribution performance (when there is a mismatch between the

training and test distributions) [Garg et al. 2021b]. Li et al. [2019a] introduce synthetic noise to unlabeled data, to propose a noise-tolerant meta-learning training approach. Zhang et al. [2021b] use randomly labeled data to perform neural architecture search. In this chapter, we leverage unlabeled data to track the memorization of label noise.

Benefits of Memorization Studies on long-tail data distributions show potential benefits from memorization when rare and atypical instances are abundant in the distribution [Feldman 2020; Feldman and Zhang 2020; Brown et al. 2021]. These studies argue that the memorization of these rare instances is required to guarantee low generalization error. On a separate line of work, previous empirical observations suggest that networks trained on noisy labels can still induce good representations from the data, despite their poor generalization performance [Li et al. 2020b; Maennel et al. 2020]. In this work, we propose an approach to track memorization when noisy labels are present.

Theoretical Results To analyze the convergence of different models on some randomly labeled set, we rely on recent results obtained by modeling the training process of wide neural networks by neural tangent kernels (NTK) [Du et al. 2018; Jacot et al. 2018; Allen-Zhu et al. 2018; Arora et al. 2019; Bietti and Mairal 2019; Lee et al. 2019]. In particular, the analysis in this chapter is motivated by recent work on convergence analysis of neural networks [Du et al. 2018; Arora et al. 2019]. Arora et al. [2019] perform fine-grained convergence analysis for wide two-layered neural networks using gradient descent. They study the convergence of models trained on different datasets (datasets with clean labels versus random labels). In Section 4.5, we study the convergence of different models trained on some randomly-labeled set. In particular, our models are obtained by training wide two-layered neural networks on datasets with varying label noise levels. As we highlight in the proofs, our work builds on previous art, especially [Du et al. 2018; Arora et al. 2019], yet we build in a non-trivial way and for a new problem statement.

Robustness to Adversarial Examples Deep neural networks are vulnerable to adversarial samples [Szegedy et al. 2013; Goodfellow et al. 2014; Akhtar and Mian 2018]: very small changes to the input image can fool even state-of-the-art neural networks. This is an important issue that needs to be addressed for security reasons. To this end, multiple studies towards generating adversarial examples and defending against them have emerged [Carlini and Wagner 2017; Papernot et al. 2016b;a; Athalye et al. 2018]. As a side topic related to the central theme of the chapter, we examined the connection between models that are resistant to memorization of noisy labels and models that are robust to adversarial attacks. Fig. 4.7 compares the memorization of these models trained on some noisy dataset, with their robustness to adversarial attacks. We do not observe any positive or negative correlation between the two. Some models are robust to adversarial attacks, but perform poorly when trained on datasets with noisy labels, and vice versa. This means that the observations made in this chapter are orthogonal to the ongoing discussion regarding the trade-off between robustness to adversarial samples and test accuracy [Tsipras et al. 2018; Stutz et al. 2019; Yang et al. 2020; Pedraza et al. 2021; Zhang et al. 2019b; Yin et al. 2019].



(a) Success rate versus accuracy on the noisy subset of the training set

(b) Success rate versus susceptibility to noisy labels ζ

Figure 4.7: SimBa [Guo et al. 2019] adversarial attack success rate versus accuracy on the noisy subset of the training set for networks trained on CIFAR-10 with 50% label noise. To evaluate the robustness of different models with respect to adversarial attacks, we compare the success rate of the adversarial attack SimBa proposed in [Guo et al. 2019] after 1000 iterations. We observe no correlation between the success rate and neither accuracy on the noisy subset nor susceptibility ζ.

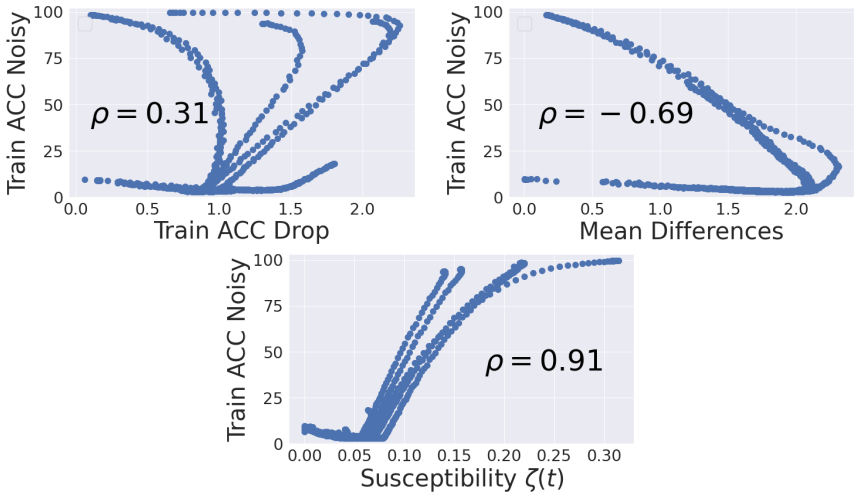


Figure 4.8: **Left:** Accuracy on the noisy subset of the training set, versus the training accuracy drop presented in [Zhang et al. 2019c] for the following neural network configurations trained on the CIFAR-10 dataset with 50% label noise: a 5-layer convolutional neural network, DenseNet, EfficientNet, MobileNet, MobileNetV2, RegNet, ResNet, ResNeXt, SENet, and ShuffleNetV2. As proposed in [Zhang et al. 2019c], to compute the training accuracy drop, we create a new dataset from the available noisy training set, by replacing 25% of its labels with random labels. We then train these networks on the new dataset, and then compute the difference in the training accuracy of the two setups, divided by 25 (the level of label noise that was injected). This is reported above as Train ACC Drop. Zhang et al. [2019c] state that desirable networks should have a large drop; therefore there should ideally be a *negative* correlation between the training accuracy drop and the accuracy on the noisy subset. However, we observe a slightly positive correlation. Therefore, it is difficult to use the results of Zhang et al. [2019c] to track memorization, i.e., training accuracy on the noisy subset. **Middle:** Accuracy on the noisy subset of the training set, versus the mean difference between the distributions obtained by applying GMM on the training losses, which is proposed in Lu and He [2022] as an early stopping criterion. We observe a negative correlation between the two, and hence conclude that this metric cannot be used to compare memorization of different settings at different stages of training. **Right:** As opposed to the other two metrics, we observe a strong positive correlation between our metric susceptibility ζ and memorization.

4.B Experimental Setup

Generating Noisy-labeled Datasets We modify original datasets similarly to Chatterjee [2020]; for a fraction of samples denoted by the label noise level (LNL), we replace the labels with independent random variables drawn uniformly from $\{1, \cdots, c\}$ for a dataset with c number of classes. On average $(1 - \text{LNL}) + (\text{LNL}) \cdot 1/c$ of the samples still have their correct labels. Note that LNL for $\widetilde{S}$ is 1.

Experiments of Fig. 4.2, Fig. 4.9, Fig. 4.10 and Fig. 4.11 The models at epoch 0 are pre-trained models on either the clean or noisy versions of the CIFAR-10 dataset for 200 epochs using SGD with learning rate 0.1, momentum 0.9, weight decay $5 \cdot 10^{-4}$, and `cosineannealing` learning rate schedule with `T max` = 200. The new sample is a sample drawn from the training set, and its label is randomly assigned to some class different from the correct class label. In Fig. 4.10, the new sample is an unseen sample drawn from the test set.

CIFAR-10[5] **Experiments** The models are trained for 200 epochs on the cross-entropy objective function using SGD with weight decay $5 \cdot 10^{-4}$ and batch size 128. The neural network architecture options are: cnn (a simple 5-layer convolutional neural network), DenseNet [Huang et al. 2017], EfficientNet [Tan and Le 2019] (with `scale=0.5, 0.75, 1, 1.25, 1.5`), GoogLeNet [Szegedy et al. 2015], MobileNet [Howard et al. 2017] (with `scale=0.5, 0.75, 1, 1.25, 1.5`), ResNet [He et al. 2016a], MobileNetV2 [Sandler et al. 2018] (with `scale=0.5, 0.75, 1, 1.25, 1.5`), Preact ResNet [He et al. 2016b], RegNet [Radosavovic et al. 2020], ResNeXt [Xie et al. 2017], SENet [Hu et al. 2018], ShuffleNetV2 [Ma et al. 2018] (with `scale=0.5, 1, 1.5, 2`), DLA [Yu et al. 2018], and VGG [Simonyan and Zisserman 2014]. The learning rate value options are: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5. The learning rate schedule options are: `cosineannealing` with `T max` 200, `cosineannealing` with `T max` 100, `cosineannealing` with `T max` 50, and no learning rate schedule. Momentum value options are 0.9, 0. In addition, we include experiments with Mixup [Zhang et al. 2017], active passive losses: normalized cross entropy with reverse cross entropy (NCE+RCE) [Ma et al. 2020], active passive losses: normalized focal loss with reverse cross entropy (NFL+RCE) [Ma et al. 2020], and robust early learning [Xia et al. 2021] regularizers.

CIFAR-100 Experiments The models are trained for 200 epochs on the cross-entropy objective function using SGD with learning rate 0.1, weight decay $5 \cdot 10^{-4}$, momentum 0.9, learning rate schedule `cosineannealing` with `T max` 200 and batch size 128. The neural network architecture options are: cnn, DenseNet, EfficientNet (with `scale=0.5, 0.75, 1, 1.25, 1.5`), GoogLeNet, MobileNet (with `scale=0.5, 0.75, 1, 1.25, 1.5`), ResNet, MobileNetV2 (with `scale=0.5, 0.75, 1, 1.25, 1.5`), RegNet, ResNeXt, ShuffleNetV2 (with `scale=0.5, 1, 1.5, 2`), DLA, and VGG. In addition, we include experiments with Mixup, active passive losses: NCE+RCE, active passive losses: NFL+RCE, and robust early learning regularizers.

SVHN [Netzer et al. 2011] Experiments The models are trained for 200 epochs on the

[5] https://www.cs.toronto.edu/~kriz/cifar.html

cross-entropy objective function with learning rate 0.1 and batch size 128. The optimizer choices are SGD with weight decay $5 \cdot 10^{-4}$, and momentum 0.9, and Adam optimizers. The learning rate schedule options for the SGD experiments are: `cosineannealing` with `T max` 200, and `exponential`. The neural network architecture options are: DenseNet, EfficientNet (with `scale=0.25, 0.5, 0.75, 1`), GoogLeNet (with `scale= 0.25, 1, 1.25`), MobileNet (with `scale= 1, 1.25, 1.5, 1.75`), MobileNetV2 (with `scale= 1, 1.25, 1.5, 1.75`), ResNet (with `scale=0.25, 0.5, 1, 1.25, 1.5, 1.75`), ResNeXt, SENet (with `scale=0.25, 0.5, 0.75, 1`), ShuffleNetV2 (with `scale=0.25, 0.5, 0.75, 1`), DLA (with `scale=0.25, 0.5, 0.75, 1`), and VGG (with `scale=0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75`).

MNIST[6] and Fashion-MNIST [Xiao et al. 2017] Experiments The models are trained for 200 epochs on the cross-entropy objective function with batch size 128, using SGD with learning rate 0.1, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The learning rate schedule options are: `cosineannealing` with `T max` 200, and `exponential`. The neural network architecture options are: AlexNet [Krizhevsky et al. 2012] (with `scale=0.25, 0.5, 0.75, 1`), ResNet18 (with `scale=0.25, 0.5, 0.75, 1`), ResNet34 (with `scale=0.25, 0.5, 0.75, 1`), ResNet50 (with `scale=0.25, 0.5, 0.75, 1`), VGG11 (with `scale=0.25, 0.5, 0.75, 1`), VGG13 (with `scale=0.25, 0.5, 0.75, 1`), VGG16 (with `scale=0.25, 0.5, 0.75, 1`), VGG19 (with `scale=0.25, 0.5, 0.75, 1`).

Tiny ImageNet [Le and Yang 2015] Experiments The models are trained for 200 epochs on the cross-entropy objective function with batch size 128, using SGD with weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The learning rate schedule options are: `cosineannealing` with `T max` 200, `exponential`, and no learning rate schedule. The learning rate options are: $0.01, 0.05, 0.1, 0.5$. The neural network architecture options are: AlexNet, DenseNet, MobileNetV2, ResNet, SqueezeNet [Iandola et al. 2016], and VGG.

Clothing-1M [Xiao et al. 2015] Experiments The models are trained for 20 epochs on the cross-entropy objective function with batch size 128 using SGD with weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The learning rate schedule options are: `cosineannealing` with `T max` 20, and `exponential`. The learning rate options are: $0.01, 0.005, 0.001$. The neural network architecture options are: AlexNet, ResNet18, ResNet 34, ResNeXt, and VGG. Note that because this dataset has random labels we do not introduce synthetic label noise to the labels.

Animal-10N [Song et al. 2019] Experiments The models are trained for 50 epochs on the cross-entropy objective function with batch size 128 using SGD with weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The learning rate schedule options are: `cosineannealing` with `T max` 50, and `exponential`. The learning rate options are: $0.001, 0.005, 0.01$. The neural network architecture options are: ResNet18, ResNet34, SqueezeNet, AlexNet and VGG. Note that because this dataset has random labels we do not introduce synthetic label noise to the labels.

[6] http://yann.lecun.com/exdb/mnist/

CIFAR-10N [Wei et al. 2022] Experiments On these experiments, we work with the aggregate version of the label set which has label noise level of 9% (CIFAR-10N-aggregate in Table 1 of Wei et al. [2022]). The models are trained for 200 epochs on the cross-entropy objective function with batch size 128 using SGD with weight decay $5 \cdot 10^{-4}$, and momentum 0.9 with `cosineannealing` learning rate schedule with `T max` 200. The learning rate options are: 0.1, 0.05. The neural network architecture options are: cnn, EfficientNet, MobileNet, ResNet, MobileNetV2, RegNet, ResNeXt, ShuffleNetV2, SENet, DLA and VGG.

Each of our experiments take few hours to run on a single Nvidia Titan X Maxwell GPU.

4.C Comparison with Baselines

Comparison to a Noisy Validation Set Following the notations from Chen et al. [2021a] (only in this paragraph we use these notations), let the clean and noisy data distributions be denoted by D and $\tilde{D}$, respectively, the classifiers by h, and the classification accuracy by A. Suppose that the optimal classifier in terms of accuracy on the clean data distribution is h^*, i.e., $h^* = \arg\max_h A_D(h)$. [Chen et al. 2021a] states that under certain assumptions on the label noise, the accuracy on the noisy data distribution is also maximized by h^*, that is $h^* = \arg\max_h A_{\tilde{D}}(h)$. However, these results do not allow us to compare any two classifiers h_1 and h_2, because we cannot conclude from the results in [Chen et al. 2021a] that if $A_D(h_1) > A_D(h_2)$, then $A_{\tilde{D}}(h_1) > A_{\tilde{D}}(h_2)$. Moreover, it is important to note that these results hold when the accuracy is computed on unlimited dataset sizes. As stated in [Lam and Stork 2003], the number of noisy validation samples that are equivalent to a single clean validation sample depends on the label noise level and on the true error rate of the model, which are both unknown in practice. Nevertheless, below we thoroughly compare our approach with having a noisy validation set.

As discussed in Section 4.4, the susceptibility ζ together with the training accuracy are able to select models with a high test accuracy. Another approach to select models is to use a subset of the available noisy dataset as a held-out validation set. Table 4.1 provides the correlation values between the test accuracy on the clean test set and the validation accuracy computed on noisy validation sets with varying sizes. On the one hand, we observe that the correlation between the validation accuracy and the test accuracy is very low for small sizes of the noisy validation set. On the other hand, in the same table, we observe that if the same size is used to compute ζ, our approach provides a high correlation to the test accuracy, even for very small sizes of the held-out set.

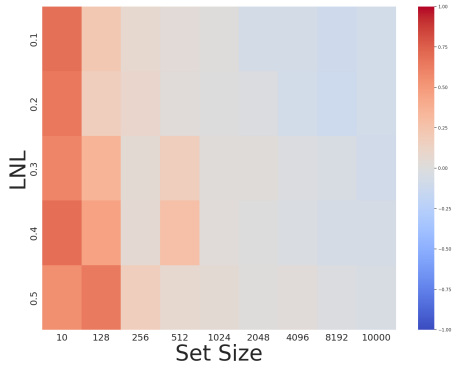
In our approach, we first filter out models for which the value of ζ exceeds a threshold. We set the threshold so that around half of the models remain after this filtration. We then report the correlation between the training and test accuracies among the remaining models. As a sanity check, we doubled checked that, with this filtration, the model with the highest test accuracy was not filtered out.

Size of the set	Train Acc	Val Acc	Our approach
10	0.513	0.244	**0.792**
128		0.458	**0.890**
256		0.707	**0.878**
512		0.799	**0.876**
1024		0.830	**0.877**
2048		0.902	**0.917**
4096		0.886	**0.912**
8192		**0.940**	0.923
10000		**0.956**	0.914

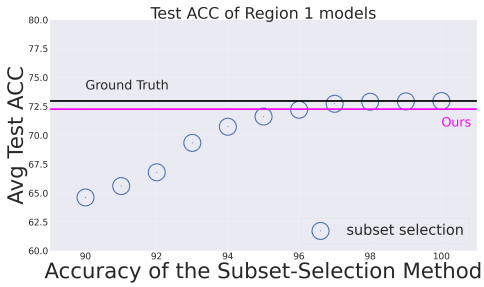
Table 4.1: The Kendall τ correlation between each metric and the test accuracy for CNN, ResNet, EfficientNet, and MobileNet trained on CIFAR-10 with 50% label noise, where the validation accuracy (Val Acc) on a held-out subset of the data and the susceptibility ζ use a set with the size indicated in each row. We observe that our approach results in a much higher correlation compared to using a noisy validation set. Furthermore, our approach is less sensitive to the size of the held-out set, compared to using a noisy validation set. In particular, to reach the same correlation value, our approach with set size = 10 is equivalent to using a noisy validation set with size = 512 (highlighted in red). Also, our approach with set size = 128 is almost equivalent to using a noisy validation set with size = 1024 (highlighted in blue). Hence, to use a noisy validation set, one requires around ten-fold the amount of held-out data.

Moreover, in Fig. 4.9a, we report the advantage of using our approach compared to using a noisy validation set for various values of the dataset label noise level (LNL) and the size of the set that computes the validation accuracy and susceptibility ζ. We observe that the lower the size of the validation set, and the higher the LNL, the more advantageous our approach is. Note also that for high set sizes and low LNLs, our approach produces comparable results to using a noisy validation set.

Comparison to Label Noise Detection Approaches Another line of work is studying methods that detect whether a label assigned to a given sample is correct or not [Zhu et al. 2021b; Song et al. 2020b; Pleiss et al. 2020a; Pulastya et al. 2021]. Such methods can estimate the clean and noisy subsets. Then, by tracking the training accuracy on the clean and noisy subsets, similar to what is done in Fig. 4.5a, they can select models that are located in Region 1, i.e., that have a low estimated accuracy on the noisy subset and a high estimated accuracy on the clean subset. In Fig. 4.9b, we compare the average test accuracy of models selected by our approach with those selected by such subset-selection methods. Let X be the accuracy of the detection of the correct/incorrect label of a sample by the subset selection benchmark: if $X = 100\%$, then the method has full access to the ground truth label for each label, if $X = 90\%$ the method correctly detects 90% of the labels. We observe a clear advantage of our method for X up to 96%, and comparable performance for $X > 96\%$.



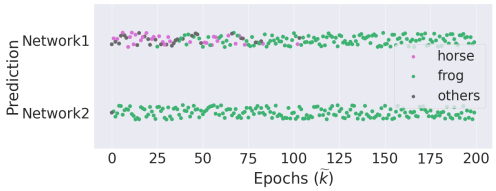
(a) The Kendall τ correlation between our approach and the test accuracy minus the Kendall τ correlation between validation accuracy (computed on a noisy set) and the test accuracy for various label noise levels (LNL) and set sizes. We observe the advantage of our approach to using a noisy validation set particularly for high LNLs and low set sizes. For other combinations, we also observe comparable results (the correlation difference is very close to zero). Note that the bottom row can be recovered from the difference in correlation values of Table 4.1.



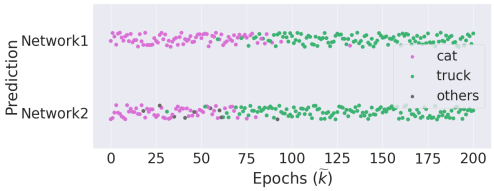
(b) Comparison between the average test accuracy obtained using our approach and the average test accuracy obtained using a method that detects correctly-labeled samples within the training set from the incorrectly-labeled ones with X% accuracy (the x-axis). The results are obtained from training {CNN, ResNet, EfficientNet, MobileNet}$\times${without regularization, +Mixup, +NCERCE, + NFLRCE, +robust early learning} on CIFAR-10 with 50% label noise. We observe that to have the same performance as our approach, such methods require a very high accuracy (above 96%), and even with higher accuracies, our approach gives comparable results.

4.D Additional Experiments for Section 4.2

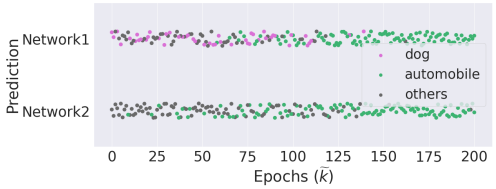
In this section, we provide additional experiments for the observation presented in Section 4.2. In Fig. 4.9, we observe that networks with a high test accuracy are resistant to memorizing a new incorrectly-labeled sample. On the other hand, in Fig. 4.10, we observe that networks with a high test accuracy tend to fit a new correctly-labeled sample faster.



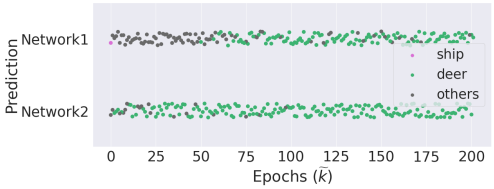
(a) Example 1: original label: horse, assigned label: frog



(b) Example 2: original label: cat, assigned label: truck



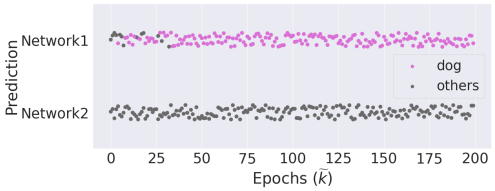
(c) Example 3: original label: dog, assigned label: automobile



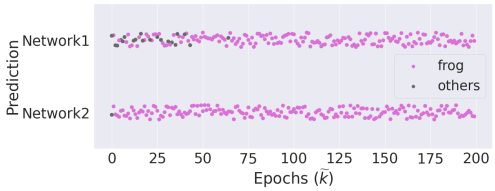
(d) Example 4: original label: ship, assigned label: deer

Figure 4.9: The evolution of output prediction of two networks that are trained on a single randomly labeled sample. In all sub-figures, Network 1 has a higher test accuracy compared to Network 2, and we observe it is more resistance to memorization of the single incorrectly-labeled sample. **Example 1:** Network 1 is a ResNeXt trained on CIFAR-10 dataset with 50% random labels and has test accuracy of 58.85%. Network 2 is a ResNeXt that is not pre-trained and has test accuracy of 9.74%. **Example 2:** Network 1 is a SENet trained on CIFAR-10 dataset with original labels and has test accuracy of 95.35%. Network 2 is a SENet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 56.38%. **Example 3:** Network 1 is a RegNet trained on CIFAR-10 dataset with original labels and has test accuracy of 95.28%. Network 2 is a RegNet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 55.36%. **Example 4:** Network 1 is a MobileNet trained on CIFAR-10 dataset with original labels and has test accuracy of 90.56%. Network 2 is a MobileNet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 82.76%.

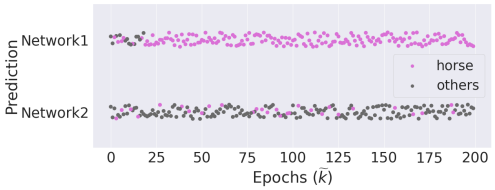
Moreover, we study the effect of calibration on the observations of Fig. 4.2 and Fig. 4.9. A poor calibration of a model may affect the confidence in its predictions, which in turn might affect the susceptibility/resistance to new samples. Therefore, in Fig. 4.11, we compare models that have almost the same calibration value. More precisely, Network 1 is trained on the clean dataset, and Network 2 (calibrated) is a calibrated version of the model that is trained on the noisy dataset using the Temperature scaling approach [Guo et al. 2017]. We observe that even with the same calibration level, the model with a higher test accuracy is more resistant to memorizing a new incorrectly-labeled sample.



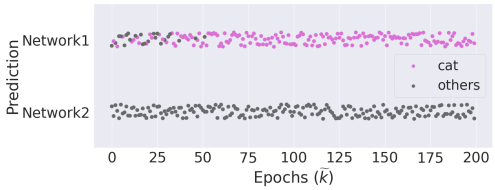
(a) Example 1: original label: dog, assigned label: dog



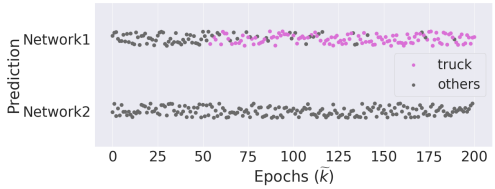
(b) Example 2: original label: frog, assigned label: frog



(c) Example 3: original label: horse, assigned label: horse

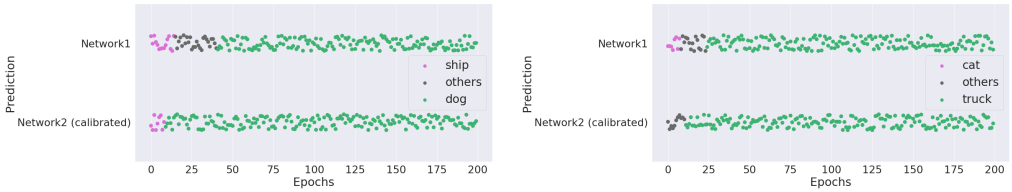


(d) Example 4: original label: cat, assigned label: cat



(e) Example 5: original label: truck, assigned label: truck

Figure 4.10: The evolution of output prediction of two networks that are trained on a single unseen correctly-labeled sample. In all sub-figures, Network 1 has a higher test accuracy compared to Network 2. We observe that given a new correctly-labeled sample Network 2 learns it later, unlike our observation in Figure 4.2 for a new incorrectly-labeled sample. **Example 1:** Network 1 is a GoogLeNet trained on CIFAR-10 dataset with clean labels and has test accuracy of 95.36%. Network 2 is a GoogLeNet trained on CIFAR-10 dataset with 50% label noise level and has test accuracy of 58.35%. **Example 2:** Network 1 is a ResNeXt trained on CIFAR-10 dataset with 50% random labels and has test accuracy of 58.85%. Network 2 is a ResNeXt that is not pre-trained and has test accuracy of 9.74%. **Example 3:** Network 1 is a SENet trained on CIFAR-10 dataset with original labels and has test accuracy of 95.35%. Network 2 is a SENet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 56.38%. **Example 4:** Network 1 is a RegNet trained on CIFAR-10 dataset with original labels and has test accuracy of 95.28%. Network 2 is a RegNet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 55.36%. **Example 5:** Network 1 is a MobileNet trained on CIFAR-10 dataset with original labels and has test accuracy of 90.56%. Network 2 is a MobileNet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 82.76%.



(a) Example 1: original label: ship, assigned label: dog

(b) Example 2: original label: cat, assigned label: truck

Figure 4.11: The evolution of output prediction of networks that are trained on a single randomly labeled sample. In all sub-figures, Network 1 (trained on the clean dataset) has a higher test accuracy than Network 2 (trained on the noisy dataset), and we observe it is more resistant to memorization of the single incorrectly-labeled sample. Furthermore, we have ensured using the temperature scaling method [Guo et al. 2017] that the two models have the same calibration (ECE) value. **Example 1:** Network 1 is a GoogleNet trained on CIFAR-10 dataset with original labels and has test accuracy of 95.36%. Network 2 is a GoogleNet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 58.35%. **Example 2:** Network 1 is a RegNet trained on CIFAR-10 dataset with original labels and has test accuracy of 95.28%. Network 2 is a RegNet that is trained on CIFAR-10 with 50% label noise and has test accuracy of 55.36%.

4.E Additional Experiments for Section 4.4

In this section, we provide additional experiments for our main results for MNIST, Fashion-MNIST, SVHN, CIFAR-100, and Tiny Imagenet datasets.

In Fig. 4.12, we observe that for networks trained on the noisy MNIST datasets, models that are resistant to memorization and are trainable have on average more than 20% higher test accuracy compared to models that are trainable but not resistant (similar results for other datasets are observed in Fig. 4.14, Fig. 4.16, Fig. 4.19, and Fig. 4.22). Furthermore, without access to the ground-truth, the models with a high (respectively, low) accuracy on the clean (resp., noisy) subsets are recovered using susceptibility ζ as shown in Fig. 4.17 and Fig. 4.20. Moreover, in Fig. 4.13, we observe that by selecting models with a low value of $\zeta(t)$ the correlation between training accuracy and test accuracy drastically increases from -0.766 to 0.863, which shows the effectiveness of the susceptibility metric $\zeta(t)$ (similar results for other datasets are observed in Fig. 4.15, Fig. 4.18, Fig. 4.21, and Fig. 4.23).

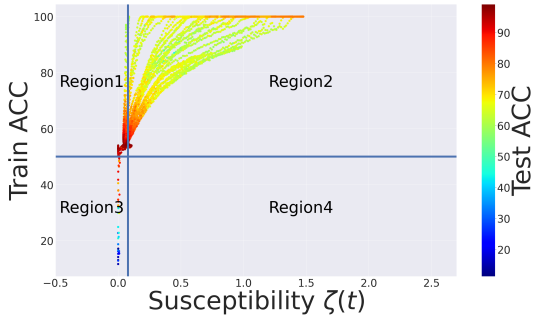
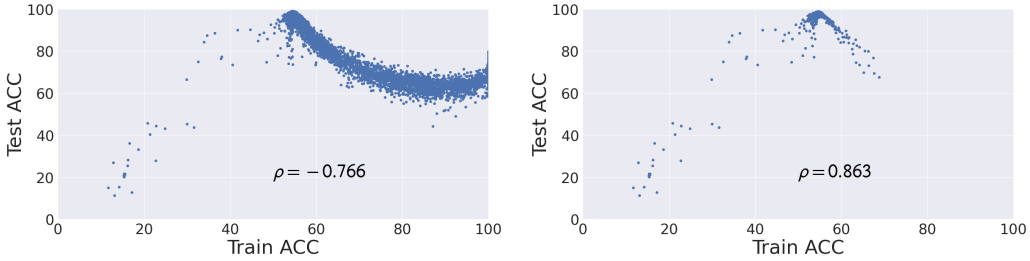


Figure 4.12: Using susceptibility $\zeta(t)$ and training accuracy, we can obtain 4 different regions for models trained on MNIST with 50% label noise (details in Section 4.B). **Region 1**: Trainable and resistant, with average test accuracy of 95.38%. **Region 2**: Trainable and but not resistant, with average test accuracy of 72.65%. **Region 3**: Not trainable but resistant, with average test accuracy of 47.69%. **Region 4**: Neither trainable nor resistant.



(a) Test accuracy versus train accuracy for all models at all epochs

(b) Test accuracy versus train accuracy for models that have $\zeta(t) \leq 0.05$

Figure 4.13: For models trained on MNIST with 50% label noise (details in Section 4.B), the correlation between training accuracy and test accuracy increases a lot by removing models based on the susceptibility metric $\zeta(t)$.

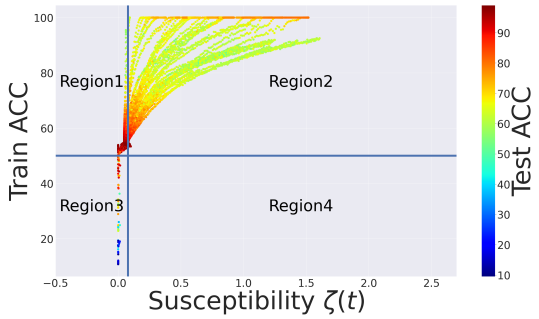
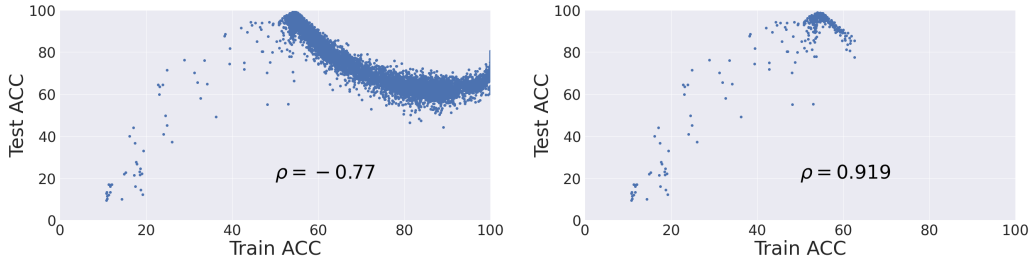


Figure 4.14: Using susceptibility $\zeta(t)$ and training accuracy we can obtain 4 different regions for models trained on Fashion-MNIST with 50% label noise (details in Section 4.B). **Region 1**: Trainable and resistant, with average test accuracy of 95.82%. **Region 2**: Trainable and but not resistant, with average test accuracy of 72.04%. **Region 3**: not trainable but resistant, with average test accuracy of 52.68%. **Region 4**: Neither trainable nor resistant.



(a) Test accuracy versus train accuracy for all models at all epochs

(b) Test accuracy versus train accuracy for models that have $\zeta(t) \leq 0.05$

Figure 4.15: For models trained on Fashion-MNIST with 50% label noise (details in Section 4.B), the correlation between training accuracy and test accuracy increases a lot by removing models based on the susceptibility metric $\zeta(t)$.

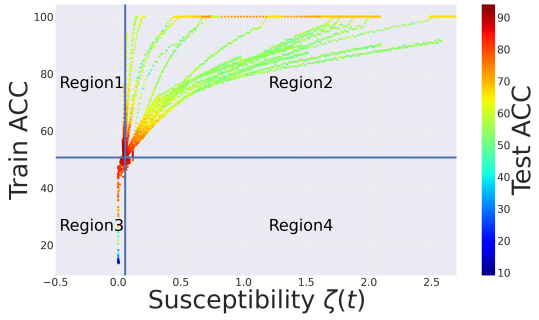
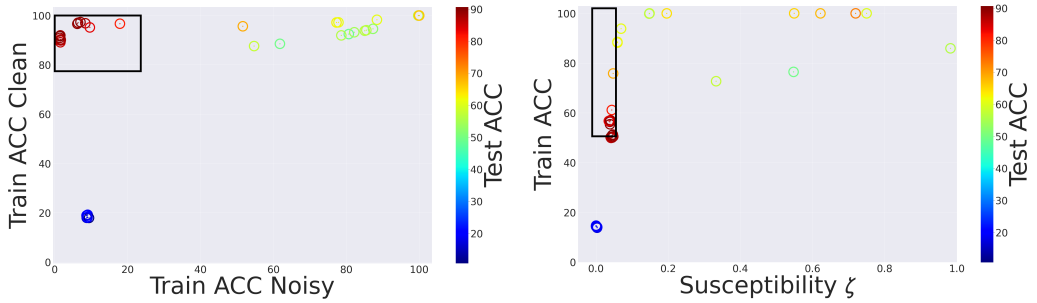


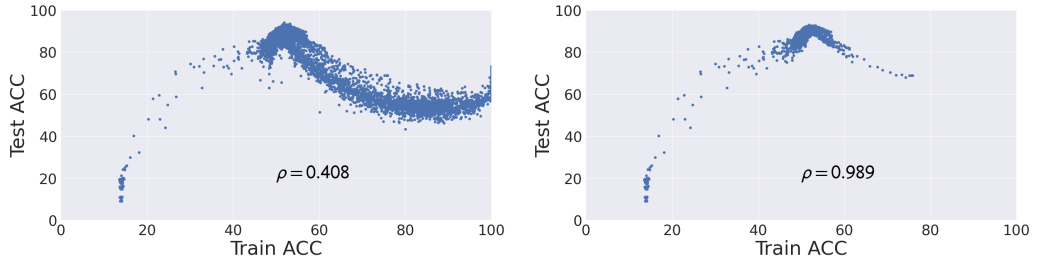
Figure 4.16: Using susceptibility $\zeta(t)$ and training accuracy we can obtain 4 different regions for models trained on SVHN with 50% label noise (details in Section 4.B). **Region 1**: Trainable and resistant, with average test accuracy of 88.64%. **Region 2**: Trainable and but not resistant, with average test accuracy of 66.34%. **Region 3**: Not trainable but resistant, with average test accuracy of 53.25%. **Region 4**: Neither trainable nor resistant, with average test accuracy of 85.17%.



(a) With oracle access to the ground-truth label. Average test accuracy of the selected models = 87.56%

(b) Without access to the ground-truth label. Average test accuracy of the selected models = 85.44%

Figure 4.17: For models trained on SVHN with 50% label noise (details in Section 4.B), with the help of our susceptibility metric $\zeta(t)$ and the overall training accuracy, we can recover models with a high/low accuracy on the clean/noisy subset.



(a) Test accuracy versus train accuracy for all models at all epochs

(b) Test accuracy versus train accuracy for models that have $\zeta(t) \leq 0.05$

Figure 4.18: For models trained on SVHN with 50% label noise (details in Section 4.B), the correlation between training accuracy and test accuracy increases a lot by removing models based on the susceptibility metric $\zeta(t)$.

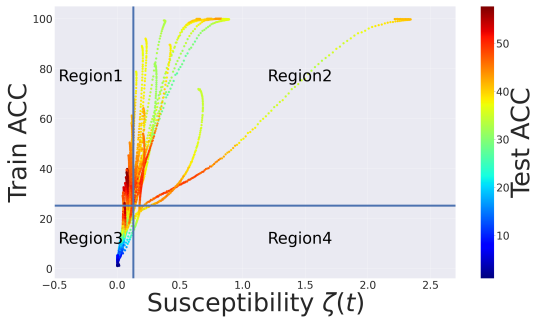
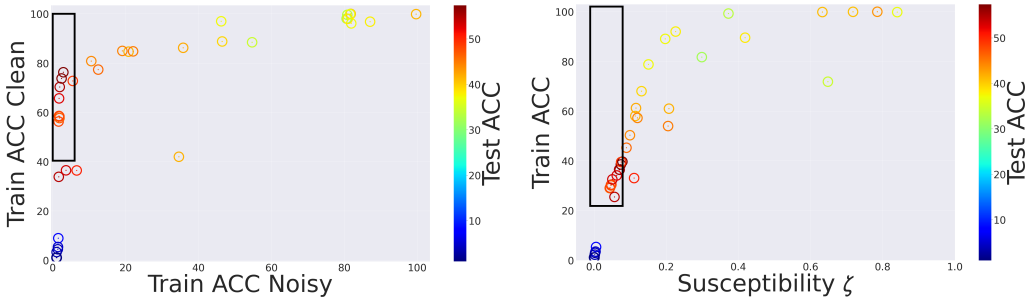


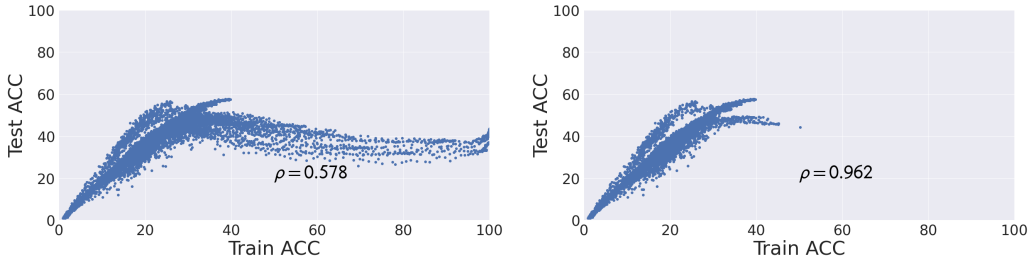
Figure 4.19: Using susceptibility $\zeta(t)$ and training accuracy we can obtain 4 different regions for models trained on CIFAR-100 with 50% label noise (details in Section 4.B). **Region 1**: Trainable and resistant, with average test accuracy of 47.09%. **Region 2**: Trainable and but not resistant, with average test accuracy of 40.96%. **Region 3**: Not trainable but resistant, with average test accuracy of 22.65%. **Region 4**: Neither trainable nor resistant, with average test accuracy of 39.07%.



(a) With oracle access to the ground-truth label. Average test accuracy of the selected models = 51.39%

(b) Without access to the ground-truth label. Average test accuracy of the selected models = 49.57%

Figure 4.20: For models trained on CIFAR-100 with 50% label noise, with the help of our susceptibility metric $\zeta(t)$ and the overall training accuracy, we can recover models with a high/low accuracy on the clean/noisy subset.



(a) Test accuracy versus train accuracy for all models at all epochs

(b) Test accuracy versus train accuracy for models that have $\zeta(t) \leq 0.1$

Figure 4.21: For models trained on CIFAR-100 with 50% label noise (details in Section 4.B), the correlation between training accuracy and test accuracy increases a lot by removing models based on the susceptibility metric $\zeta(t)$.

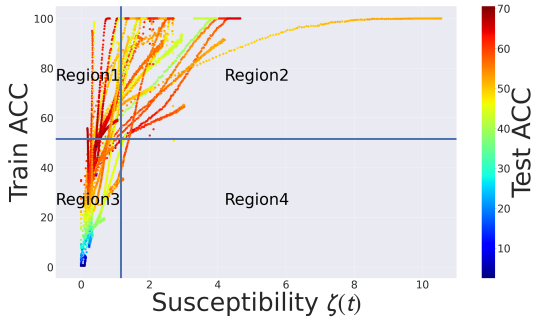
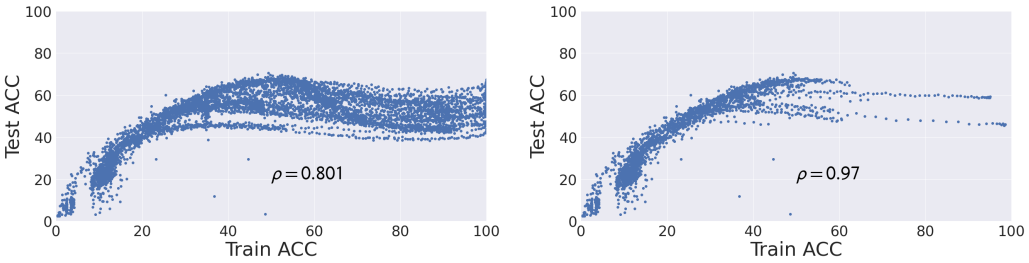


Figure 4.22: Using susceptibility $\zeta(t)$ and training accuracy we can obtain 4 different regions for models trained on Tiny Imagenet with 10% label noise (details in Section 4.B). **Region 1**: Trainable and resistant, with average test accuracy of 57.51%. **Region 2**: Trainable and but not resistant, with average test accuracy of 53.25%. **Region 3**: Not trainable but resistant, with average test accuracy of 18.53%. **Region 4**: Neither trainable nor resistant, with average test accuracy of 53.26%.



(a) Test accuracy versus train accuracy for all models at all epochs

(b) Test accuracy versus train accuracy for models that have $\zeta(t) \leq 0.05$

Figure 4.23: For models trained on Tiny Imagenet with 10% label noise (details in Section 4.B), the correlation between training accuracy and test accuracy increases by removing models based on the susceptibility metric $\zeta(t)$.

4.F Experiments Related to Section 4.7

In this section, we provide some ablation studies that are discussed in Section 4.7.

In Fig. 4.24, we observe that even among neural network architectures with a good resistance to memorization, susceptibility to noisy labels $\zeta(t)$ detects the most resistant model. We observe that the high correlation between ζ and memorization of the noisy subset is not limited to a specific learning rate schedule in Fig. 4.25, or a label noise level in Fig. 4.26 and Fig. 4.27. Moreover, in Fig. 4.28 and Fig. 4.29, we observe that for datasets with the label noise level of 10%, the susceptibility to noisy labels ζ and training accuracy still select models with high test accuracy. The same consistency is observed in Fig. 4.32 for models trained with asymmetric label noise.

In this chapter, we choose $\widetilde{S}$ to be only a single mini-batch of a randomly-labeled set for computational efficiency. But we also made sure that this does not harm the correlation between Train ACC Noisy and $\zeta(t)$. We analyze the effect of size of $\widetilde{S}$ in Fig. 4.30 (left), which confirms that a single mini-batch is large enough to have a high correlation between Train ACC Noisy and $\zeta(t)$. Moreover, we observe the robustness of the susceptibility metric to the exact choice of the mini-batch in Fig. 4.30 (right).

To better illustrate the match between Train ACC Noisy and $\zeta(t)$, we provide the overlaid curves in Fig. 4.31. This figure clearly shows how using ζ, one can detect/select checkpoints of the model with low memorization.

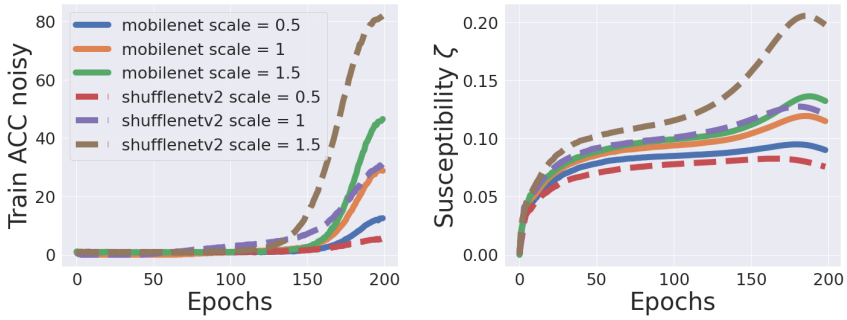
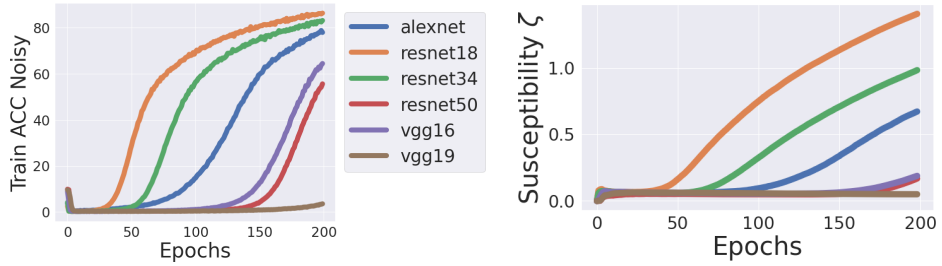
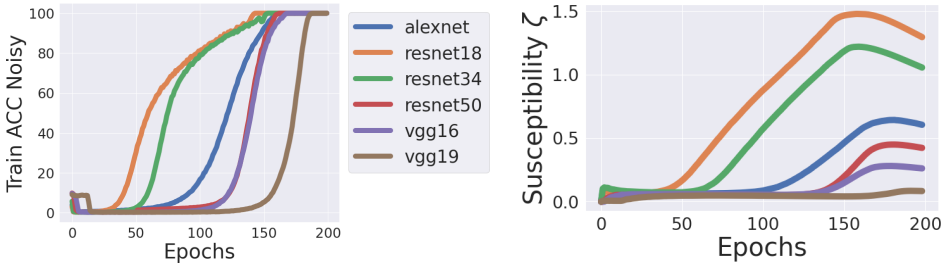


Figure 4.24: Accuracy on the noisy subset of the training set versus the susceptibility $\zeta(t)$ (Eq. (4.2)) for MobileNet and ShuffleNetV2 configurations trained on CIFAR-100 with 50% label noise. Pearson correlation between the Train ACC Noisy and susceptibility ζ is $\rho = 0.749$. `Scale` is a hyper-parameter that proportionally scales the number of hidden units and number of channels in the neural network configuration.



(a) `Exponential`



(b) `Cosineannealing`

Figure 4.25: Accuracy on the noisy subset of the training set versus the susceptibility $\zeta(t)$ for networks trained on MNIST with 50% label noise. On top and bottom, we have models trained with `exponential` and `cosineannealing` learning rate schedulers, respectively. Pearson correlation between Train ACC Noisy and ζ for `exponential` and `cosineannealing` schedules are $\rho = 0.89$ and $\rho = 0.772$, respectively.

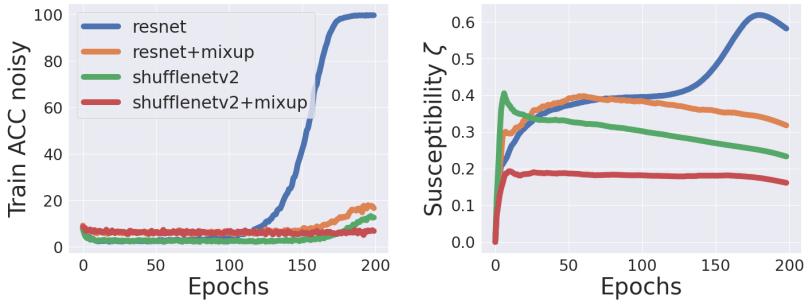


Figure 4.26: Accuracy on the noisy subset of the training set versus susceptibility to noisy labels $\zeta(t)$ for networks trained on CIFAR-10 with 10% label noise. Pearson correlation between Train ACC Noisy and ζ is $\rho = 0.634$.

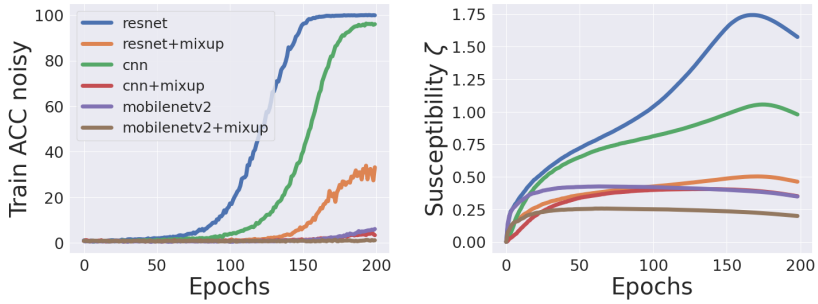
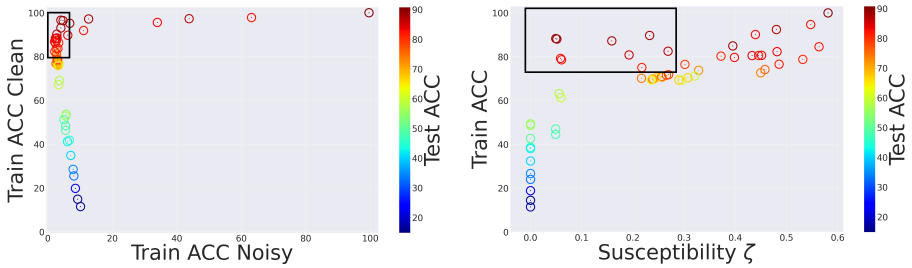


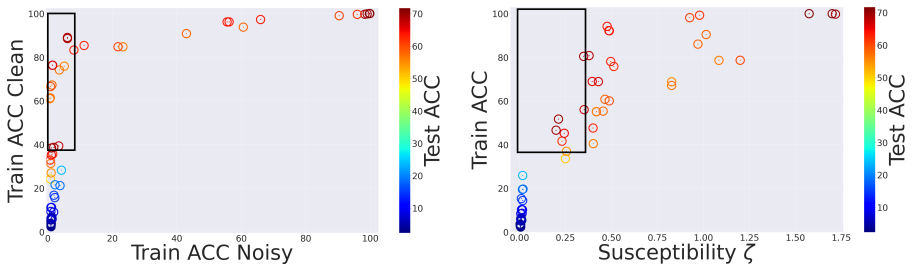
Figure 4.27: Accuracy on the noisy subset of the training set versus susceptibility to noisy labels $\zeta(t)$ for networks trained on CIFAR-100 with 10% label noise. Pearson correlation between Train ACC Noisy and ζ is $\rho = 0.849$.



(a) With access to the ground-truth label. Average test accuracy of the selected models = 82.93%

(b) Without access to the ground-truth label. Average test accuracy of the selected models = 86.08%

Figure 4.28: For models trained on CIFAR-10 with 10% label noise for 200 epochs, using susceptibility ζ and the overall training accuracy, the average test accuracy of the selected models is comparable with (even higher than) the case of having access to the ground-truth label.



(a) With access to the ground-truth label. Average test accuracy of the selected models = 62.61%

(b) Without access to the ground-truth label. Average test accuracy of the selected models = 64.33%

Figure 4.29: For models trained on CIFAR-100 with 10% label noise for 200 epochs, using susceptibility ζ and the overall training accuracy, the average test accuracy of the selected models is comparable with (even higher than) the case of having access to the ground-truth label.

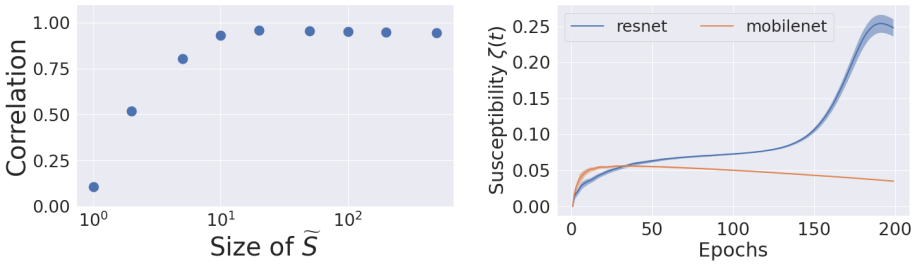


Figure 4.30: **Left**: Pearson correlation coefficient between the accuracy on the noisy subset of the training set and susceptibility ζ (Eq. (4.2)) for different choices of dataset size for $\widetilde{S}$ for ResNet [He et al. 2016a], MobileNet [Howard et al. 2017], and 5−layer cnn that are trained on CIFAR-100 dataset with 50% label noise. We observe that unless the dataset is very small, the choice of the dataset size $\widetilde{S}$ does not affect the correlation value. Therefore, throughout our experiments, we choose the size 128 for this set, which is the batch size used for the regular training procedure as well. Note that this size is very small compared to the size of the training set itself, which is 50000, hence the computational overhead to compute ζ is negligible compared to the original training process. **Right**: We can observe the variance of the susceptibility metric over 10 different random seeds. We can observe that as the variance is quite low, the metric is robust to the exact choice of the mini-batch and to the random labels that are assigned to the mini-batch.

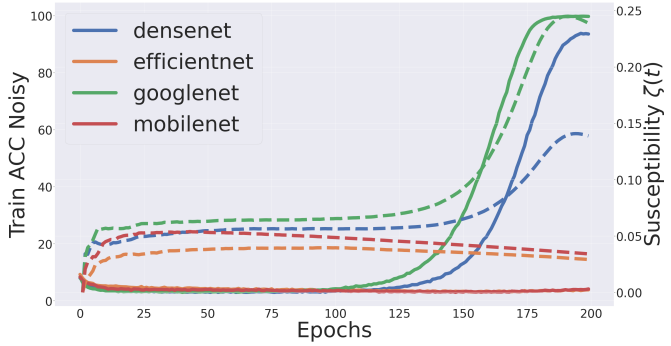
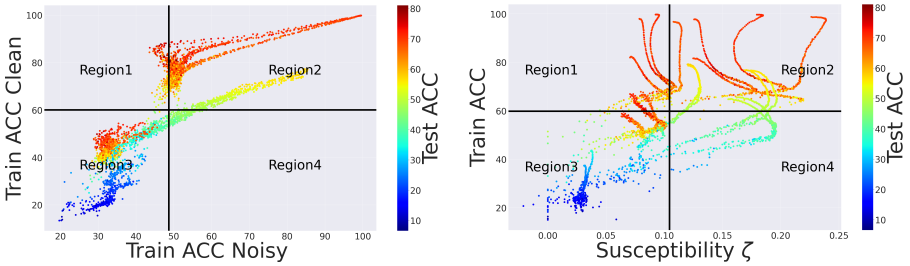


Figure 4.31: Accuracy on the noisy subset (solid lines) versus Susceptibility $\zeta(t)$ (dashed lines) for neural networks trained on CIFAR-10 with 50% label noise. We observe a very strong match between the two, which suggests that susceptibility can be used to perform early stopping by selecting the checkpoint for each model with the least memorization. For example, for MobileNet and EfficientNet, ζ does not warn about memorization, hence one can select the end checkpoint. On the other hand, for DenseNet and GoogleNet, ζ suggests selecting those checkpoints that are before the sharp increases. This is also consistent with the signal given by the fit on the noisy subset, which requires ground-truth label access, unlike susceptibility ζ which does not require such access.



(a) With access to the ground-truth label. Average test accuracy of the selected models = 66.793%

(b) Without access to the ground-truth label. Average test accuracy of the selected models = 66.307%

Figure 4.32: For models trained for 200 epochs on CIFAR-10 with 50% asymmetric label noise as proposed in [Xia et al. 2021], using susceptibility ζ and the overall training accuracy, the average test accuracy of the selected models is comparable with the case of having access to the ground-truth label.

A study on different thresholds used to select models We would like to point out that if we can tune these thresholds (instead of using the average values of training accuracy and susceptibility over the available models), we can select models with even higher test accuracies than what is reported in this chapter. For example, for models of Fig. 4.5a, by tuning these two thresholds one could reach a test accuracy of 79.15% (instead of the reported 76%) as shown in Fig. 4.33 (left). However, we want to remain in the practical setting where we do not have any access to a clean validation set for tuning. As a consequence, we must avoid any hyper-parameter tuning. And indeed, throughout our experiments, these thresholds are never tuned nor set manually to any extent. Among thresholds that can be computed without access to a clean validation set, we opted for the average values of susceptibility and training accuracy (over the available models) for simplicity. We empirically observe that this choice is robust and produces favorable results in various experimental settings. We could take other percentiles for the threshold, but they are more complex to obtain than simple averages, because they would then depend on the distribution among models. In Fig. 4.33, we study various values of percentiles for these thresholds. We observe that depending on the available models and the given dataset, some other percentiles might give higher test accuracies than simply using the average values. These percentiles range however typically from 35 to 55 and are therefore not far from the mean, hence their benefit in increasing test accuracy appears small compared to the increased complexity to compute them or relying on additional assumptions on the distribution of susceptibility and training accuracy. We observe in Fig. 4.33 (right) that except for very extreme values of the thresholds (which basically select all models as resistant to memorization), the average test accuracy of models in Region 1 is much higher than the average test accuracy of models in Region 2. Hence, our proposed model selection approach is robust to the choice of these thresholds.

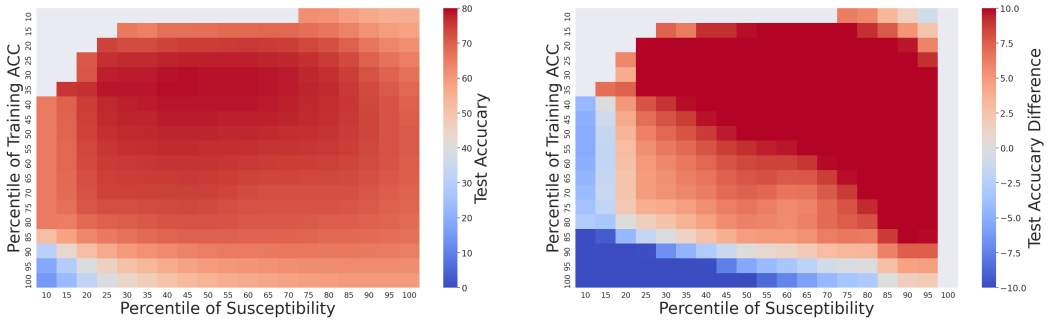


Figure 4.33: **Left:** Average test accuracy of models in Region 1 of Fig. 4.5a for various thresholds used to find Region 1. In Fig. 4.5a and throughout this chapter, Region 1 has models with susceptibility ζ < t1 and training accuracy > t2, where t1 and t2 are average ζ and training accuracy over the available models, respectively. Here, we study different values of these thresholds t1 and t2, and their effect on the average test accuracy of models of Region 1. We explore different percentiles of ζ and training accuracy over all models to be used to find these thresholds. The extreme would be to have 100th percentiles for both thresholds (low rightmost item of this table), which means models of Region 1 have ζ < maximum susceptibility and training accuracy > minimum training accuracy. In this extreme case, all models are selected in Region 1. Overall, we observe that some other percentiles might give higher test accuracies than simply using the average values. These percentiles range however typically from 35 to 55 and are therefore not far from the mean, hence their benefit in increasing test accuracy appears small compared to the increased complexity to compute them or relying on additional assumptions on the distribution of susceptibility and training accuracy. **Right:** The difference in the average test accuracies of models in Region 1 and models in Region 2 for various values of percentiles used to find different regions. A positive value implies that models in Region 1 have a higher average test accuracy. We can observe that except for very extreme values of the thresholds, which basically select all models as trainable, the average test accuracy of models in Region 1 is much higher than the average test accuracy of models in Region 2. Hence, our approach to select *resistant and trainable* models is robust to the choice of these thresholds.

4.G Theoretical Preliminaries

In this section, we provide some technical tools that we use throughout our proofs. Recall from Section 4.2 that the first layer weights of the neural network are initialized as

$$\theta_r(0) \sim \mathcal{N}\left(\mathbf{0}, \kappa^2 \mathbf{I}\right), \quad \forall r \in [m], \tag{4.7}$$

where $0 < \kappa \leq 1$ is the magnitude of initialization and $\mathcal{N}$ denotes the normal distribution. And the second layer weights a_rs are independent random variables taking values uniformly in $\{-1, 1\}$ at initialization.

4.G.1 Properties of the Gram-matrix

Properties Here, we recall a few useful properties of the Gram-matrix (Eq. (4.3)).

1. As shown by [Du et al. 2018], $\mathbf{H}^\infty$ is positive definite and $\lambda_0 = \lambda_{\min}(\mathbf{H}^\infty) > 0$.

2. The matrix $\mathbf{H}^\infty$ has eigen decomposition $\mathbf{H}^\infty = \sum_{i=1}^n \lambda_i \mathbf{v}_i \mathbf{v}_i^T$, where the eigenvectors are orthonormal. Therefore, $\mathbf{v}_i^T \mathbf{v}_j = \delta_{i,j}$ for $i, j \in [n]$, the $n \times n$ identity matrix $\mathbf{I}$ is decomposed as $\sum_{i=1}^n \mathbf{v}_i \mathbf{v}_i^T$ and any n-dimentional (column-wise) vector $\mathbf{y}$ can be decomposed as $\mathbf{I}\,\mathbf{y} = \sum_{i=1}^n (\mathbf{v}_i^T \mathbf{y}) \mathbf{v}_i$.

3. (Recalled from [Du et al. 2018; Arora et al. 2019]) We have $\|\mathbf{H}^\infty\|_2 \leq \mathrm{tr}(\mathbf{H}^\infty) = \frac{n}{2} = \sum_{i=1}^n \lambda_i$, and
$$\gamma = O(\frac{\lambda_0}{n^2}) = O\left(\frac{\lambda_{\min}(\mathbf{H}^\infty)}{\|\mathbf{H}^\infty\|_2^2}\right) \leq \frac{1}{\|\mathbf{H}^\infty\|_2}.$$
Hence,
$$\left\|\mathbf{I} - \gamma \mathbf{H}^\infty\right\|_2 \leq 1 - \gamma\lambda_0.$$

4.G.2 Corollaries Adapted from [Du et al. 2018; Arora et al. 2019]

Corollary 1. *(Adapted Theorem 3.1 of [Arora et al. 2019] to our setting) For $m = \Omega\left(\frac{n^6}{\lambda_0^4 \kappa^2 \delta^3}\right)$ and $\gamma = O\left(\frac{\lambda_0}{n^2}\right)$, for any $\delta \in (0, 1]$, with probability at least $1 - \delta$ over random initialization Eq. (4.7):*

$$\Phi\left(\Theta(0)\right) = O\left(\frac{n}{\delta}\right),$$

and

$$\begin{cases} \Phi\left(\Theta(t+1)\right) \leq (1 - \frac{\eta\lambda_0}{2}) \Phi(\Theta(t)), & \text{if } 0 \leq t < k, \\ \widetilde{\Phi}\left(\Theta(t+1)\right) \leq (1 - \frac{\eta\lambda_0}{2}) \widetilde{\Phi}(\Theta(t)), & \text{if } k \leq t < k + \tilde{k}. \end{cases}$$

Therefore, by replacing Eq. (4.1), throughout the proof we can use:

$$\left\|\mathbf{f}_{\Theta(0)}-\mathbf{y}\right\|_2 = O\left(\sqrt{\frac{n}{\delta}}\right),$$

and

$$\begin{cases} \left\|\mathbf{f}_{\Theta(t+1)}-\mathbf{y}\right\|_2 & \leq \sqrt{1-\frac{\gamma\lambda_0}{2}}\left\|\mathbf{f}_{\Theta(t)}-\mathbf{y}\right\|_2 \\ & \leq \left(1-\frac{\gamma\lambda_0}{4}\right)\left\|\mathbf{f}_{\Theta(t)}-\mathbf{y}\right\|_2, \quad \text{if } 0\leq t<k, \\ \left\|\mathbf{f}_{\Theta(t+1)}-\widetilde{\mathbf{y}}\right\|_2 & \leq \left(1-\frac{\gamma\lambda_0}{4}\right)\left\|\mathbf{f}_{\Theta(t)}-\widetilde{\mathbf{y}}\right\|_2, \quad \text{if } k\leq t<k+\tilde{k}, \end{cases}$$

where we use inequality $\sqrt{1-\alpha}\leq 1-\alpha/2$, which holds for $0\leq\alpha\leq 1$.

Corollary 2. *(Adapted from Equation (25) of [Arora et al. 2019]) If the parameter vector is updated at step t by one gradient descent step on $\frac{1}{2}\left\|\mathbf{f}_{\Theta(t)}-\mathbf{u}\right\|_2^2$ for some label vector $\mathbf{u}$, and for t such that with probability at least $1-\delta$:*

$$\left\|\mathbf{H}(t)-\mathbf{H}(0)\right\|_F = O\left(\frac{n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right),$$

then the output of the neural network is as follows

$$\mathbf{f}_{\Theta(t+1)}-\mathbf{f}_{\Theta(t)} = -\gamma\mathbf{H}^{\infty}\left(\mathbf{f}_{\Theta(t)}-\mathbf{u}\right)+\xi(t),$$

where $\xi(\cdot)$ is considered to be a perturbation term that can be bounded with probability at least $1-\delta$ over random initialization Eq. (4.7) by

$$\left\|\xi(t)\right\|_2 = O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right)\left\|\mathbf{f}_{\Theta(t)}-\mathbf{u}\right\|_2. \tag{4.8}$$

Remark: In our setting, this corollary holds for $0\leq t\leq k-1$ with $\mathbf{u}=\mathbf{y}$, and for $k\leq t\leq k+\tilde{k}-1$ with $\mathbf{u}=\widetilde{\mathbf{y}}$. We only need to show that for our setting for $t\geq k$, $\left\|\mathbf{H}(t)-\mathbf{H}(0)\right\|_F$ is bounded, which is done in Lemma 3.

Corollary 3. *(From Equation (27) of [Arora et al. 2019]) We have for $1\leq t\leq k$*

$$\mathbf{f}_{\Theta(t)}-\mathbf{y} = \left(\mathbf{I}-\gamma\mathbf{H}^{\infty}\right)^t\left(\mathbf{f}_{\Theta(0)}-\mathbf{y}\right)+\sum_{s=0}^{t-1}\left(\mathbf{I}-\eta\mathbf{H}^{\infty}\right)^s\xi\left(t-s-1\right),$$

where $\left\|\xi(\cdot)\right\|_2$ is some perturbation term that can be bounded using Eq. (4.8) with $\mathbf{u}=\mathbf{y}$.

4.G.3 Additional Lemmas

Lemma 1. *For the setting described in Section 4.2, we have*

$$\mathbf{f}_{\Theta(k)}-\widetilde{\mathbf{y}} = \sum_{i=1}^{n}\left[\left(\mathbf{v}_i^T\mathbf{y}\right)-\left(1-\gamma\lambda_i\right)^k\left(\mathbf{v}_i^T\mathbf{y}\right)-\left(\mathbf{v}_i^T\widetilde{\mathbf{y}}\right)\right]\mathbf{v}_i+\chi(k), \tag{4.9}$$

where $\chi(k)$ is some perturbation term that with probability at least $1-\delta$ over the random initialization Eq. (4.7)

$$\left\|\chi(k)\right\|_2 = O\left(\frac{n^{3/2}\kappa}{\sqrt{\delta}\lambda_0} + \frac{n^{9/2}}{\sqrt{m}\lambda_0^3\kappa\delta^2}\right). \tag{4.10}$$

Lemmas to Bound $\mathbf{H}(t)$ with $\mathbf{H}^\infty$

Because the two datasets S and $\widetilde{S}$ have the same input samples, the Gram matrix defined in Eq. (4.3) is the same for both of them. We now recall two lemmas from [Du et al. 2018; Arora et al. 2019] and provide a lemma extending them to bound $\mathbf{H}(t)$ with $\mathbf{H}^\infty$, where

$$\mathbf{H}_{ij}(t) = \frac{\mathbf{x}_i^T\mathbf{x}_j}{m}\sum_{r=1}^{m}\mathbb{I}_{r,i}(t)\mathbb{I}_{r,j}(t),$$

and $\mathbb{I}_{r,i}(t) = \mathbb{I}\{\theta_r^T(t)\mathbf{x}_i \geq 0\}$.

Lemma 2. *(recalled from [Du et al. 2018; Arora et al. 2019]) For $\lambda_0 = \lambda_{min}(\boldsymbol{H}^\infty) > 0$, $m = \Omega\left(\frac{n^6}{\lambda_0^4\kappa^2\delta^3}\right)$, and $\gamma = O\left(\frac{\lambda_0}{n^2}\right)$ with probability at least $1-\delta$ over the random initialization Eq.* (4.7)*, for all $0 \leq t \leq k$, we have:*

$$\|\mathbf{H}(t) - \mathbf{H}(0)\|_F = O\left(\frac{n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right).$$

Lemma 3. *(Our extension) For $\lambda_0 = \lambda_{min}(\boldsymbol{H}^\infty) > 0$, $m = \Omega\left(\frac{n^6}{\lambda_0^4\kappa^2\delta^3}\right)$, and $\gamma = O\left(\frac{\lambda_0}{n^2}\right)$ with probability at least $1-\delta$ over the random initialization Eq.* (4.7)*, for all $k+1 \leq t \leq k+\tilde{k}$, we have:*

$$\|\mathbf{H}(t) - \mathbf{H}(0)\|_F = O\left(\frac{n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right).$$

Lemma 4. *(recalled from [Du et al. 2018; Arora et al. 2019]) With probability at least $1-\delta$ over the random initialization Eq.* (4.7)*, we have:*

$$\left\|\mathbf{H}(0) - \mathbf{H}^\infty\right\|_F = O\left(\frac{n\sqrt{\log\frac{n}{\delta}}}{\sqrt{m}}\right).$$

Remark for Lemma 4: The indicator function $\mathbb{I}\{\theta_r(t)^T\mathbf{x}_i \geq 0\}$ is invariant to the scale κ of θ_r, hence $\mathbb{E}[\mathbf{H}_{ij}(0)] = \mathbf{H}_{ij}^\infty$, even though the expectation on the left hand side is taken with respect to $\theta \sim \mathcal{N}(\mathbf{0}, \kappa^2\mathbf{I})$ and the expectation on the right hand side is taken with respect to $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

4.H Proof of Lemma 3

Proof. We recall from the proof of Lemma C.2 of [Arora et al. 2019] that if with probability at least $1-\delta$, $\|\theta_r(t)-\theta_r(0)\|_2 \le R$, then with probability at least $1-\delta$, we have $\|\mathbf{H}(t)-\mathbf{H}(0)\|_F \le \frac{4n^2R}{\sqrt{2\pi\kappa\delta}} + \frac{2n^2}{m}$. So, we first find an upper bound on $\|\theta_r(t)-\theta_r(0)\|_2$ for $t > k$ and replace its value R in $\frac{4n^2R}{\sqrt{2\pi\kappa\delta}} + \frac{2n^2}{m}$.

To find an upper bound on $\|\theta_r(t)-\theta_r(0)\|_2$ for $t > k$, we can follow a similar approach as in the proof of Lemma C.1 of [Arora et al. 2019]:

$$
\begin{aligned}
\|\theta_r(t)-\theta_r(0)\|_2 &\le \sum_{\tau=0}^{t-1} \|\theta_r(\tau+1)-\theta_r(\tau)\|_2 \\
&= \sum_{\tau=0}^{k-1} \|\theta_r(\tau+1)-\theta_r(\tau)\|_2 + \sum_{\tau=k}^{t-1} \|\theta_r(\tau+1)-\theta_r(\tau)\|_2 \\
&\overset{(a)}{\le} \sum_{\tau=0}^{k-1} \frac{\gamma\sqrt{n}}{\sqrt{m}} \left\|\mathbf{f}_{\Theta(\tau)}-\mathbf{y}\right\|_2 + \sum_{\tau=k}^{t-1} \frac{\gamma\sqrt{n}}{\sqrt{m}} \left\|\mathbf{f}_{\Theta(\tau)}-\widetilde{\mathbf{y}}\right\|_2 \\
&\overset{(b)}{\le} \sum_{\tau=0}^{k-1} \frac{\gamma\sqrt{n}}{\sqrt{m}} \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tau} \left\|\mathbf{f}_{\Theta(0)}-\mathbf{y}\right\|_2 \\
&\quad + \sum_{\tau=k}^{t-1} \frac{\gamma\sqrt{n}}{\sqrt{m}} \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tau-k} \left\|\mathbf{f}_{\Theta(k)}-\widetilde{\mathbf{y}}\right\|_2 \\
&\overset{(c)}{\le} O\left(\frac{\gamma n}{\sqrt{m\delta}}\right) \sum_{\tau=0}^{k-1} \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tau} \\
&\quad + \sum_{\tau=k}^{t} \frac{\gamma\sqrt{n}}{\sqrt{m}} \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tau-k} \left[\left\|\mathbf{f}_{\Theta(k)}-\mathbf{y}\right\|_2 + \left\|\widetilde{\mathbf{y}}-\mathbf{y}\right\|_2\right] \\
&\overset{(d)}{\le} O\left(\frac{\gamma n}{\sqrt{m\delta}}\right) \sum_{\tau=0}^{k-1} \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tau} + O\left(\frac{\gamma n}{\sqrt{m\delta}}\right) \sum_{\tau=0}^{t-k} \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tau} \\
&\le O\left(\frac{\gamma n}{\sqrt{m\delta}}\right) \sum_{\tau=0}^{\infty} \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tau} \le O\left(\frac{n}{\lambda_0\sqrt{m\delta}}\right),
\end{aligned}
\tag{4.11}
$$

where (a) holds because for an update step on label vector $\mathbf{u}$ according to gradient descent, we have

$$
\begin{aligned}
\|\theta_r(\tau+1)-\theta_r(\tau)\| &= \left\| \frac{\gamma}{\sqrt{m}} a_r \sum_{i=1}^{n} \left(f_{\Theta(\tau)}(\mathbf{x}_i)-u_i\right) \mathbb{I}_{r,i}(k) \mathbf{x}_i \right\| \\
&\le \frac{\gamma}{\sqrt{m}} \sum_{i=1}^{n} \left|f_{\Theta(\tau)}(\mathbf{x}_i)-u_i\right| \le \frac{\gamma\sqrt{n}}{\sqrt{m}} \left\|\mathbf{f}_{\Theta(\tau)}-\mathbf{u}\right\|,
\end{aligned}
$$

inequalities (b) and (c) use Corollary 1. Inequality (d) holds because we have $\left|y_i\right| \le 1$ and $\left|\widetilde{y}_i\right| = 1$, so again by using Corollary 1

$$
\left\|\mathbf{f}_{\Theta(k)}-\mathbf{y}\right\|_2 + \left\|\widetilde{\mathbf{y}}-\mathbf{y}\right\|_2 \le \left(1-\frac{\gamma\lambda_0}{4}\right)^{k} O\left(\sqrt{\frac{n}{\delta}}\right) + O(\sqrt{n}) = O\left(\sqrt{\frac{n}{\delta}}\right). \tag{4.12}
$$

Therefore, with probability at least $1-\delta$ over random initialization Eq. (4.7) for $t > k$

$$\|\mathbf{H}(t) - \mathbf{H}(0)\|_F \leq \frac{4n^2 R}{\sqrt{2\pi}\kappa\delta} + \frac{2n^2}{m} = O\left(\frac{n^3}{\kappa\lambda_0\delta^{3/2}\sqrt{m}}\right),$$

where we replaced R with the upper bound in Eq. (4.11).

□

4.I Proof of Lemma 1

Proof. Note that throughout the proof, we refer to events with probability at least $1-\delta$, as high probability events. Using the union bound, the probability of intersection of α high-probability events is an event with probability at least $1-\alpha\delta$. Therefore, we can again refer to this event as a high probability event with probability at least $1-\delta$, but re-scale δ in the event accordingly. Because δ only appears on bounds of the perturbation terms, and therefore in the form of $O(\delta^{-1})$, then re-scaling δ would not change the order of these perturbation terms. Hence, throughout the proof we do not put concerns on the exact probability of events, we only refer to them as high probability events, and eventually we know that the probability of our computations is at least $1-\delta$ over the random initialization Equation (4.7).

Because Lemma 2 holds for $t = k-1$, we use Corollary 2 with $t = k-1$ and $\mathbf{u} = \mathbf{y}$:

$$\mathbf{f}_{\Theta(k)} - \mathbf{f}_{\Theta(k-1)} = -\eta\mathbf{H}^{\infty}\left(\mathbf{f}_{\Theta(k-1)} - \mathbf{y}\right) + \xi(k-1),$$

where with probability at least $1-\delta$

$$\|\xi(k-1)\|_2 = O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right)\left\|\mathbf{f}_{\Theta(k-1)} - \mathbf{y}\right\|_2,$$

because of Eq. (4.8). We then compute

$$\begin{aligned}\mathbf{f}_{\Theta(k)} - \widetilde{\mathbf{y}} &= \mathbf{f}_{\Theta(k-1)} - \widetilde{\mathbf{y}} - \gamma\mathbf{H}^{\infty}\left(\mathbf{f}_{\Theta(k-1)} - \mathbf{y}\right) + \xi(k-1) \\ &= \mathbf{f}_{\Theta(0)} - \widetilde{\mathbf{y}} - \gamma\sum_{t=0}^{k-1}\mathbf{H}^{\infty}\left(\mathbf{f}_{\Theta(t)} - \mathbf{y}\right) + \sum_{t=0}^{k-1}\xi(t). \end{aligned} \tag{4.13}$$

Let

$$\begin{aligned}\chi(k) = &-\gamma\sum_{t=0}^{k-1}\mathbf{H}^{\infty}\left(\mathbf{I} - \gamma\mathbf{H}^{\infty}\right)^t\mathbf{f}_{\Theta(0)} \\ &-\gamma\sum_{t=1}^{k-1}\sum_{s=0}^{t-1}\mathbf{H}^{\infty}\left(\mathbf{I} - \gamma\mathbf{H}^{\infty}\right)^s\xi\left(t-s-1\right) + \sum_{t=0}^{k-1}\xi(t) + \mathbf{f}_{\Theta(0)}. \end{aligned} \tag{4.14}$$

Then Eq. (4.13) becomes:

$$\mathbf{f}_{\Theta(k)} - \widetilde{\mathbf{y}} = \gamma \sum_{t=0}^{k-1} \mathbf{H}^{\infty} \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{t} \mathbf{y} - \widetilde{\mathbf{y}} + \chi(k), \tag{4.15}$$

where we have replaced $\mathbf{f}_{\Theta(t)} - \mathbf{y}$ for $1 \leq t \leq k$ in Eq. (4.13) using Corollary 3 by

$$\left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{t} \left(\mathbf{f}_{\Theta(0)} - \mathbf{y}\right) + \sum_{s=0}^{t-1} \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{s} \xi(t-s-1).$$

Next, we would like to find an upper bound for $\left\| \chi(k) \right\|$. As shown in Du et al. [2018]; Arora et al. [2019], with probability at least $1-\delta$ over random initialization Eq. (4.7), we have

$$\left\| \mathbf{f}_{\Theta(0)} \right\|_2^2 \leq \frac{n\kappa^2}{\delta}. \tag{4.16}$$

The first term in $\chi(k)$ (Section 4.I) is bounded because of Eq. (4.16) from above with high probability as

$$\begin{aligned}
\left\| \gamma \sum_{t=0}^{k-1} \mathbf{H}^{\infty} \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{t} \mathbf{f}_{\Theta(0)} \right\|_2 &\leq \gamma \sum_{t=0}^{k-1} \left\| \mathbf{H}^{\infty} \right\|_2 \left\| \mathbf{I} - \gamma \mathbf{H}^{\infty} \right\|_2^{t} \left\| \mathbf{f}_{\Theta(0)} \right\|_2 \\
&\leq \gamma \sum_{t=0}^{k-1} \frac{n}{2} \left(1 - \gamma \lambda_0\right)^{t} O\left(\frac{\sqrt{n}\kappa}{\sqrt{\delta}}\right) \\
&= O\left(\frac{n^{3/2}\kappa}{\sqrt{\delta}\lambda_0}\right),
\end{aligned}$$

where the second inequality uses Property 3.

The second term of $\chi(k)$ in Section 4.I can also be bounded with high probability by

$$\begin{aligned}
&\left\| \gamma \sum_{t=1}^{k-1} \sum_{s=0}^{t-1} \mathbf{H}^{\infty} \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^s \xi(t-s-1) \right\|_2 \\
&\quad \leq \gamma \sum_{t=1}^{k-1} \sum_{s=0}^{t-1} \left\|\mathbf{H}^{\infty}\right\|_2 \left\|\mathbf{I} - \gamma \mathbf{H}^{\infty}\right\|_2^s \|\xi(t-s-1)\|_2 \\
&\quad \overset{(a)}{\leq} \eta \sum_{t=1}^{k-1} \sum_{s=0}^{t-1} \frac{n}{2} \left(1-\gamma\lambda_0\right)^s O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0 \kappa \delta^{3/2}}\right) \left\|\mathbf{f}_{\Theta(t-s-1)} - \mathbf{y}\right\|_2 \\
&\quad \overset{(b)}{\leq} \gamma \sum_{t=1}^{k-1} \sum_{s=0}^{t-1} \frac{n}{2} \left(1-\gamma\lambda_0\right)^s O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0 \kappa \delta^{3/2}}\right) \left(1 - \frac{\gamma\lambda_0}{4}\right)^{t-s-1} O\left(\sqrt{\frac{n}{\delta}}\right) \\
&\quad = O\left(\frac{\gamma^2 n^{9/2}}{\sqrt{m}\lambda_0 \kappa \delta^2}\right) \sum_{t=1}^{k-1} \sum_{s=0}^{t-1} \left(1-\gamma\lambda_0\right)^s \left(1 - \frac{\gamma\lambda_0}{4}\right)^{t-s-1} \\
&\quad = O\left(\frac{\gamma^2 n^{9/2}}{\sqrt{m}\lambda_0 \kappa \delta^2}\right) \sum_{t=1}^{k-1} \left(1 - \frac{\gamma\lambda_0}{4}\right)^{t-1} \sum_{s=0}^{t-1} \left(\frac{1-\gamma\lambda_0}{1-\gamma\lambda_0/4}\right)^s \\
&\quad = O\left(\frac{\gamma^2 n^{9/2}}{\sqrt{m}\lambda_0 \kappa \delta^2}\right) \sum_{t=1}^{k-1} \left(1 - \frac{\gamma\lambda_0}{4}\right)^{t-1} \frac{4-\gamma\lambda_0}{3\gamma\lambda_0} \left[1 - \left(\frac{1-\gamma\lambda_0}{1-\gamma\lambda_0/4}\right)^t\right] \\
&\quad = O\left(\frac{\gamma^2 n^{9/2}}{\sqrt{m}\lambda_0 \kappa \delta^2}\right) \frac{\left(4-\gamma\lambda_0\right)}{\left(3\gamma\lambda_0\right)\left(1-\gamma\lambda_0/4\right)} \sum_{t=1}^{k-1} \left[\left(1 - \frac{\gamma\lambda_0}{4}\right)^t - \left(1-\gamma\lambda_0\right)^t\right] \\
&\quad = O\left(\frac{\gamma^2 n^{9/2}}{\sqrt{m}\lambda_0 \kappa \delta^2}\right) \frac{4}{3\gamma\lambda_0} \left[\frac{1-\left(1-\gamma\lambda_0/4\right)^k}{\gamma\lambda_0/4} - \frac{1-\left(1-\gamma\lambda_0\right)^k}{\gamma\lambda_0}\right] \\
&\quad \leq O\left(\frac{n^{9/2}}{\sqrt{m}\lambda_0^3 \kappa \delta^2}\right),
\end{aligned}$$

where (a) uses Property 3 and Eq. (4.8), (b) uses Corollary 1, and the rest of the computations use algebraic tricks.

The third term of $\chi(k)$ in Section 4.I can be bounded with high probability using Eq. (4.8) by:

$$\begin{aligned}
\left\| \sum_{t=0}^{k-1} \xi(t) \right\|_2 \leq \sum_{t=0}^{k-1} \|\xi(t)\|_2 &\leq \sum_{t=0}^{k-1} O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0 \kappa \delta^{3/2}}\right) \left\|\mathbf{f}_{\Theta(t)} - \mathbf{y}\right\|_2 \\
&\leq O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0 \kappa \delta^{3/2}}\right) O\left(\sqrt{\frac{n}{\delta}}\right) \sum_{t=0}^{k-1} \left(1 - \frac{\gamma\lambda_0}{4}\right)^t \\
&\leq O\left(\frac{n^{7/2}}{\sqrt{m}\lambda_0^2 \kappa \delta^2}\right),
\end{aligned}$$

where we use Corollary 1.

Summing up, and using Eq. (4.16) to bound the last term of Section 4.I, we showed that with

high probability:

$$\begin{aligned}\left\|\chi(k)\right\|_2 &\leq O\left(\frac{n^{3/2}\kappa}{\sqrt{\delta}\lambda_0}+\frac{n^{9/2}}{\sqrt{m}\lambda_0^3\kappa\delta^2}+\frac{n^{7/2}}{\sqrt{m}\lambda_0^2\kappa\delta^2}+\frac{\sqrt{n}\kappa}{\sqrt{\delta}}\right)\\ &= O\left(\frac{n^{3/2}\kappa}{\sqrt{\delta}\lambda_0}+\frac{n^{9/2}}{\sqrt{m}\lambda_0^3\kappa\delta^2}\right).\end{aligned} \tag{4.17}$$

We now reformulate Section 4.I in terms of the eigenvectors and eigenvalues of the Gram matrix (Eq. (4.3)) as follows, by repeatedly using Property 2

$$\begin{aligned}\mathbf{f}_{\Theta(k)}-\widetilde{\mathbf{y}} &= \gamma\sum_{t=0}^{k-1}\mathbf{H}^{\infty}(\mathbf{I}-\gamma\mathbf{H}^{\infty})^t\mathbf{y}-\widetilde{\mathbf{y}}+\chi(k)\\ &= \gamma\sum_{t=0}^{k-1}\sum_{i=1}^{n}\lambda_i\mathbf{v}_i\mathbf{v}_i^T\sum_{j=1}^{n}\left(1-\gamma\lambda_j\right)^t\mathbf{v}_j\mathbf{v}_j^T\sum_{z=1}^{n}(\mathbf{v}_z^T\mathbf{y})\mathbf{v}_z-\sum_{i=1}^{n}(\mathbf{v}_i^T\widetilde{\mathbf{y}})\mathbf{v}_i+\chi(k)\\ &= \gamma\sum_{t=0}^{k-1}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{z=1}^{n}\lambda_i\left(1-\gamma\lambda_j\right)^t\mathbf{v}_i(\mathbf{v}_i^T\mathbf{v}_j)(\mathbf{v}_j^T\mathbf{v}_z)(\mathbf{v}_z^T\mathbf{y})-\sum_{i=1}^{n}(\mathbf{v}_i^T\widetilde{\mathbf{y}})\mathbf{v}_i+\chi(k)\\ &\overset{(a)}{=} \gamma\sum_{t=0}^{k-1}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{z=1}^{n}\lambda_i\left(1-\gamma\lambda_j\right)^t\mathbf{v}_i\delta_{i,j}\delta_{j,z}(\mathbf{v}_z^T\mathbf{y})-\sum_{i=1}^{n}(\mathbf{v}_i^T\widetilde{\mathbf{y}})\mathbf{v}_i+\chi(k)\\ &= \gamma\sum_{t=0}^{k-1}\sum_{i=1}^{n}\lambda_i\left(1-\gamma\lambda_i\right)^t\mathbf{v}_i(\mathbf{v}_i^T\mathbf{y})-\sum_{i=1}^{n}(\mathbf{v}_i^T\widetilde{\mathbf{y}})\mathbf{v}_i+\chi(k)\\ &= \gamma\sum_{i=1}^{n}\lambda_i\sum_{t=0}^{k-1}\left(1-\gamma\lambda_i\right)^t(\mathbf{v}_i^T\mathbf{y})\mathbf{v}_i-\sum_{i=1}^{n}(\mathbf{v}_i^T\widetilde{\mathbf{y}})\mathbf{v}_i+\chi(k)\\ &= \sum_{i=1}^{n}\left[1-\left(1-\gamma\lambda_i\right)^k\right](\mathbf{v}_i^T\mathbf{y})\mathbf{v}_i-\sum_{i=1}^{n}(\mathbf{v}_i^T\widetilde{\mathbf{y}})\mathbf{v}_i+\chi(k)\\ &= \sum_{i=1}^{n}\left[(\mathbf{v}_i^T\mathbf{y})-\left(1-\gamma\lambda_i\right)^k(\mathbf{v}_i^T\mathbf{y})-(\mathbf{v}_i^T\widetilde{\mathbf{y}})\right]\mathbf{v}_i+\chi(k),\end{aligned} \tag{4.18}$$

where in (a) with some abuse of notation $\delta_{i,j}$ and $\delta_{j,z}$ refer to the Kronecker delta function. This concludes the proof. □

4.J Proof of Theorem 3

Proof. We start similarly to the proof of Lemma 1. Because for $t=k+\tilde{k}-1$ Lemma 3 holds, using Corollary 2 for $t=k+\tilde{k}-1$ and $\mathbf{u}=\widetilde{\mathbf{y}}$, we have:

$$\mathbf{f}_{\Theta(k+\tilde{k})}-\mathbf{f}_{\Theta(k+\tilde{k}-1)} = -\gamma\mathbf{H}^{\infty}\left(\mathbf{f}_{\Theta(k+\tilde{k}-1)}-\widetilde{\mathbf{y}}\right)+\xi(k+\tilde{k}-1),$$

where from Eq. (4.8) with high probability

$$\left\|\xi(k+\tilde{k}-1)\right\|_2 = O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right)\left\|\mathbf{f}_{\Theta(k+\tilde{k}-1)}-\widetilde{\mathbf{y}}\right\|_2.$$

Therefore, recursively we can write:

$$\begin{aligned}
\mathbf{f}_{\Theta(k+\tilde{k})} - \widetilde{\mathbf{y}} &= \mathbf{f}_{\Theta(k+\tilde{k}-1)} - \widetilde{\mathbf{y}} - \gamma \mathbf{H}^{\infty}\left(\mathbf{f}_{\Theta(k+\tilde{k}-1)} - \widetilde{\mathbf{y}}\right) + \xi(k+\tilde{k}-1) \\
&= \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)\left(\mathbf{f}_{\Theta(k+\tilde{k}-1)} - \widetilde{\mathbf{y}}\right) + \xi(k+\tilde{k}-1) \\
&= \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{\tilde{k}}\left(\mathbf{f}_{\Theta(k)} - \widetilde{\mathbf{y}}\right) + \sum_{t=0}^{\tilde{k}-1}\left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{t}\xi(k+\tilde{k}-1-t) \\
&= \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{\tilde{k}}\left[\mathbf{y} - \widetilde{\mathbf{y}} - \sum_{i=1}^{n}\left(1-\gamma\lambda_i\right)^{k}\left(\mathbf{v}_i^{T}\mathbf{y}\right)\mathbf{v}_i + \chi(k)\right] \\
&\quad + \sum_{t=0}^{\tilde{k}-1}\left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{t}\xi(k+\tilde{k}-1-t) \\
&= \left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{\tilde{k}}\left[\mathbf{y} - \widetilde{\mathbf{y}} - \sum_{i=1}^{n}\left(1-\eta\lambda_i\right)^{k}\left(\mathbf{v}_i^{T}\mathbf{y}\right)\mathbf{v}_i\right] \\
&\quad + \left(\mathbf{I} - \eta \mathbf{H}^{\infty}\right)^{\tilde{k}}\chi(k) + \sum_{t=0}^{\tilde{k}-1}\left(\mathbf{I} - \eta \mathbf{H}^{\infty}\right)^{t}\xi\left(k+\tilde{k}-1-t\right), \qquad (4.19)
\end{aligned}$$

where the 4th operation follows from Lemma 1. Now, we find bounds for the perturbation terms. Let

$$\kappa = O(\frac{\epsilon\sqrt{\delta}\lambda_0}{n^{3/2}}), \qquad m = \Omega(\frac{n^9}{\lambda_0^6\epsilon^2\kappa^2\delta^4}). \qquad (4.20)$$

Then, using Lemma 1, the first perturbation term of Eq. (4.19) is upper bounded as

$$\begin{aligned}
\left\|\left(\mathbf{I} - \gamma \mathbf{H}^{\infty}\right)^{\tilde{k}}\chi(k)\right\|_2 &\leq \left(1-\gamma\lambda_0\right)^{\tilde{k}} O\left(\frac{n^{3/2}\kappa}{\sqrt{\delta}\lambda_0} + \frac{n^{9/2}}{\sqrt{m}\lambda_0^3\kappa\delta^2}\right) \\
&= O\left((1-\gamma\lambda_0)^{\tilde{k}}\epsilon\right) \in O(\epsilon),
\end{aligned}$$

where the last line comes from inserting our choices of κ and m from Eq. (4.20).
The last term of Eq. (4.19) can be upper bounded with high probability as

$$\begin{aligned}
&\left\| \sum_{t=0}^{\tilde{k}-1} \left(\mathbf{I}-\gamma\mathbf{H}^{\infty}\right)^t \xi(k+\tilde{k}-1-t) \right\| \leq \sum_{t=0}^{\tilde{k}-1} \left\|\mathbf{I}-\gamma\mathbf{H}^{\infty}\right\|_2^t \left\|\xi(k+\tilde{k}-1-t)\right\|_2 \\
&\overset{(a)}{\leq} \sum_{t=0}^{\tilde{k}-1} \left(1-\gamma\lambda_0\right)^t O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right) \left\|\mathbf{f}_{\Theta(k+\tilde{k}-1-t)}-\widetilde{\mathbf{y}}\right\|_2 \\
&\overset{(b)}{\leq} O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right) \sum_{t=0}^{\tilde{k}-1} \left(1-\gamma\lambda_0\right)^t \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tilde{k}-1-t} \left\|\mathbf{f}_{\Theta(k)}-\widetilde{\mathbf{y}}\right\|_2 \\
&\leq O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right) \sum_{t=0}^{\tilde{k}-1} \left(1-\gamma\lambda_0\right)^t \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tilde{k}-1-t} \left[\left\|\mathbf{f}_{\Theta(k)}-\mathbf{y}\right\|_2 + \left\|\widetilde{\mathbf{y}}-\mathbf{y}\right\|_2\right] \\
&\overset{(c)}{\leq} O\left(\frac{\gamma n^3}{\sqrt{m}\lambda_0\kappa\delta^{3/2}}\right) \sum_{t=0}^{\tilde{k}-1} \left(1-\gamma\lambda_0\right)^t \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tilde{k}-1-t} \left[O\left(\sqrt{\frac{n}{\delta}}\right)\right] \\
&\leq O\left(\frac{\gamma n^{7/2}}{\sqrt{m}\lambda_0\kappa\delta^2}\right) \left(1-\frac{\gamma\lambda_0}{4}\right)^{\tilde{k}-1} \sum_{t=0}^{\tilde{k}-1} \left(\frac{1-\gamma\lambda_0}{1-\gamma\lambda_0/4}\right)^t \\
&\leq \frac{4}{3\gamma\lambda_0} O\left(\frac{\gamma n^{7/2}}{\sqrt{m}\lambda_0\kappa\delta^2}\right) = O\left(\frac{n^{7/2}}{\sqrt{m}\lambda_0^2\kappa\delta^2}\right) \overset{(d)}{=} O\left(\frac{\lambda_0\epsilon}{n}\right),
\end{aligned}$$

where (a) uses Property 3 and Eq. (4.8), (b) uses Corollary 1 and (c) uses Eq. (4.12). Finally (d) follows from inserting our choice of m from Eq. (4.20). Because we have $\sum_{i=1}^n \lambda_i = n/2$ from Property 3, and $\lambda_0 = \min\{\lambda_i\}_i^n$, then $\lambda_0 \leq 1/2$. Both perturbation terms in Section 4.J are therefore at most in the order of ϵ with our choices of m and κ from Eq. (4.20).

Using Property 2, the squared norm of the first term in Eq. (4.19) is

$$\begin{aligned}
&\left\| \sum_{i=1}^{n} \left(1-\gamma\lambda_i\right)^{\tilde{k}} \left[\left(\mathbf{v}_i^T\mathbf{y}\right) - \left(1-\gamma\lambda_i\right)^k \left(\mathbf{v}_i^T\mathbf{y}\right) - \left(\mathbf{v}_i^T\widetilde{\mathbf{y}}\right)\right] \mathbf{v}_i \right\|_2^2 \\
&\quad = \sum_{i=1}^{n}\sum_{j=1}^{n} \left(1-\gamma\lambda_i\right)^{\tilde{k}} \left[\left(\mathbf{v}_i^T\mathbf{y}\right) - \left(1-\gamma\lambda_i\right)^k \left(\mathbf{v}_j^T\mathbf{y}\right) - \left(\mathbf{v}_j^T\widetilde{\mathbf{y}}\right)\right] \\
&\qquad \left(1-\gamma\lambda_j\right)^{\tilde{k}} \left[\left(\mathbf{v}_j^T\mathbf{y}\right) - \left(1-\gamma\lambda_j\right)^k \left(\mathbf{v}_j^T\mathbf{y}\right) - \left(\mathbf{v}_j^T\widetilde{\mathbf{y}}\right)\right] \mathbf{v}_i^T\mathbf{v}_j \\
&\quad = \sum_{i=1}^{n} \left[\left(\mathbf{v}_i^T\mathbf{y}\right) - \left(1-\gamma\lambda_i\right)^k \left(\mathbf{v}_j^T\mathbf{y}\right) - \left(\mathbf{v}_j^T\widetilde{\mathbf{y}}\right)\right]^2 \left(1-\gamma\lambda_i\right)^{2\tilde{k}}
\end{aligned} \tag{4.21}$$

The norm of Eq. (4.19) is therefore, for our choice of κ and m given in Eq. (4.20), with probability at least $1-\delta$

$$\left\|\mathbf{f}_{\Theta(k+\tilde{k})}-\widetilde{\mathbf{y}}\right\|_2 = \sqrt{\sum_{i=1}^{n} \left[\mathbf{v}_i^T\mathbf{y} - \mathbf{v}_i^T\widetilde{\mathbf{y}} - \left(1-\gamma\lambda_i\right)^k \mathbf{v}_i^T\mathbf{y}\right]^2 \left(1-\gamma\lambda_i\right)^{2\tilde{k}}} \pm \epsilon,$$

which concludes the proof. □

4.K Proof and Numerical Evaluations of Theorem 4

Proof. Because $\widetilde{y}_j \sim U(\{-1,1\})$ and $\widetilde{y}_j \perp\!\!\!\perp \widetilde{y}_i$ for $i \neq j$, and $\|\mathbf{v}_i\|_2 = 1$, for $i \in [n]$, we have:

$$\mathbb{E}\left[\tilde{p}_i^2\right] = \mathbb{E}\left[\sum_{j=1}^{n} \mathbf{v}_{i,j}^2 \widetilde{y}_j^2 + \sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n} \mathbf{v}_{i,j}\widetilde{y}_j \mathbf{v}_{i,k}\widetilde{y}_k\right] = \sum_{j=1}^{n} \mathbf{v}_{i,j}^2 + \sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq j}}^{n} \mathbf{v}_{i,j}\mathbf{v}_{i,k}\mathbb{E}\left[\widetilde{y}_j\right]\mathbb{E}\left[\widetilde{y}_k\right] = 1+0 = 1.$$

Recall from Eq. (4.4) that

$$\widetilde{\Phi}(k+\tilde{k}) = \frac{1}{2}\sum_{i=1}^{n}\left[p_i - \tilde{p}_i - p_i\left(1-\gamma\lambda_i\right)^k\right]^2\left(1-\gamma\lambda_i\right)^{2\tilde{k}}.$$

This expression is a random variable that depends on random vectors $\widetilde{\mathbf{p}}$ and $\mathbf{p}$, which are functions of the random label vectors $\widetilde{\mathbf{y}}$ and $\mathbf{y}$, respectively. We now compute the expectation of the above objective function with respect to $\widetilde{\mathbf{p}}$ and $\mathbf{p}$:

$$\begin{aligned}
\mathbb{E}_{\widetilde{\mathbf{p}},\mathbf{p}}\left[\widetilde{\Phi}(k+\tilde{k})\right] &= \frac{1}{2}\sum_{i=1}^{n}\mathbb{E}\left[p_i^2\right]\left[1-\left(1-\gamma\lambda_i\right)^k\right]^2\left(1-\gamma\lambda_i\right)^{2\tilde{k}} \\
&\quad + \frac{1}{2}\sum_{i=1}^{n}\mathbb{E}\left[\tilde{p}_i^2\right]\left(1-\gamma\lambda_i\right)^{2\tilde{k}} - \sum_{i=1}^{n}\mathbb{E}\left[p_i\tilde{p}_i\right]\left[1-\left(1-\gamma\lambda_i\right)^k\right]\left(1-\gamma\lambda_i\right)^{2\tilde{k}} \\
&= \frac{1}{2}\sum_{i=1}^{n}\mathbb{E}\left[p_i^2\right]\left[1-\left(1-\gamma\lambda_i\right)^k\right]^2\left(1-\gamma\lambda_i\right)^{2\tilde{k}} + \frac{1}{2}\sum_{i=1}^{n}\left(1-\gamma\lambda_i\right)^{2\tilde{k}} \\
&\quad - \sum_{i=1}^{n}\left[\sum_{j=1}^{n}\sum_{k=1}^{n}\mathbf{v}_{i,j}\mathbf{v}_{i,k}\mathbb{E}\left[\widetilde{y}_j y_k\right]\right]\left[1-\left(1-\gamma\lambda_i\right)^k\right]\left(1-\gamma\lambda_i\right)^{2\tilde{k}} \\
&= \frac{1}{2}\mu + \frac{1}{2}\sum_{i=1}^{n}\left(1-\gamma\lambda_i\right)^{2\tilde{k}},
\end{aligned} \tag{4.22}$$

where follows with μ given by Eq. (4.5), and because $\widetilde{y}_j \perp\!\!\!\perp y_k$ for all $j,k \in [n]$.

Because of Chebyshev inequality and Eq. (4.22), with probability at least $1-\delta$, we have:

$$\left|\widetilde{\Phi}(k+\tilde{k}) - \frac{1}{2}\sum_{i=1}^{n}\left(1-\gamma\lambda_i\right)^{2\tilde{k}} - \frac{\mu}{2}\right| \leq \sqrt{\frac{\Sigma}{\delta}}, \tag{4.23}$$

where

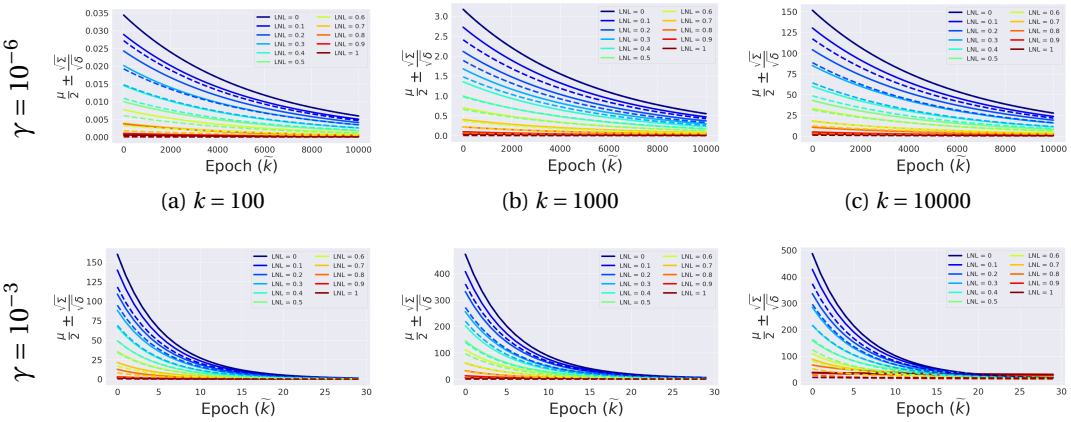
$$\Sigma = \mathrm{Var}_{\widetilde{\mathbf{p}},\mathbf{p}}\left[\widetilde{\Phi}(k+\tilde{k})\right], \tag{4.24}$$

which concludes the proof.

□

Numerical Evaluations We now empirically evaluate the lower and upper bounds in

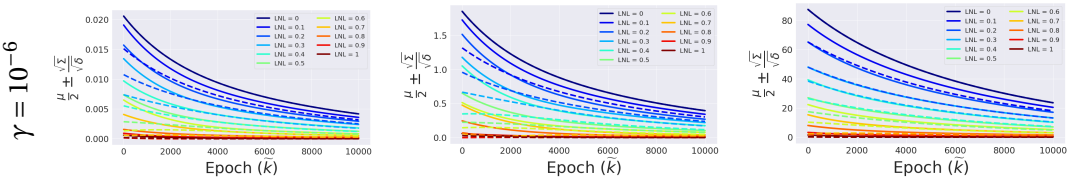
Eq. (4.23) for networks trained on label vector $\mathbf{y}$ with varying label noise levels (LNL). To do so, we discard the middle term of the left hand side of Eq. (4.23), as it does not depend on $\mathbf{y}$. We then study the rest in Fig. 4.34, Fig. 4.35, Fig. 4.36 and Fig. 4.37 for different datasets and values of γ and k. We observe consistently that both the lower and the upper bounds are a decreasing function of the label noise level (LNL) in the label vector $\mathbf{y}$.



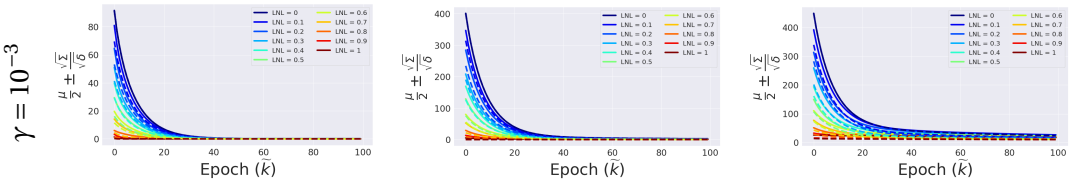
(a) $k = 100$ (b) $k = 1000$ (c) $k = 10000$

(d) $k = 10$ (e) $k = 100$ (f) $k = 1000$

Figure 4.34: The lower (dashed lines) and upper (solid lines) bound terms of Theorem 4 that depend on the label noise level (LNL) are depicted as a function of the number of epochs $\widetilde{k}$, for different hyper-parameter values (the learning rate γ and k) with $\delta = 0.05$. The eigenvector projections p_i that appear in μ and Σ are computed from the Gram-matrix of 1000 samples from the MNIST dataset. The resulting values are obtained from an average over 10 random draws of the label vector $\mathbf{y}$. We observe that both the lower and the upper bounds of Eq. (4.23) are decreasing functions of LNL and of $\widetilde{k}$.

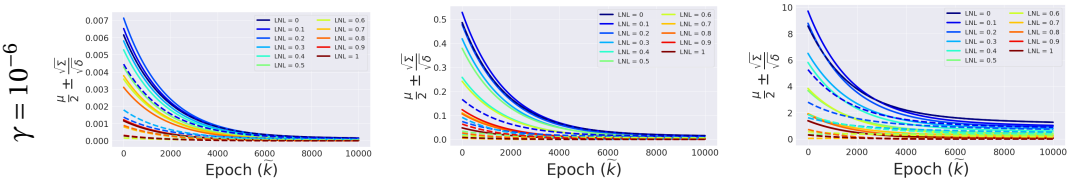


(a) $k = 100$ (b) $k = 1000$ (c) $k = 10000$

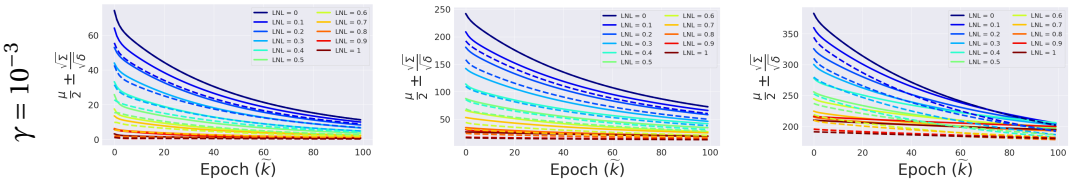


(d) $k = 10$ (e) $k = 100$ (f) $k = 1000$

Figure 4.35: The lower (dashed lines) and upper (solid lines) bound terms of Theorem 4 that depend on the label noise level (LNL) are depicted as a function of the number of epochs $\widetilde{k}$, for different hyper-parameter values (the learning rate γ and k) with $\delta = 0.05$. The eigenvector projections p_i that appear in μ and Σ are computed from the Gram-matrix of 1000 samples from the Fashion-MNIST dataset. The resulting values are obtained from an average over 10 random draws of the label vector $\mathbf{y}$. We observe that both the lower and the upper bounds of Eq. (4.23) are decreasing functions of LNL and of $\widetilde{k}$.

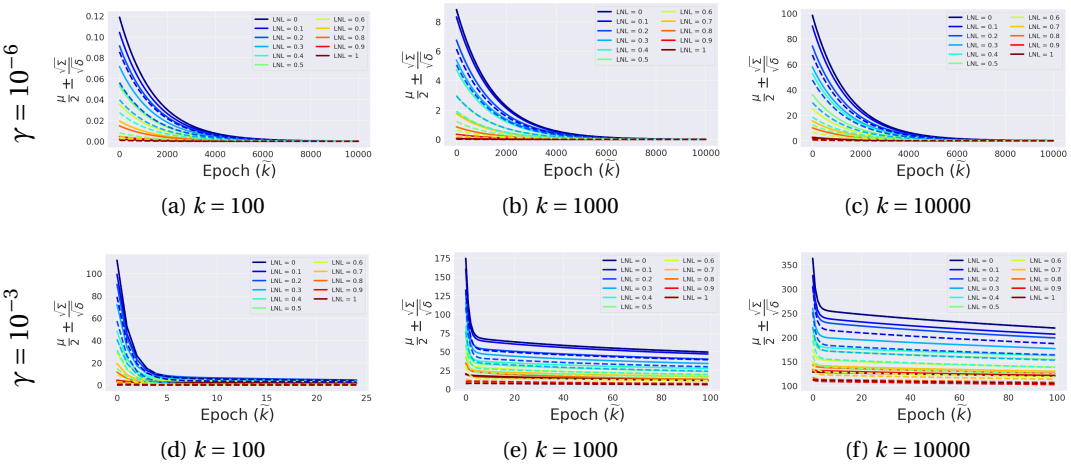


(a) $k = 100$ (b) $k = 1000$ (c) $k = 10000$



(d) $k = 100$ (e) $k = 1000$ (f) $k = 10000$

Figure 4.36: The lower (dashed lines) and upper (solid lines) bound terms of Theorem 4 that depend on the label noise level (LNL) are depicted as a function of the number of epochs $\widetilde{k}$, for different hyper-parameter values (the learning rate γ and k) with $\delta = 0.05$. The eigenvector projections p_i that appear in μ and Σ are computed from the Gram-matrix of 1000 samples from the CIFAR-10 dataset. The resulting values are obtained from an average over 10 random draws of the label vector $\mathbf{y}$. We observe that both the lower and the upper bounds of Eq. (4.23) are decreasing functions of LNL and of $\widetilde{k}$.



(a) $k = 100$ (b) $k = 1000$ (c) $k = 10000$

(d) $k = 100$ (e) $k = 1000$ (f) $k = 10000$

Figure 4.37: The lower (dashed lines) and upper (solid lines) bound terms of Theorem 4 that depend on the label noise level (LNL) are depicted as a function of the number of epochs $\widetilde{k}$, for different hyper-parameter values (the learning rate γ and k) with $\delta = 0.05$. The eigenvector projections p_i that appear in μ and Σ are computed from the Gram-matrix of 1000 samples from the SVHN dataset. The resulting values are obtained from an average over 10 random draws of the label vector $\mathbf{y}$. We observe that both the lower and the upper bounds of Eq. (4.23) are decreasing functions of LNL and of $\widetilde{k}$.

5 Differences Between Hard and Noisy-labeled Samples: An Empirical Study

Extracting noisy or incorrectly labeled samples from a labeled dataset with hard/difficult samples is an important yet under-explored topic. Two general and often independent lines of work exist, one focuses on addressing noisy labels, and another deals with hard samples. However, when both types of data are present, most existing methods treat them equally, which results in a decline in the overall performance of the model. In this chapter, we first design various synthetic datasets with custom hardness and noisiness levels for different samples. Our proposed systematic empirical study enables us to better understand the similarities and more importantly the differences between hard-to-learn samples and incorrectly-labeled samples. These controlled experiments pave the way for the development of methods that distinguish between hard and noisy samples. Through our study, we introduce a simple yet effective metric that filters out noisy-labeled samples while keeping the hard samples. We study various data partitioning methods in presence of label noise and observe that filtering out noisy samples from hard samples with this proposed metric results in the best datasets as evidenced by the high test accuracy achieved after models are trained on the filtered datasets. We demonstrate this for both our created synthetic datasets and for datasets with real-world label noise.

5.1 Introduction

Deep neural networks have revolutionized many applications, especially in the field of image classification, mostly due to the availability of large, high-quality labeled datasets [Rawat and Wang 2017]. In practice, obtaining such datasets is often challenging, time-consuming, and expensive, thus leading to the inclusion of label noise in the obtained datasets [Roh et al. 2019]. Label noise can arise for various reasons such as the use of cheap label collection alternatives, for instance crowdsourcing, or the obtainment of the label of an image from the accompanying text on the Web [Cordeiro and Carneiro 2020; Frénay and Verleysen 2013; Algan and Ulusoy 2020; Karimi et al. 2020]. The problem with label noise is that deep neural networks tend to easily memorize these noisy labels, which can negatively impact their generalization performance [Zhang et al. 2021a]. Many methods propose to mitigate the effects of label noise,

including the use of robust loss functions [Ma et al. 2020; Thulasidasan et al. 2019; Wang et al. 2019; Patrini et al. 2017] and modifications to the training procedures [Yu et al. 2019; Zhang et al. 2020b; Jiang et al. 2018a; Malach and Shalev-Shwartz 2017]. Another popular approach to deal with label noise is to use a noisy-label detection method [Nguyen et al. 2019; Li et al. 2020a; Huang et al. 2019; Pleiss et al. 2020b;a]. This approach can involve data cleansing, where the noisy data is entirely removed from the data set altogether, data re-weighting, where noisy data are given lower weights during training, or re-labeling, where the noisy data is re-annotated by experts.

When using noisy-label detection methods, an issue that often arises is their inability to differentiate between noisy-labeled samples and hard-to-learn samples. Hard-to-learn samples, also simply known as hard samples, refer to samples in a dataset that are particularly challenging for the classifier to learn [Arpit et al. 2017; Kishida and Nakayama 2019; Wu et al. 2018]. Empirical observations have revealed that noisy samples and hard samples share certain characteristics, such as high loss or low confidence or being learned later in the training process. Consequently, when noisy label detection methods are employed, which are based on these characteristics, hard samples are also treated as noisy samples and are either filtered out or given lower emphasis during training. Unfortunately, discarding hard samples may result in gaps in the classifier's knowledge of the true decision boundary, making it crucial to preserve as many hard samples as possible and prioritize their leaning [Chang et al. 2017; Bengio et al. 2009; Wang et al. 2020]. It is thus vital to propose noisy-label detection methods capable of distinguishing between noisy-labeled samples and hard samples, keeping as many hard samples as possible while removing noisy ones.

Recent studies that propose noisy-label detection methods which are claimed to retain hard samples lack a quantitative evaluation of such claim, as there is no precise measure to quantify sample difficulty [Liang et al. 2022; Bai and Liu 2021; Zhu et al. 2021a; Zhang et al. 2022]. To overcome this limitation, in this work, we propose synthetic datasets that simulate varying levels of hardness and noisiness. Although synthetic label noise has been previously studied [Zhang et al. 2021a; Rolnick et al. 2017], our work, to the best of our knowledge, is the first to introduce synthetic hardness/difficulty levels and to assign each sample with a custom hardness level.

There are various reasons that a sample is difficult to classify: it can be under-represented in the data set, or it can have distinct characteristics from other samples in its class, or it can be near the decision boundary. We propose three main approaches to artificially, by applying transformations to the original dataset samples, produce hard samples, we refer to these three approaches as *hardness type*. We have: (i) imbalanced-ness, (ii) diversification, and (iii) closeness to the decision boundary. The first approach (i) introduces sample difficulty by creating an imbalanced dataset, and subsampling different classes with varying cardinality. This results in under-represented classes being more difficult to learn. In the second approach (ii), the difficulty is introduced by making different classes more or less distinct in their samples, through custom data augmentations. This is achieved by applying a varying number of data

augmentations to different classes, thus resulting in classes with fewer distinct samples and with more augmented samples per distinct sample being easier to learn. In the third approach (iii), the difficulty is introduced by modifying the input samples to be closer to the decision boundary. The decision boundary is estimated using a pre-trained model with a high test accuracy, and the samples are modified to be closer to this estimated decision boundary. Overall, we do not claim these approaches cover all contexts in which hard samples arise, and in practice, a sample might be difficult because of any combination of the aforementioned reasons, or some other reason. Yet, these common-sense approaches of introducing sample difficulty enable us to perform controlled experiments to assess different metrics and methods in terms of their ability to distinguish between hard and noisy samples.

In this chapter, our key observation is that the feature embeddings of hard samples become closer to each other during training, whereas noisy-labeled samples do not necessarily exhibit this behavior because of the visual dissimilarity between samples in the same class. This observation leads us to propose the distance between the feature layer vector of each sample and the centroid feature vector of its assigned class, as a metric to distinguish between hard and noisy-labeled samples, which we call static-centroid distance (SCD). Next, we propose a label noise detection method based on SCD. While other methods perform well in only one of the two tasks among filtering out noisy samples and retaining hard samples, our method is the only one that performs well in *both* tasks, and consistently so in all of our synthetic datasets as well as datasets with real-world label noise. We demonstrate the superior performance of this approach for noisy label detection when used for data cleansing. Overall, our key contributions are as follows:

- We propose a novel approach for creating synthetically difficult samples using three different approaches: imbalanced-ness, diversification, and closeness to the decision boundary. To the best of our knowledge, we are the first to use controlled experiments to simulate hard samples and assess how different label noise detection methods perform in terms of retaining hard samples.
- We study various metrics for detecting noisy labels and show that static centroid distance (SCD) is the most effective metric in distinguishing between hard and noisy-labeled samples. While other metrics remain monotonic by increasing either as a function of hardness or noisiness, we show that SCD is the only metric that is increasing with noisiness but is *not* increasing with hardness.
- We propose and evaluate different methods for data cleansing and sample selection and we show that a two-dimensional Gaussian mixture model, which uses (i) the accuracy over the training and (ii) SCD as features, performs the best in terms of filtering out noisy samples while retaining hard ones, on the synthetic datasets.
- We empirically show that our method produces the best generalization performance when models are trained on the filtered datasets. This holds both in the synthetic datasets and even better in datasets with real-world label noise.

5.2 Background and Related Work

In this section, we first introduce the problem setup developed in this work to produce datasets with custom noisiness and hardness levels. Then, we recall some metrics from previous works that are relevant to our study. Next, we introduce static centroid distance (SCD). Finally, we present various partitioning approaches that can be used in conjunction with each metric for sample selection.

Consider a classification task with input $x \in \mathcal{X}$ and ground truth one-hot label vector $y \in \{0,1\}^K$, where K is the number of classes. The original training set, with ground-truth labels, is denoted by $S_{\text{org}} = \{(x_i, y_i)\}_{i=1,\cdots,N_{\text{org}}}$, which consists of N_{org} input-output pairs. We assume that the available given training set S is a transformation of S_{org}. The available training set S consists of N[1] training sample pairs $S = \mathcal{T}(S_{\text{org}}) = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1,\cdots,N}$, where the transformation $\mathcal{T}$ applied to S_{org} is such that S might have label noise and might contain samples which are more difficult for a classifier to learn compared to the rest.

We partition the training set $S = S_n \cup S_e \cup S_h$, where S_n are the incorrectly-labeled samples (or noisy samples), S_e are the correctly-labeled and easy-to-learn samples, and S_h are the correctly-labeled and hard-to-learn samples. In practical applications, it is often challenging to clearly distinguish between easy and hard samples in a given dataset. This can be a limitation when studying the performance of different models or algorithms, as the lack of a clear differentiation between easy and hard samples can result in inaccurate estimates of the sets S_h and S_e. To address this issue, we synthetically, from some original available dataset S_{org}, create a spectrum of samples that have different hardness levels $h \in \{0,1,2,3,4\}$ and noisiness levels $n \in \{0,1,2,3,4\}$: samples are harder to learn as h increases, and are noisier as n increases. We will now discuss how we transform an original dataset $S_{\text{org}} = \{(x_i, y_i)\}_{i=1,\cdots,N_{\text{org}}}$, with no label noise and with uniform hardness levels among its samples, to a dataset $S = \mathcal{T}(S_{\text{org}})$. This transformation provides us with a knob that we can tune to make alterations, allowing for a systematic study of easy, hard, and noisy samples while offering a good comparison base.

Noisiness transformation Let $S' = \{(x_i, y_i)\}_{i=1,\cdots,N'} \subset S_{\text{org}}$, be a subset of size N' from the original dataset S_{org}. We want to transform S' such that the resulted set $S'_t = \mathcal{T}_n(S')$ has samples with noisiness level $n \in \{0,1,2,3,4\}$. The noisiness level n determines the label noise level $q(n)$. The transformation $\mathcal{T}_n$ is such that $S'_t = \{(x_i, \tilde{y}_i)\}_{i=1,\cdots,N'}$, where with probability $1-q(n)$, $\tilde{y}_i = y_i$, and with probability $q(n)$, the non-zero element of $\tilde{y}_i$ is set (uniformly at random) at index $j \sim U(\{1,2,\cdots,k\})$. Such dataset transformation is a common practice to study label noise in a controlled setting, which is done by fixing some label noise level q for the entire transformed dataset.

Hardness transformation Let $S' = \{(x_i, y_i)\}_{i=1,\cdots,N'} \subset S_{\text{org}}$, be a subset of size N' from the original dataset S_{org}. We want to transform S' such that the resulted set $S'_t = \mathcal{T}_h(S')$ has

[1] Note that only in this chapter the training set size is denoted by N, whereas in the rest of the thesis it is denoted by n. In this chapter, with some abuse of notation compared to the rest of the thesis, we use n for the noisiness level.

samples with hardness level $h \in \{0,1,2,3,4\}$. In this chapter, we consider this transformation to be a composition of two transformations, i.e., $\mathscr{T}_h = \mathscr{F}_h \circ \mathscr{I}_h$. The first transformation is one-to-one, and maps S' into $S'' = \mathscr{I}_h(S') = \{(x_j, y_j)\}_{j \in J_h}$, where $J_h \subset \{1, \cdots, N'\}$. The second transformation is one-to-many, and maps S'' into $S'_t = \mathscr{F}_h(S'') = \{(\tilde{x}_{I(j)}, y_j)\}_{j \in J_h}$, where $\tilde{x}_{I(j)}$ is a transformed version of input sample x_j, and $I(j)$ can be a one-to-many function. In Section 5.3, we discuss how the transformation $\mathscr{T}_h$ is done.

The final transformed dataset is $S = \cup_{S'_{h,n}} \mathscr{T}_n \circ \mathscr{T}_h(S'_{h,n})$, where the subsets $S'_{h,n} \subset S_{\text{org}} \forall h \in \{0,1,2,3,4\}, n \in \{0,1,2,3,4\}$ are determined according to a pre-defined policy. With $S'_{t,h,n} = \mathscr{T}_n \circ \mathscr{T}_h(S'_{h,n})$, the three sets partitioning S are

$$
\begin{aligned}
S_h &= \{(\tilde{x}_i, \tilde{y}_i) \in S'_{t,h,n} | \tilde{y}_i = y_i\}, \\
S_n &= \{(\tilde{x}_i, \tilde{y}_i) \in S | \tilde{y}_i \neq y_i\}, \\
S_e &= S \setminus (S_h \cup S_n).
\end{aligned}
$$

We consider the classifier trained on the dataset S to be a neural network. For each input sample x_i, let $\mathbf{p}(x_i) = (p_i^1, p_i^2, \cdots, p_i^k)$ be the prediction probability output vector of the neural network. Furthermore, the last layer of the neural network is a fully-connected layer with feature vector $\mathbf{h}(x_i) \in \mathbb{R}^m$, where m is the number of units in the feature layer of the neural network. Note that because the parameters of the neural network depend on the epoch during training, both vectors $\mathbf{p}(x_i)$ and $\mathbf{f}(x_i)$ depend on the epoch $t \in \{1,2,\cdots,k\}$, where k is the maximum number of training epochs. We often remove the dependence on t for simplicity. Training is done by performing stochastic gradient descent (SGD) on the cross-entropy training loss function at each epoch t

$$
L_S(t) = \frac{1}{N} \sum_{i=1}^{N} l_i(t) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_i^j \log p_i^j(t) = -\frac{1}{N} \sum_{i=1}^{N} \log p_i^{c_i}(t)
$$

where $l_i(t)$ is the training cross-entropy loss of sample x_i at epoch $t \in \{1,2,\cdots,k\}$. The prediction of the neural network classifier at epoch t for sample x_i is class $\tilde{c}_i = \arg\max_j p_i^j(t)$, and the confidence of the classifier is $p_i^{\tilde{c}_i}$ that is the prediction probability of the classifier for the predicted class label.

Metrics Here, we recall a few metrics that are introduced in prior work and introduce a new metric, static centroid distance (SCD). In the following sections, we perform a comprehensive study on all these metrics in order to detect which metric, or combination of which metrics, is the best at distinguishing between samples in S_n and S_h. In order to remain on a computationally limited budget, we study metrics that require only a single model and not an ensemble of models. For each sample x_i, we compute the following metrics defined below. Throughout our study, we primarily focus on loss, confidence, and SCD metrics. The former two are widely used in the literature due to the valuable information they provide about each sample, while the latter is a metric proposed in our work.

- *Loss*: The training loss $l_i(k)$ at the end of the training, i.e., at epoch k.
- *Confidence*: The prediction probability $p_i^{\tilde{c}_i}(k)$ at the end of the training, i.e., at epoch k.
- First Prediction Epoch: The epoch t^* such that $\tilde{c}_i(t^*) = c_i(t^*)$ and $\tilde{c}_i(s) \neq c_i(s)$ for $s < t^*$.
- Accuracy of Predictions over Training: The accuracy of the classifier predictions for x_i against the assigned class label c_i over the training process, i.e., $\frac{1}{k}\sum_{t=1}^{k} \mathbb{1}(c_i(t) = \tilde{c}_i(t))$, where $\mathbb{1}$ is the indicator function. This is used by Sun et al. [2020] to detect noisy samples and to remove them from the dataset.
- AUL: Area under Loss: $\sum_{t=1}^{k} l_i(t)$, which is used by Pleiss et al. [2020b] to detect noisy samples.
- AUM: Area under Margin: $\frac{1}{k}\sum_{t=1}^{k} p_i^{\tilde{c}_i}(t) - p_i^{c_i'}(t)$, where $p_i^{c_i'}(t)$ is the largest second logit at time t. This is used by Pleiss et al. [2020a] to detect noisy samples.
- JSD: Jensen-Shannon divergence between $\mathbf{p}_i$ and $\tilde{y}_i$ at the end of the training, i.e., at epoch k. Its weighted version, WJSD, is proposed by Zhang et al. [2022] to detect noisy samples.
- ACD: Adaptive Centroid Distance, which is the cosine distance between the feature vector $\mathbf{h}(x_i)$ and the *adaptive* centroid of feature vectors $\mathbf{o}_{c_i}$ at the end of the training, i.e., at epoch k. The term *adaptive* refers to the computation of $\mathbf{o}_{c_i}$ as it is the centroid for samples that the classifier suspects to be in class c_i: either these samples have been assigned to another class but the classifier predicts them to be in class c_i, or they have been assigned to class c_i and the classifier has high $p_i^{c_i}$. This metric is proposed by Zhang et al. [2022] to detect noisy samples.
- SCD (in our work): Static Centroid Distance, which is the Euclidean distance between $\mathbf{h}(x_i)$ and the *static* centroid of feature vectors $\mathbf{o}_{c_i}^s$ at the middle of training, k^*, where $ACC(k^*) \geq 50\%$ and $ACC(s) < 50\%$ for $s < k^*$. This is developed in our work as opposed to ACD, as a result of our empirical observations in studying hard samples. The adaptations from ACD to compute SCD are the following: (1) the distance function in SCD is Euclidean distance instead of cosine distance as we are interested in different regions of the feature space and not in different angles/directions of the feature vectors; (2) SCD is computed in the middle of training, where the training accuracy is 50%. This eliminates the need to proceed with training until the very end, when the purpose of training is only to detect noisy samples. However, we observe that in this intermediate stage of training the differences between hard and noisy samples are at their highest; (3) the centroid vector $\mathbf{o}_{c_i}^s$ is computed for *all* samples that are assigned to class c_i, and not only for those samples which the classifier predicts to be in class c_i. Because for hard classes, the classifier, which might not have learned them correctly, does not predict samples that actually belong to class c_i, the distance to $\mathbf{o}_{c_i}$ becomes rather meaningless. We observe that relying too much on the classifier to compute this centroid, which is

done in the computation of ACD, results in inferior performance, especially when the quality of the data is low.

Partitioning Methods Using the above metrics, one can then cluster the available samples in S in order to find estimates for the easy, hard and noisy subsets, which we refer to by $\tilde{S}_e$, $\tilde{S}_h$ and $\tilde{S}_n$, respectively. Note that our overall goal is to find the estimated noisy subset $\tilde{S}_n$ and the estimated clean $\tilde{S}_c = \tilde{S}_e \cup \tilde{S}_h$, but not precisely the two subsets $\tilde{S}_e$ and $\tilde{S}_h$.

- Thresholding (*Thres* in short) - In this method, a threshold value is chosen for the specific metric, which can be the average value over the samples or any other predetermined value. The samples are then partitioned into two regions, one containing samples whose values on the metric are greater than the threshold, and the other one containing the samples whose values on the metric are smaller than the threshold Because we work with values in the dataset that might have outliers (in particular for hard samples and noisy samples), we choose the median value as the default threshold when using this method.
- 1d-GMM - A one-dimensional Gaussian mixture model is used for partitioning the dataset into multiple subsets or partitions by modeling the values of the metric for each sample as a mixture of several Gaussian distributions, and each Gaussian distribution represents one of the partitions.
- 2d-GMM - Similarly to 1d-GMM, a two-dimensional GMM can partition the dataset by using two metrics instead of only one.

5.3 Dataset Design with Different Hardness Levels

We introduce three new datasets, which are transformations of a real-world dataset, that are created to assess the robustness of machine-learning models against label noise and sample difficulty. These datasets are transformed from the TinyImagenet dataset [Le and Yang 2015], with each dataset having 200 classes, i.e., S_{org} is the TinyImagenet training set. To ensure consistency within each class, we systematically adjusted the hardness h and noisiness n levels across five levels, such that all samples of a given class hold the same h and n. To provide a comprehensive spectrum of samples, we varied h and n along two orthogonal axes, resulting in a 2D space with four quadrants (and a total of 25 different sample categories): easy-clean ($h = 0$ and $n = 0$), easy-noisy ($h = 0$ and $n = 4$), hard-clean ($h = 4$ and $n = 0$), and hard-noisy ($h = 4$ and $n = 4$). Each sample in S_{org} has a pair of (h, n) according to Fig. 5.1. The set of samples with (h, n) pair is denoted by $S'_{h,n}$.

Denote the set of samples with hardness level h by S'_h with size N'_h. These samples undergo a transformation $\mathscr{T}_h = \mathscr{F}_h \circ \mathscr{I}_h$ to become more or less hard to learn depending on the value of h. We refer to the hardest samples, which are samples with $h = 4$, in the transformed

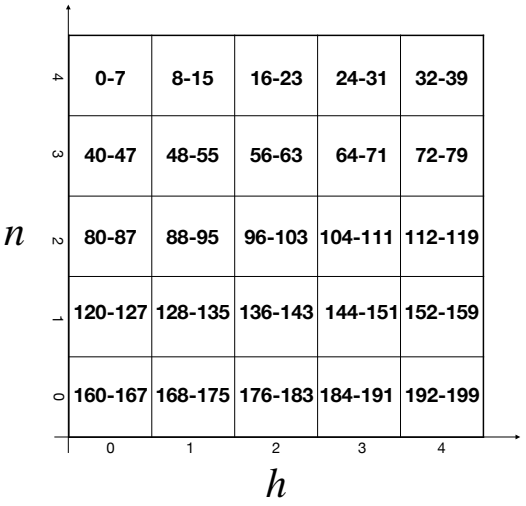
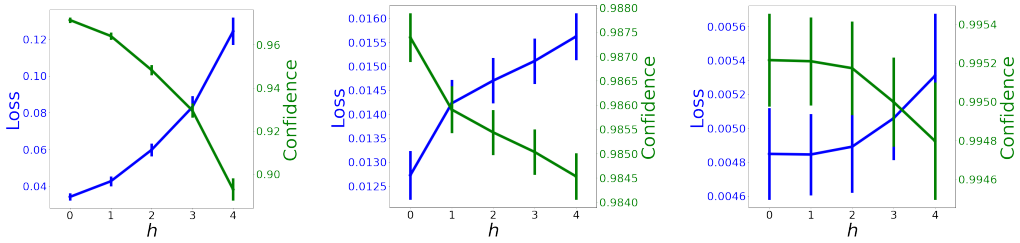


Figure 5.1: The hardness level h and noisiness level n of samples in the datasets are determined by their class, as shown on the 2-dimensional plot. For each pair (h,n), we allocate class number $c = 40*(4-n)+8*h+\beta$ where $\beta \in \{0,1,\cdots,7\}$, so that each pair (h,n) is represented by 8 different classes. For example, samples with $h = 0$ and $n = 4$ are in classes 0 to 7. We first adjust the difficulty level h of each sample, based on its class-specific hardness, by using the original labels from the TinyImagenet dataset. We then introduce label noise and assign new labels to each sample, based on the level of noise n that is determined by the original class label. This process results in a comprehensive dataset with samples that range in difficulty from easy to hard, and with labels that can be either correct or noisy.

dataset as the hard subset S_h. In Sections 5.3.1, 5.3.2, and 5.3.3, we provide details of the transformation $\mathcal{T}_h$ in each of the three hardness types. We then demonstrate the effectiveness of this sample creation process in terms of making samples more or less difficult. This is achieved by using metrics that are widely accepted from the literature and that quantify sample difficulty/hardness, such as loss [Kishida and Nakayama 2019; Loshchilov and Hutter 2015], and confidence [Swayamdipta et al. 2020; Chang et al. 2017; Wang et al. 2020]. Prior studies show that hard samples have larger losses and lower confidence. By examining the correlation between loss/confidence and the hardness levels assigned by our approach, we find that our assignments are accurate, as shown in Fig. 5.2 (using a one-way ANOVA test). Finally, in the last part of this section, we describe how we create samples with different values noise levels n. We then observe that the same correlation sign between loss/confidence and hardness also exists between loss/confidence and noisiness. This indicates that hardness and noisiness exhibit similar characteristics, and that removing noisy samples from the training dataset by using loss or confidence could also lead to the inadvertent removal of correctly-labeled hard-to-learn samples.

5.3.1 Hardness via Imbalance

In this dataset, we make samples with different h values, by making the dataset imbalanced. To achieve this, we subsample the dataset such that the number of samples for each class with hardness h is $X/2^h$, where X is the maximum number of samples per class in the dataset. With this approach, the number of samples per class decreases exponentially as h increases,



(a) Hardness via Imbalance (b) Hardness via Diversification (c) Hardness via Closeness to the Decision Boundary

Figure 5.2: In each of the three created datasets, we observe that samples in classes with higher h have generally traits that correspond to their difficulty. These traits include having a higher loss value and having a lower confidence (or prediction probability). To further show the significance of loss and confidence values as h increases, we perform a one-way ANOVA test, which results in the following F-statistics and p-values: **(a) Hardness via Imbalance:** for loss we have F=672 and p$< 1e-100$, for confidence we have F=1519 and p$= 0$; **(b) Hardness via Diversification:** for loss we have F=234 and p$< 1e-190$, for confidence we have F=237 and p$< 1e-200$; **(c) Hardness via Closeness to the Decision Boundary:** For loss we have F=5.7 and p$< 1e-3$; for confidence we have F=5.8 and p$< 1e-3$. We can observe that in all three settings, the F-statistics are relatively large and the p-values are small. We conclude that our assigned h values do indeed indicate sample difficulty and in all three settings our approaches for creating hard samples is justified.

which makes these samples under-represented hence more difficult to learn. We have

$$S'_t = \mathscr{T}_h(S'_h) = \mathscr{I}_h(S'_h) = \{(x_j, y_j)\}_{j \in J_h}$$

where $J_h \sim \{1, \cdots, N'_h\}$ is a subset of size $X/2^h$. In this hardness type the transformation $\mathscr{F}_h$ is the identity map and $\mathscr{T}_h = \mathscr{I}_h$.

Are samples in $S_{h=4}$ with this hardness type actually more difficult to learn? As we can see in Fig. 5.2a, samples that are assigned with a larger hardness h value have indeed a higher loss and lower confidence values. We conclude that samples with $h = 4$, which are those in S_h, are more difficult for the classifier to learn than the rest of the samples.

5.3.2 Hardness via Diversification

Compare the following two set of samples: (i) 16 images of different cats possibly with different breads, (ii) 16 images of the same cat with different image augmentations, e.g., rotation, flipping, etc. Which of the two set of samples is more difficult for a classifier to learn? We can argue that the first set is more difficult, because the classifier requires to learn some common features that belong to all 16 different images. The first set of samples are more *diverse* and hence harder. Such diversification is the second hardness type we consider.

In this dataset, we keep the number of samples balanced between classes, but we vary the

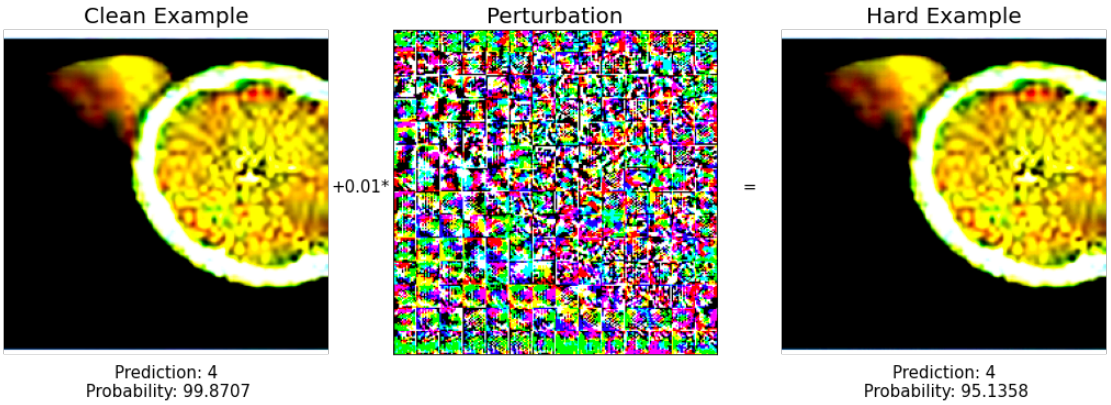


Figure 5.3: This figure illustrates an example of how a sample can be made more difficult by moving it closer to the decision boundary of the estimated ground-truth model, using a similar approach as in fast gradient sign method[Goodfellow et al. 2014]. In this case, the epsilon value ϵ is set as 0.01.

diversity of samples per class. To do so, we maintain the original samples in hard classes in order to preserve their level of diversity/difficulty. For samples in the easy classes, we perform data augmentation techniques, such as rotation or flipping, to make samples less diverse and hence easier to learn. This approach, by changing the number of data augmentations used in each level of hardness h, enables us to generate a range of hardness levels within a single dataset. In particular, for hardness level h, we first subsample the number of samples per class by $X/2^{4-h}$, where X is the maximum number of unique samples per class. Then, we create $2^{4-h}-1$ new augmented samples per sample. As a result, all classes have the same overall number of samples and the dataset is balanced. We have

$$\begin{aligned} S'_t = \mathcal{T}_h(S'_h) &= \mathcal{F}_h \circ \mathcal{I}_h(S'_h) \\ &= \mathcal{F}_h\left(\{(x_j, y_j)\}_{j\in J_h}\right) \\ &= \{(\tilde{x}_{I_h(j)}, y_j)\}_{j\in J_h}, \end{aligned}$$

where $J_h \sim \{1,\cdots,N'_h\}$ is a subset of size $X/2^{4-h}$, $I_h(j) = \{j + a\cdot|J_h|\}_{a\in\{0,\cdots,2^{4-h}-1\}}$, and for $I_h(j) \neq j$, $\tilde{x}_{I_h(j)}$ is an augmented version of x_j.

Are samples in $S_{h=4}$ with this hardness type actually more difficult to learn? As we can see in Fig. 5.2b, samples that are assigned with a larger hardness h value, have higher loss and lower confidence values. Hence, we conclude that samples with $h=4$, which are in S_h, are the most difficult for the classifier to learn.

5.3.3 Hardness via Closeness to the Decision Boundary

In this dataset, we produce hard samples by modifying them to be closer to the decision boundary. To achieve this, we must first identify the true decision boundary for the classi-

fication task at hand. As we are working with the TinyImagenet dataset, the true decision boundary is unknown. The closest estimate to the true decision boundary that we found is a DeiT model [Touvron et al. 2021] that was pre-trained on ImageNet and fine-tuned on TinyImagenet, with a training accuracy of 98.8% and a test accuracy of 90.88% on the original train and test sets of TinyImagenet. Then, we create samples that are closer to the decision boundary of this model, hoping that they will also be closer to the decision boundary of the actual ground truth model.

We have

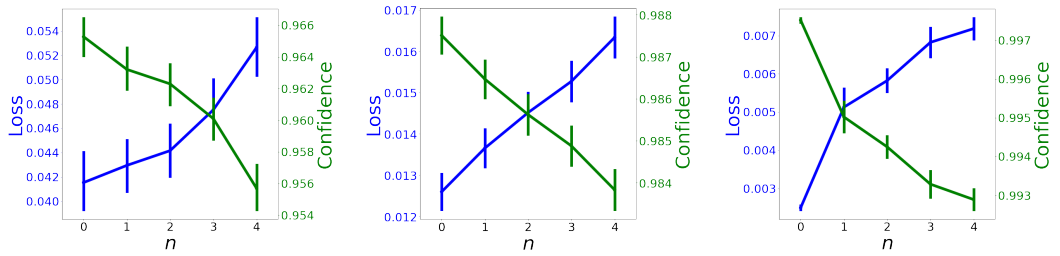
$$S'_t = \mathscr{T}_h(S'_h) = \mathscr{F}(S'_h) = \{(\tilde{x}_j, y_j)\}_{j \in J_h},$$

where $J_h = \{1, \cdots, N'_h\}$, and $\tilde{x}_j$ is a transformation of the sample x_j, such that $\tilde{x}_j$ is closer to the decision boundary compared to x_j. In this hardness type the transformation $\mathscr{I}_h$ is an identical map and $\mathscr{T}_h = \mathscr{F}_h$. Our hard sample creation is done similarly to the fast gradient sign method for creating adversarial samples, which is introduced by Goodfellow et al. [2014]. This method, for an input sample x_i, creates adversarial input

$$x_i^{\text{adv}} = x_i + \epsilon \text{sign}\,(\nabla_x l_i)$$

where ϵ is a hyper-parameter used for scaling the added noise. For creating adversarial samples, ϵ should large enough to change the label of the sample. For our dataset creation however, unlike with adversarial sample creation, we make sure that the resulting sample $\tilde{x}_i$ does *not* change its label, by adjusting ϵ (see an example in Fig. 5.3). Different levels of h are obtained by adjusting the value of ϵ. Higher values of ϵ move the sample closer to the decision boundary, making it harder to learn with a new model. After generating new samples, we keep only those that maintain their original class label, as our goal is to increase the difficulty of samples in their original class. It is critical to choose the appropriate range of ϵ, as a value that is too high can cause most samples to change their predicted class and be excluded from the data set. This would result in the remaining samples being easier to learn instead of harder, because they were originally very easy and far from the decision boundary. Therefore, we carefully choose the range of ϵ during our dataset creation, which results in a smaller range of the hardness levels, as seen in Fig. 5.2c. These samples are challenging particularly for models that make decisions near the decision boundary, which is why in this setting, unlike the other settings, we use models which are pre-trained on the ImageNet dataset instead of using random parameter initialization.

Are samples in $S_{h=4}$ with with hardness type actually more difficult to learn? As we can see in Fig. 5.2c, samples that are assigned with a larger hardness h value, have higher loss and lower confidence values. The loss values vary however less with the hardness level h than in the two previous settings. This is due to the sensitivity of the range of ϵ that must be large enough to bring the sample close to the boundary, but small enough to keep it from changing its label. But Fig. 5.2c still concludes that samples with $h = 4$, which are in S_h, are more difficult for the classifier to learn compared to the rest of the samples.



(a) Hardness via Imbalance

(b) Hardness via Diversification

(c) Hardness via Closeness to the Decision Boundary

Figure 5.4: We observe that, similarly to the effect of hardness increase discussed in Section 5.3.1-Section 5.3.3, an increase in the level of noisiness increases loss and decreases confidence. Therefore, if we were to rely solely on these metrics to identify and remove noisy-labeled samples, hard samples can also be mistakenly removed from the dataset. This is problematic because hard samples contain valuable information about the underlying data distribution and should not be ignored. We can observe the created harder sample has still the same prediction class but with a lower confidence, which suggests that it is closer to the decision boundary.

5.3.4 Addition of Label Noise; Its Similarities to Hardness

After creating each of the hard datasets, we then add label noise to samples of the dataset according to Fig. 5.1. For each sample with noisiness n, we assign the label noise level of $\delta * n/4$, where $\delta \in [0,1]$ is the maximum label noise level of the created dataset (by default we use $\delta = 0.4$ in our experiments). Label-noise level refers to the probability of the label being replaced uniformly at random with one of the class labels. For example, samples with $n = 1$, have 0.25δ probability of having a noisy label. To prevent confusion between samples in classes with different levels of noisiness n, we limit the choice of class labels to only those with the same n when assigning noisy labels. This ensures that samples are assigned to a class with $n = 0$ if and only if they actually belong to that class. Overall, for a set of samples with noisiness level n, denoted by S'_n, we have

$$S'_t = \mathcal{T}_n(S'_n) = \{(x_i, \tilde{y}_i)\}_{i \in \{1,\cdots,N'_h\}},$$

where with probability $1 - q(n)$, $\tilde{y}_i = y_i$, and with probability $q(n)$, the non-zero element of $\tilde{y}_i$ is at index $j \sim U(1,2,\cdots,k)$, where U is the uniform distribution, for $q(n) = \delta * n/4$.
In Fig. 5.4, we observe that increasing n, similar to increasing h discussed in previous sections, results in a higher loss, and a lower confidence. The similar behavior of the loss and confidence on as function of h and n hints on the inefficiency of these metrics to distinguish between hard samples and noisy labels. If these metrics were used to remove incorrectly labeled samples, hard samples would also be eliminated, which is undesirable. This highlights the need for more advanced metrics/methods to identify and preserve hard samples while removing noisy-labeled ones.

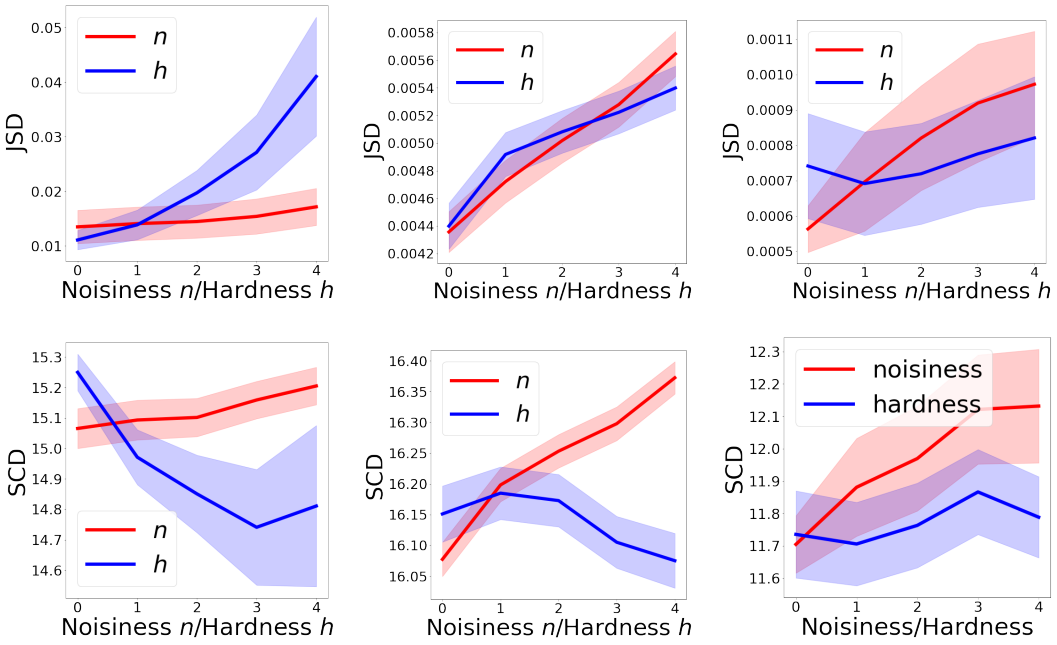
5.4 Easy-Hard-Noisy Data Partitioning and Training

In this section, we use the metrics and methods discussed in Section 5.2 to partition the synthetic datasets constructed in Section 5.3 into two subsets: the estimated clean subset $\tilde{S}_c$ and the estimated noisy subset $\tilde{S}_n$. We evaluate the effectiveness of these methods and metrics in this partitioning task, with a focus on the quality of the filtering between hard and noisy samples. We then compare the performance of different partitioning methods based on the test accuracy of models trained on the estimated clean subset produced from each method. It is important to note that although the training datasets can contain samples with issues, such as hardness or noisiness in certain classes, the test datasets do not contain any such noisy-labeled or imbalanced samples. Therefore, traditional evaluation metrics, such as test accuracy, provide a fair representation of the models' generalization performance in each setting.

5.4.1 Partitioning

We compare different methods that partition the available dataset into two subsets: $\tilde{S}_c$ and $\tilde{S}_n$. Our primary objective is to include the incorrectly labeled samples within $\tilde{S}_n$, which is the focus of previous studies as well, but in addition, we are also interested in including all hard samples within $\tilde{S}_c$. Therefore, we aim to achieve high recall for hard samples, i.e., $\text{Recall}_\text{h} = \frac{|\tilde{S}_c \cap S_h|}{|S_h|}$, while maintaining a high recall for noisy samples, i.e., $\text{Recall}_\text{n} = \frac{|\tilde{S}_n \cap S_n|}{|S_n|}$. Although having a high precision for noisy samples, i.e., $\text{Precision}_\text{n} = \frac{|\tilde{S}_n \cap S_n|}{|\tilde{S}_n|}$, is desirable, it is less critical. Even if some easy samples are mistakenly included in $\tilde{S}_n$, this is not too harmful because they can be either relabeled or not used in training. This is because such easy and correctly labeled samples are often redundant in the training set, and are likely to be inexpensive to label or replace.

Various label-noise detection and data partitioning methods have been proposed in the literature, including Thres_Loss [Huang et al. 2019], $\text{Thres}_\text{acc over training}$ [Sun et al. 2020], Thres_AUM [Pleiss et al. 2020a], 1d-GMM_AUL [Pleiss et al. 2020b], and $\text{2d-GMM}_\text{WJSD-ACD}$ [Zhang et al. 2022]. However, as discussed in Section 5.3, metrics such as loss and confidence fail to differentiate between hardness h and noisiness n, making it challenging to remove only noisy samples without removing hard ones. The same issue applies to other metrics such as accuracy over training, AUL, AUM, JSD, and ACD, as depicted in Fig. 5.7 provided in the appendix; these metrics are monotonic in both h and n, as shown in the figure. Consequently, if these metrics are used for data partitioning, the identified noisy subset may mistakenly contain hard samples, reducing the reliability of these methods when dealing simultaneously with noisy and hard samples. In contrast, SCD increases with n but does *not* increase with h. We can observe this behavior in Fig. 5.5 and compare it with for example the behavior of JSD when hardness and noisiness increase. This behavior makes SCD a particularly promising metric for removing noisy samples while preserving hard samples.



(d) Hardness via Imbalance (e) Hardness via Diversification (f) Hardness via Closeness to the Decision Boundary

Figure 5.5: JSD (top) and SCD (botton) applied to the modified datasets with noisiness level n and hardness level h. We observe that SCD increases with n, but this is not observed with h. In contrast, JSD increases with both n and h.

While SCD is a promising metric for distinguishing hard samples from noisy ones, it is beneficial to pair it with another metric that can detect easy samples from the hard and noisy ones. In our experiments, we have observed that accuracy over training is an effective metric for this purpose. It can accurately identify the easy samples in the dataset while mixing the hard and noisy samples together. Since these two metrics - SCD and accuracy over training - are complementary, we use a 2d-GMM with them as its dimensions to estimate all three subsets - S_e, S_h, and S_n. By combining SCD and accuracy over training, we are able to achieve better subset detection overall. Specifically, using SCD improves Recall_h, while accuracy over training improves Recall_n.

Table 5.1 presents the results of different partitioning methods on all three synthetic hard datasets, including the resulting dataset size, correct label percentage, $\text{Precision}_\text{n}$, Recall_n, and Recall_h. Several interesting observations emerge from the results. Firstly, some methods, such as Thres_AUM, have high Recall_n but low Recall_h. These methods struggle to differentiate hard samples from noisy ones, and thus discard most of the hard samples. Secondly, some methods, such as $\text{2d-GMM}_\text{WJSD-ACD}$, have high Recall_h but low Recall_n. Although they preserve hard samples, they also retain noisy ones. Thirdly, these observations vary depending on the type of hardness, making the conclusions less robust. However, we consistently observe that our proposed method, $\text{2d-GMM}_\text{acc-SCD}$, performs the best overall and robustly across all three datasets, with high Recall_n and Recall_h. Such robustness is particularly crucial in practice

since samples can be hard to learn due to any combination of the hardness types, making a reliable metric even more necessary.

5.4.2 Training on the Filtered Subset

In this section, we present the results of training models from scratch on the estimated clean datasets $\tilde{S}_c$ obtained in the previous section. Table 5.2 displays the generalization performance of models trained on the estimated clean datasets using each method. The results demonstrate that our proposed method 2d-GMM$_{\text{acc-SCD}}$ consistently outperforms the other methods in all three settings. We observe that the 1d-GMM$_{\text{AUL}}$ method performs slightly better than 2d-GMM$_{\text{acc-SCD}}$ in the second and third settings. However, in the first setting, 1d-GMM$_{\text{AUL}}$ performs significantly worse than 2d-GMM$_{\text{acc-SCD}}$, which suggests that it is not robust to the hardness type and thus unreliable for practical use.

Overall, the table emphasizes the importance of selecting an appropriate metric/approach for data filtration in presence of both noisy and hard samples. Moreover, our experiments highlight the advantage of using the label noise detection method 2d-GMM$_{\text{acc-SCD}}$ in different settings.

Experiments on datasets with real-world label noise To further evaluate and compare the performance of each data filtration method, we apply them to datasets with real-world label noise. Tables 5.3 and 5.4 display the results of each method for partitioning and data filtration, for models trained on the Animal-10N [2] and CIFAR-10N [Wei et al. 2022] datasets, respectively. Both datasets have real-world label noise and unknown easy-hard-noisy subsets. Since we lack knowledge of which samples are difficult or incorrectly labeled, we cannot compute Recall$_{\text{h}}$ or Recall$_{\text{n}}$, as we did for our created synthetic datasets in Table 5.1. Nonetheless, we can apply each partitioning/label-noise detection method to these datasets, partition them into an estimated clean subset $\tilde{S}_c$ and an estimated noisy subset $\tilde{S}_n$, and train models on $\tilde{S}_c$. The generalization performance of models trained on $\tilde{S}_c$ obtained from each method is an indication of the quality of the estimated clean subsets $\tilde{S}_c$. Our results in Tables 5.3 and 5.4 demonstrate that our proposed method, 2d-GMM$_{\text{acc-SCD}}$, outperforms all other methods by a significant margin in terms of test performance and in terms of estimating the label noise level of the given dataset.

5.5 Discussion

Robustness to Hardness Type In this study, we aimed to explore different approaches for manipulating the difficulty of samples and investigate their impact on label noise detection methods. We simulated three different hardness types and compared the performance of various methods in identifying and removing noisy samples while retaining hard ones. It is important to highlight that sample difficulty$h$ is a relative concept and can only be evaluated

[2] https://dm.kaist.ac.kr/datasets/animal-10n/

when comparing samples within a given dataset. This is different from noisiness level n, which is an absolute measure. Our simulations showed that the three hardness types that we tested were sufficiently distinct. Interestingly, we found that some methods perform better in distinguishing certain types of hard samples from noisy ones, while not performing well in distinguishing other hard samples. However, the 2D-GMM on top of accuracy over training and SCD, demonstrated the most robust performance across all three types of hardness. The reason for the rather significant superior performance of this method in real-world datasets stems from its ability to effectively detect label noise and hard samples caused by various underlying reasons, which are often unpredictable in real-world datasets. When proposing a label noise detection method, it is essential to consider the unpredictability of hard samples in real-world datasets.

Number of Clusters in GMM We observe that in our synthetic datasets, the location of easy S_e, hard S_h, and noisy S_n samples in the 2D spectrum with accuracy over training and SCD followed a certain pattern. We illustrate this pattern in Figure 5.6 for the hardness via diversification dataset. Furthermore, we found that this pattern was consistent across different hardness types. To further investigate this observation, we trained GMM models on two and three clusters and analyzed the resulting clusters. We found that when we used two clusters, many hard and noisy samples were clustered together, resulting in poor detection performance. However, when we used three clusters, the detection performance improved significantly, and the resulting clusters were much more coherent with the actual clusters. This observation led us to use a 3-cluster GMM in our proposed method, 2d-$\text{GMM}_{\text{acc-SCD}}$, which produced more accurate results. It is important to note that such investigation is only possible through our controlled experiments on synthetic datasets because we know the exact partitions of the datasets into easy, hard, and noisy samples. Nevertheless, our findings provide valuable insights into the design of label noise detection methods.

Conclusion We propose an empirical approach to investigate hard samples in an image classification setting. To this end, we create synthetic datasets that enable us to study hard and noisy samples in a controlled environment. Through our investigation, we make several interesting observations, including the importance of analyzing feature layer vectors of neural networks to distinguish between hard and noisy samples. Furthermore, we propose a label noise detection method that outperforms other existing methods in terms of both removing noisy samples and retaining hard ones. Our method can be applied to filter datasets with label noise, leading to better generalization performance when training models on the filtered set. Importantly, our label noise detection method is of quite general use and can be used in combination with any other method designed to deal with label noise and/or hard samples. Although our synthetic datasets were tailored towards image classification tasks, our conclusions could be applied to any application that involves a neural network and label noise. As a future direction, it would be interesting to test our method in other applications/domains as well. Moreover, our data filtration method could be combined with semi-supervised learning methods, such as FixMatch and FlexMatch, to further improve the model's generalization performance by making use of the discarded subset of the data. Overall, our study provides

valuable insights into the design and optimization of label noise detection methods and their applications in improving the performance of machine learning models in real-world settings.

Method	$\lvert\tilde{S}_c\rvert$	Correct Label % of $\tilde{S}_c$	$\text{Precision}_\text{n}$ $\frac{\lvert\tilde{S}_n \cap S_n\rvert}{\lvert\tilde{S}_n\rvert}$	Recall_n $\frac{\lvert\tilde{S}_n \cap S_n\rvert}{\lvert S_n\rvert}$	Recall_h $\frac{\lvert\tilde{S}_c \cap S_h\rvert}{\lvert S_h\rvert}$
Original dataset	$31k$	0.80	NA	NA	NA
Thres_Loss [Huang et al. 2019]	$15.5k$	0.88	0.27	0.69	0.03
$\text{Thres}_\text{acc over training}$ [Sun et al. 2020]	$15.7k$	**0.95**	0.35	**0.88**	0.003
Thres_AUM [Pleiss et al. 2020a]	$15.5k$	**0.95**	0.34	0.86	0.001
1d-GMM_Loss [Li et al. 2020a]	$29.1k$	0.81	0.31	0.10	0.57
1d-GMM_AUL [Pleiss et al. 2020b]	$25.6k$	0.88	**0.59**	0.52	0.12
2d-$\text{GMM}_\text{WJSD-ACD}$ [Zhang et al. 2022]	$18.4k$	0.77	0.14	0.30	**0.73**
2d-$\text{GMM}_\text{acc-SCD}$ (Ours)	$25.7k$	0.83	0.44	0.46	0.61

(a) Hardness via imbalance dataset

Original dataset	$80k$	0.80	NA	NA	NA
Thres_Loss [Huang et al. 2019]	$40k$	0.89	0.29	0.73	0.38
$\text{Thres}_\text{acc over training}$ [Sun et al. 2020]	$41k$	**0.96**	0.36	**0.90**	0.41
Thres_AUM [Pleiss et al. 2020a]	$40k$	**0.96**	0.35	**0.90**	0.37
1d-GMM_Loss [Li et al. 2020a]	$65.8k$	0.84	0.38	0.34	0.66
1d-GMM_AUL [Pleiss et al. 2020b]	$63k$	0.91	**0.58**	0.63	**0.69**
2d-$\text{GMM}_\text{WJSD-ACD}$ [Zhang et al. 2022]	$46.1k$	0.80	0.19	0.41	0.50
2d-$\text{GMM}_\text{acc-SCD}$ (Ours)	$62.2k$	0.86	0.38	0.43	0.68

(b) Hardness via diversification dataset

Original dataset	$27.6k$	0.80	NA	NA	NA
Thres_Loss [Huang et al. 2019]	$13.8k$	0.93	0.32	0.82	0.46
$\text{Thres}_\text{acc over training}$ [Sun et al. 2020]	$15.9k$	**0.99**	0.46	**0.99**	0.59
Thres_AUM [Pleiss et al. 2020a]	$13.8k$	**0.99**	0.39	**0.99**	0.51
1d-GMM_Loss [Li et al. 2020a]	$26.2k$	0.84	**0.84**	0.22	**0.79**
1d-GMM_AUL [Pleiss et al. 2020b]	$19.4k$	**0.99**	0.65	0.97	0.70
2d-$\text{GMM}_\text{WJSD-ACD}$ [Zhang et al. 2022]	$22k$	0.85	0.37	0.38	0.67
2d-$\text{GMM}_\text{acc-SCD}$ (Ours)	$23.1k$	0.92	0.81	0.68	0.77

(c) Hardness via closeness to the decision boundary dataset

Table 5.1: Partitioning results obtained by applying various label noise detection methods on our three synthetic datasets with hard samples. We can observe that, on one hand, some methods have good performance on recall only on noisy samples (for example $\text{Thres}_\text{acc over training}$), whereas they have very bad performance in terms of recalling hard samples. This means that, in an attempt to remove noisy samples from the dataset, they remove almost all hard samples as well. On the other hand, some methods have a good performance in terms of keeping hard samples (for example 2d-$\text{GMM}_\text{WJSD-ACD}$), but fail to remove the noisy samples from the training dataset, as evidenced by the very low value of Recall_n. Our proposed metric, 2d-$\text{GMM}_\text{acc-SCD}$, shows the overall best performance in terms of removing noisy samples while keeping hard samples, as evidenced by the relatively high values of Recall_h and Recall_n. The result is consistent in all three settings, unlike some methods that only perform well in one of the settings.

Method	Test Accuracy	Test Loss	Top-5 Test Accuracy
$\text{Thres}_{\text{Loss}}$	$25.83_{\pm 0.06}\%$	$4.59_{\pm 0.02}$	$43.94_{\pm 0.17}\%$
$\text{Thres}_{\text{acc over training}}$	$25.34_{\pm 0.42}\%$	$6.04_{\pm 0.02}$	$39.66_{\pm 0.25}\%$
$\text{Thres}_{\text{AUM}}$	$23.85_{\pm 0.21}\%$	$6.31_{\pm 0.01}$	$36.65_{\pm 0.08}\%$
1d-GMM_{Loss}	$31.08_{\pm 0.13}\%$	$3.68_{\pm 0.02}$	$53.98_{\pm 0.30}\%$
1d-GMM_{AUL}	$30.27_{\pm 0.34}\%$	$5.22_{\pm 0.01}$	$47.55_{\pm 0.26}\%$
2d-$\text{GMM}_{\text{WJSD-ACD}}$	$32.11_{\pm 0.15}\%$	$3.29_{\pm 0.00}$	$56.26_{\pm 0.11}\%$
2d-$\text{GMM}_{\text{acc-SCD}}$ (Ours)	$\mathbf{34.88}_{\pm \mathbf{0.13}}\%$	$\mathbf{3.27}_{\pm 0.01}$	$\mathbf{60.17}_{\pm 0.29}\%$

(a) Hardness via imbalance dataset

$\text{Thres}_{\text{Loss}}$	$42.65_{\pm 0.07}\%$	$2.86_{\pm 0.01}$	$67.08_{\pm 0.17}\%$
$\text{Thres}_{\text{acc over training}}$	$44.39_{\pm 0.16}\%$	$2.75_{\pm 0.00}$	$69.79_{\pm 0.15}\%$
$\text{Thres}_{\text{AUM}}$	$44.35_{\pm 0.41}\%$	$2.74_{\pm 0.02}$	$70.10_{\pm 0.39}\%$
1d-GMM_{Loss}	$42.93_{\pm 0.37}\%$	$2.96_{\pm 0.02}$	$66.02_{\pm 0.23}\%$
1d-GMM_{AUL}	$\mathbf{45.56}_{\pm 0.39}\%$	$\mathbf{2.74}_{\pm 0.01}$	$\mathbf{70.36}_{\pm 0.13}\%$
2d-$\text{GMM}_{\text{WJSD-ACD}}$	$38.83_{\pm 0.34}\%$	$3.11_{\pm 0.04}$	$62.78_{\pm 0.67}\%$
2d-$\text{GMM}_{\text{acc-SCD}}$ (Ours)	$\mathbf{45.36}_{\pm 0.33}\%$	$\mathbf{2.78}_{\pm 0.01}$	$\mathbf{69.25}_{\pm 0.32}\%$

(b) Hardness via diversification dataset

$\text{Thres}_{\text{Loss}}$	$43.30_{\pm 0.33}\%$	$2.61_{\pm 0.00}$	$69.27_{\pm 0.15}\%$
$\text{Thres}_{\text{acc over training}}$	$49.13_{0.43}\%$	$2.42_{\pm 0.01}$	$74.20_{\pm 0.12}\%$
$\text{Thres}_{\text{AUM}}$	$47.67_{\pm 0.09}\%$	$2.44_{\pm 0.01}$	$72.79_{\pm 0.05}\%$
1d-GMM_{Loss}	$44.93_{\pm 0.17}\%$	$2.65_{\pm 0.01}$	$69.29_{\pm 0.18}\%$
1d-GMM_{AUL}	$\mathbf{50.67}_{\pm 0.26}\%$	$\mathbf{2.31}_{\pm 0.00}$	$\mathbf{75.68}_{\pm 0.10}\%$
2d-$\text{GMM}_{\text{WJSD-ACD}}$	$43.23_{\pm 0.21}\%$	$2.70_{\pm 0.00}$	$68.04_{\pm 0.18}\%$
2d-$\text{GMM}_{\text{acc-SCD}}$ (Ours)	$\mathbf{49.80}_{\pm 0.24}\%$	$\mathbf{2.34}_{\pm 0.01}$	$\mathbf{75.17}_{\pm 0.09}\%$

(c) Hardness via closeness to the decision boundary dataset

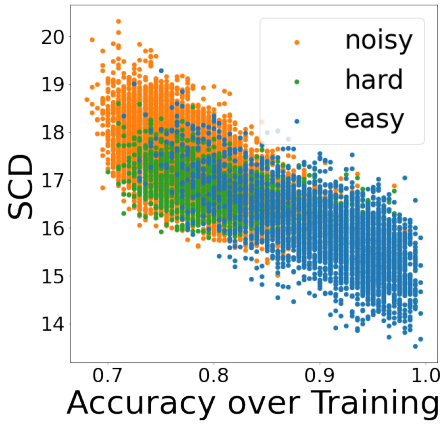
Table 5.2: DenseNet performance comparison on the estimated clean datasets using different metrics/approaches in three experimental settings. Our method 2d-$\text{GMM}_{\text{acc-SCD}}$ consistently performs well in all settings, unlike the 1d-GMM_{AUL} method that performs well only in the second and third settings. It is important to choose a method that works well in all settings because, in practice, hard samples can arise from any of the three reasons tested here.

Method	Test Accuracy	Test Loss	Estimated LNL
$\text{Thres}_{\text{Loss}}$	$80.32_{\pm 0.38}$	$0.62_{\pm 0.01}$	50%
$\text{Thres}_{\text{acc over training}}$	$83.03_{\pm 0.11}$	$0.52_{\pm 0.01}$	2.48%
$\text{Thres}_{\text{AUM}}$	$76.30_{\pm 0.16}$	$0.90_{\pm 0.04}$	50%
1d-GMM_{Loss}	$83.05_{\pm 0.25}\%$	$0.53_{\pm 0.01}$	2.82%
1d-GMM_{AUL}	$82.80_{\pm 0.28}\%$	$0.53_{\pm 0.01}$	2.11%
2d-$\text{GMM}_{\text{WJSD-ACD}}$	$83.06_{\pm 0.26}\%$	$0.54_{\pm 0.01}$	1.03%
2d-$\text{GMM}_{\text{acc-SCD}}$ (Ours)	$\mathbf{83.11}_{\pm 0.10}\%$	$\mathbf{0.51}_{\pm 0.01}$	**3.3%**

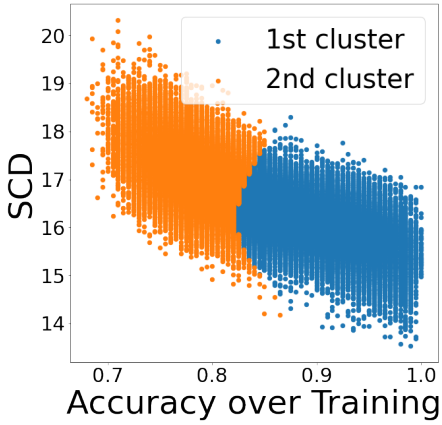
Table 5.3: Performance comparison of models trained on the estimated clean datasets using different metrics/approaches in the Animal-10N dataset. The estimated label noise level (LNL) of this dataset is around 8%, and we observe that our method provides the best LNL estimate. The LNL estimate of each method is computed using $1 - \left|\tilde{S}_c\right| / |S|$. Moreover, our method 2d-$\text{GMM}_{\text{acc-SCD}}$ provides the cleanest dataset as evidenced by the very large margin in the generalization performance improvement; both test accuracy and test loss are the best for 2d-$\text{GMM}_{\text{acc-SCD}}$.

Method	Test Accuracy	Test Loss	Estimated LNL
$\text{Thres}_{\text{Loss}}$	$85.45_{\pm 0.22}$	$0.64_{\pm 0.01}$	50%
$\text{Thres}_{\text{acc over training}}$	$83.61_{\pm 0.17}$	$0.78_{\pm 0.01}$	46%
$\text{Thres}_{\text{AUM}}$	$81.89_{\pm 0.18}$	$0.83_{\pm 0.01}$	50%
$\text{1d-GMM}_{\text{Loss}}$	$85.87_{\pm 0.30}$	$0.62_{\pm 0.00}$	46%
$\text{1d-GMM}_{\text{AUL}}$	$86.21_{\pm 0.06}$	$0.66_{\pm 0.01}$	30%
$\text{2d-GMM}_{\text{WJSD-ACD}}$	$89.44_{\pm 0.22}$	$0.35_{\pm 0.01}$	17%
$\text{2d-GMM}_{\text{acc-SCD}}$ (Ours)	$\mathbf{89.90}_{\pm 0.21}$	$\mathbf{0.31}_{\pm 0.00}$	**10%**

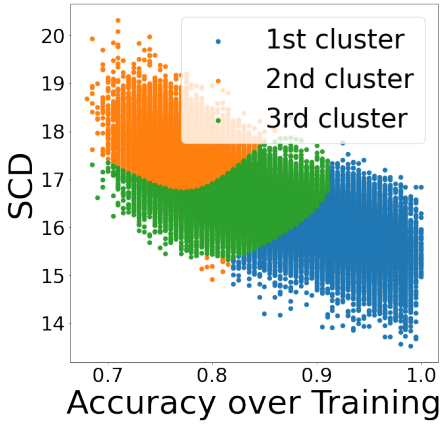
Table 5.4: Performance comparison of models trained on the estimated clean datasets using different metrics/approaches in the CIFAR-10N dataset. The label noise level (LNL) of this dataset is 9.01%, and we observe that our method $\text{2d-GMM}_{\text{acc-SCD}}$ provides the best LNL estimate, computed from $1-\left|\tilde{S}_c\right| / |S|$. Moreover, $\text{2d-GMM}_{\text{acc-SCD}}$ provides the cleanest dataset as evidenced by the generalization performance improvement; both test accuracy and test loss are the best for $\text{2d-GMM}_{\text{acc-SCD}}$.



(a) Easy/Hard/Noisy Clusters



(b) Results of a GMM with 2 Clusters



(c) Results of a GMM with 3 Clusters

Figure 5.6: (a) Easy-hard-noisy clusters for samples of the hardness via diversification dataset with their associated static centroid distance (SCD) and accuracy over training values. We can observe that noisy samples are located on the top left of the figure, followed by the hard samples in the middle parts and the easy samples on the other end, i.e., the bottom right. Easy samples have a high accuracy over training and a low SCD. Depicting such figure requires full ground-truth access of the available dataset and the knowledge about label noise and hardness levels of samples. However, this information is not possible in practice, hence in the sub-figures (b) and (c) we evaluate the results of Gaussian mixture models to recover these clusters without any knowledge about samples. (b) Clustering results of 2d-GMM$_{\text{acc}-\text{SCD}}$ with 2 number of clusters. This clustering does not require the ground-truth access of samples and only requires computation of SCD and accuracy over training. If we compare the clusters with actual easy-hard-noisy partitions in sub-figure (a) we can observe that most of the hard and noisy samples are mixed in the second cluster. (c) Clustering results of 2d-GMM$_{\text{acc}-\text{SCD}}$ with 3 number of clusters. In contrast to clustering with 2 clusters, we can observe that clustering with 3 cluster is much better at including as much noisy samples as possible in the second cluster while not including hard samples. Hence, throughout our study, when referred to our label noise detection method 2d-GMM$_{\text{acc}-\text{SCD}}$, we apply 3 clusters and use the top left cluster as the detected noisy subset.

Appendices

5.A Experimental Setup

Generating Noisy-labeled Datasets We modify original datasets similarly to Chatterjee [2020]; for a fraction of samples denoted by the label noise level (LNL), we replace the labels with independent random variables drawn uniformly from other classes with the same noisiness level n.

For each hardness type, there are two sets of experiments: (i) experiments done to detect clean-noisy subsets, (ii) experiments done to train on filtered sets. We provide details for each of these experiments below.

Hardness via Imbalance Experiments (i) The models are trained for 200 epochs on the cross-entropy objective function with batch size 100, using SGD with learning rate 0.001, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The maximum label noise is set as 40%. The neural network architecture is MobileNetV2. (ii) The models are trained for 50 epochs on the cross-entropy objective function with batch size 100, using SGD with learning rate 0.001, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The neural network architecture is DenseNet121, pre-trained on ImageNet dataset.

Hardness via Diversification Experiments (i) The models are trained for 200 epochs on the cross-entropy objective function with batch size 100, using SGD with learning rate 0.001, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The maximum label noise is set as 40%. The neural network architecture is MobileNetV2. (ii) The models are trained for 50 epochs on the cross-entropy objective function with batch size 100, using SGD with learning rate 0.001, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The neural network architecture is DenseNet121, pre-trained on ImageNet dataset.

Hardness via Closeness to the Decision Boundary Experiments (i) The models are trained for 50 epochs on the cross-entropy objective function with batch size 100, using SGD with learning rate 0.001, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The maximum label noise is set as 40%. The neural network architecture is DesneNet121, pre-trained on ImageNet. (ii) The models are trained for 50 epochs on the cross-entropy objective function with batch size 100, using SGD with learning rate 0.001, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The neural

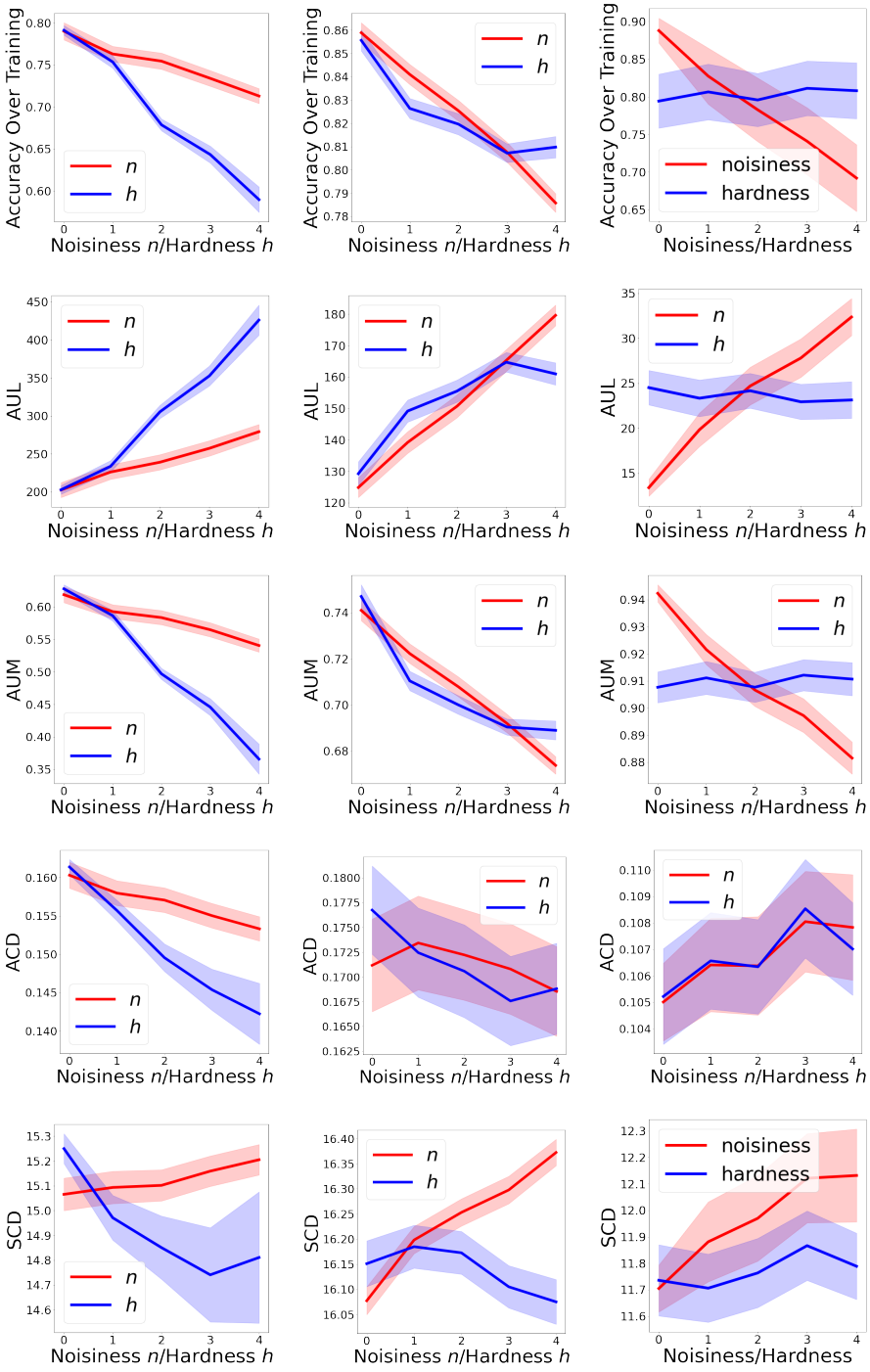
network architecture is DenseNet121, pre-trained on ImageNet dataset.

CIFAR-10N Experiments The models are trained for 50 epochs on the cross-entropy objective function with batch size 64, using SGD with learning rate 0.05, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The neural network architecture is MobileNetV2.

Animal-10N Experiments The models are trained for 40 epochs on the cross-entropy objective function with batch size 64, using SGD with learning rate 0.001, weight decay $5 \cdot 10^{-4}$, and momentum 0.9. The neural network architecture is AlexNet.

5.B Comparison to Other Metrics

Fig. 5.7 shows a comparison of different metrics and how they behave concerning changes in the hardness level h and noisiness level n. From the graph, we can see that all metrics either increase or decrease as the value of n increases. However, we also observe that the same increase or decrease happens when at least one type of hardness h increases. In contrast, the SCD metric stands out as it does not show the same trend towards an increase in any type of hardness level h as the value of n increases.



(m) Hardness via Imbalance (n) Hardness via Diversification (o) Hardness via Closeness to the Decision Boundary

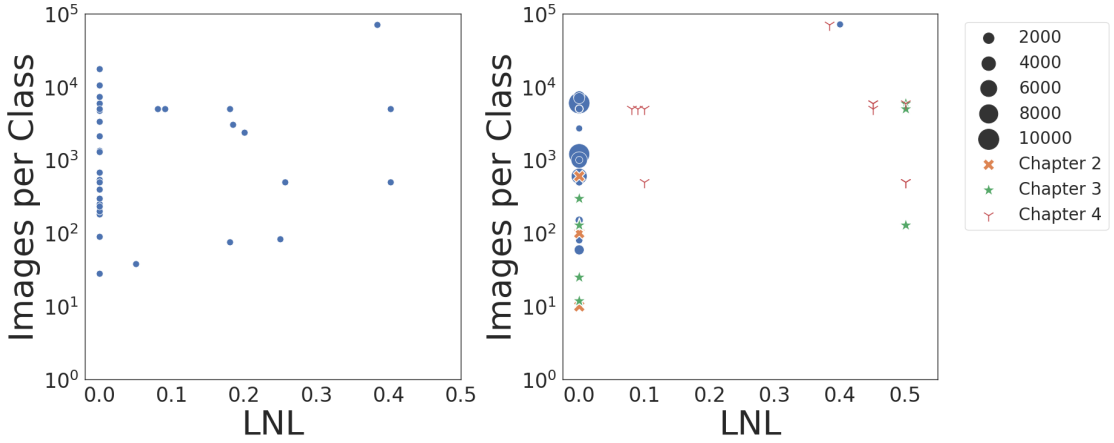
Figure 5.7: Different metrics applied to the transformed datasets with noisiness level n and hardness level h. Left, middle and right figures correspond to hardness types imbalanced, diversification, and closeness to the decision boundary, respectively. We observe that all the metrics have at least one type of hardness h that with its increase, the metric shows the same behavior as with increase of noisiness n. In contrast, SCD is the only metric that shows different behaviors with increases in h and n.

6 Conclusion

The goal of this thesis is to investigate practical methods to gain information regarding model generalization, learning, memorization and the dataset quality specifically in the regime of datasets with non-zero label noise level and limited labeled samples. In this concluding chapter, we summarize the key findings that emerged from our research. We have demonstrated the practical advantages of our proposed methods. However, our research was not without limitations, and we will discuss them as well, along with suggestions for future research directions. Ultimately, this thesis has contributed to a deeper understanding of deep neural networks generalization ability and has provided valuable insights that can inform both the deep-learning research and industrial communities.

The motivation for this thesis came from our observation summarized in Fig. 1.2. We showed in Fig. 1.2, there is a need to put more emphasis on studies that focus on datasets with limited size and with label noise, which are commonly encountered in practical applications. Deep learning research often benefits from working with large and high-quality datasets, which require significant resources in terms of funding, time, and expertise. However, many practitioners who apply machine learning methods may not have access to such resources. Therefore, it is crucial to develop and evaluate methods that can effectively operate under these more realistic conditions. This is the mindset that motivated the development of different methods presented in this thesis. In the end, we are confident that we have achieved our goal. In Fig. 6.1, we showcase the datasets examined in the publications resulting from this thesis. Our research successfully fills the gap in the right-center/bottom segment of the dataset spectrum, that is datasets with limited size and label noise.

In Chapter 2, we explore the advantages and limitations of sensitivity, defined in Eq. (2.2), a computationally cheap metric that requires only unlabeled data, as a generalization measure and provide alternative explanations for benefits of certain design choices. We establish a link between neural network output sensitivity and test loss, which allows for sensitivity to be potentially used in applications beyond just classification. By linking sensitivity and generalization, we present a new viewpoint on understanding the success of current state-of-the-art architectures and initialization techniques. In this chapter, we look at the success



(a) Datasets (b) Number of Papers

Figure 6.1: (a) Number of images per class (a token for dataset size) and the label noise level (LNL) of datasets introduced in recent image classification survey papers. (b) Evolution of Fig. 1.2(b) positioning the contributions of the thesis: The number of images per class and the label noise level (LNL) of datasets in https://paperswithcode.com/ website under the filtration of image classification. In addition, in this figure, we add the three papers which were published as part of this thesis. Chapter 2 is based on [Forouzesh et al. 2021]. Chapter 3 is based on [Forouzesh and Thiran 2021]. Chapter 4 is based on [Forouzesh et al. 2023].

of dropout and batch normalization from another perspective: These methods decrease the output sensitivity to random input perturbations in a same manner as they decrease the test loss, resulting in better generalization performance. Our conclusion is that, sensitivty is a very accurate generalization measure. In particular, when the parameter initialization of the neural networks follows standard normal, and when the architecture is rather shallow (for example a 4-layer fully-connected neural network, or a 4-layer convolutional neural network), then the test loss prediction can be done very accurately; the empirical values follow our established link rather strongly, which is positively surprising. This is very important to note, so that when in practice, the setting does not allow us to use fancy architectures, we are able to predict test loss to a decent degree. We also further test sensitivity, as a zero-cost neural architecture search tool, and show its great potential compared to other metrics, having in mind that sensitivity does not require training, that it can be computed at initialization, and that it does not require any backward pass, and that it is very computationally cheap.

In Chapter 3, we introduce gradient disparity, defined by Eq. (3.7), as an efficient early stopping criterion computed entirely on the training set. Gradient disparity is derived from a probabilistic upper bound on generalization penalty (Eq. (3.4)) under the PAC-Bayesian framework, by assuming that the weight vectors of the neural network follow a Gaussian distribution. Its computational efficiency stems from the fact that it requires only the computation of gradients of two mini-batches, which are already computed in an iterative optimization method based on gradient descent. Hence, no computational overhead is added to the regular

training process for computing gradient disparity. The advantage of gradient disparity lies in its use of only the training set, making it particularly effective when compared to k-fold cross-validation. Despite using all available samples in both training and validation, k-fold cross-validation still results in worse performance than models trained on the entire dataset, as part of the data is always set aside for validation in each of its k folds. Our experiments demonstrate the performance improvement brought by gradient disparity when dealing with limited and/or noisy datasets. Additionally, gradient disparity as an early stopping criterion is less sensitive to the early stopping threshold than the validation loss, and is computationally advantageous compared to k-fold cross-validation. Furthermore, our tests of gradient disparity on different cross-validation methods in time-series applications suggest that it is a promising technique in such settings, outperforming other cross-validation methods due to its ability to train the model on the entire available dataset.

In Chapter 4, we propose an efficient and effective method to track label noise memorization, using the susceptibility metric defined in Eq. (4.2), which uses only a single mini-batch of unlabeled data. This metric is motivated by our empirical and theoretical observations that models with high test accuracy are less prone to memorizing a randomly-labeled set of samples. Our theoretical observation is based on convergence analysis of models trained on randomly-labeled datasets. Specifically, in Theorem 4, we demonstrate that models pre-trained on high-quality data, and hence with high generalization performance, take longer to converge on low-quality data. Our proposed susceptibility metric efficiently computes a model's resistance to memorizing a randomly-labeled set. Furthermore, through extensive experiments on various datasets, architectures, and hyper-parameters, we demonstrate that combining susceptibility with overall training accuracy is an effective approach for model selection. Models with low susceptibility and high training accuracy demonstrate lower memorization on the training set and better generalization to unseen data. This method eliminates the need for accessing ground-truth labels for tracking memorization or conducting model selection. We also perform ablation studies to show the robustness of our approach, including varying levels of label noise and mini-batch sizes. In summary, we propose our model selection method as a superior alternative to using a validation set, particularly in datasets with high levels of label noise and limited size. In other scenarios, our method performs comparably to using a validation set. Our approach ensures that we do not sacrifice performance in good-quality datasets while achieving substantial gains in low-quality datasets.

In Chapter 5, we present a novel label noise detection method based on the static centroid distance metric, defined in Section 5.2. This metric is well-suited for datasets that contain both noisy and hard samples. We introduce this metric through a controlled empirical study on datasets with synthetically created hard samples. To create these samples, we use three different approaches that cover a wide range of reasons why a sample might be difficult for a classifier to learn. Our results demonstrate that our method has high recall jointly for both noisy and hard samples, contrary to existing label noise detection methods. These methods often struggle to distinguish between hard and noisy samples, resulting in the incorrect removal of hard samples or retention of noisy samples. However, our method consistently

outperforms others in all three types of hardness, resulting in the best overall performance for removing noisy samples while retaining hard ones. Furthermore, when we train models on the filtered datasets, we observe that our method outperforms other label noise detection methods. Additionally, when we apply our method to datasets with real-world label noise, we find that it, *even much more* significantly than our controlled experiments, outperforms other methods in terms of filtering noisy data. This is evidenced by the high generalization performance of models trained on the filtered set.

We now discuss some of this work's limitations.

In the rich man's world. The central focus of this thesis is to address the challenges associated with datasets that have label noise and are limited in size. Although extensive research has been conducted on datasets with a large number of samples and low label noise levels, datasets with limited size and label noise levels require a specific approach. This thesis does not target researchers or practitioners who have access to a dataset with high quality and a large number of samples, and have high computational resources. However, it is crucial to acknowledge that high-quality data is not always easily accessible, even for large companies with seemingly unlimited resources such as money and expertise. Time constraints can often prove to be a limiting factor. Companies need to develop and improve their products as quickly as possible to generate revenue, but data collection rates may not keep up with the demand for such products. Therefore, even large companies may face challenges in obtaining high-quality data.

Thesis focus. Our studies are primarily focused on image classification tasks. For instance, the synthetic datasets that we developed in Chapter 5, in particular the hardness via diversification dataset was created with the intent of diversifying image hardness and was exclusively designed for images. Consequently, our claims are limited to this specific task. Although several of the metrics and methods that we proposed have potential for generalization to other tasks like text classification, we have not extensively evaluated them in those domains. Thorough testing in other contexts would broaden the scope of our work. Nonetheless, our study offers valuable insights and strategies for enhancing image classification performance, which could serve as a foundation for future research in various domains.

Method requirements. In addition to our primary focus on image classification tasks, our research is further constrained by the specific settings that we assume and study. For instance, in the gradient disparity metric proposed in Chapter 3, we make the assumption that the optimization process is a variant of mini-batch gradient descent, which is a commonly used optimization method in deep learning. Although such constraints are unavoidable, we have made a diligent effort to clearly state our assumptions and the settings we have studied.

Theoretical assumptions. Our research is also constrained by certain assumptions in our theoretical work. For instance, in Chapter 4, our theoretical results apply to a one-layer neural network with a large number of hidden units. While this assumption is commonly used in theoretical deep learning research, it is not necessarily reflective of the neural networks utilized

in practice. As a result, our theoretical results may not be directly applicable to real-world scenarios, as most theoretical results in this area, but they offer valuable insights into the issues that arise in such scenarios. Moreover, while we have attempted to provide theoretical justifications for our proposed methods and metrics, we do not have theoretical guarantees for every claim we make. It is challenging to prove every empirical observation, especially in the context of deep learning, where the complexity of the models can make theoretical analysis difficult.

Empirical limits. The majority of our claims in this thesis are based on empirical findings, and we acknowledge that we cannot definitively prove the generalizability of our claims in every situation. Despite our efforts to provide detailed, thorough, and systematic empirical studies, there may be certain scenarios that we did not test, such as specific architectures, methods, or datasets. This is partly due to time and computational budget constraints. However, for each study, we made a decent attempt to test the most critical elements that could affect the results by conducting thoughtful ablation studies. While we cannot claim universal generalizability, we believe that our findings provide valuable insights and directions for further research in the field of deep learning.

What have we done to the world? It is essential to consider the environmental impact of our empirical studies. This aspect should be given greater attention in the scientific community. While this thesis involved running numerous experiments, which inevitably results in negative environmental effects, our goal was to propose methods that can replace more environmentally harmful approaches. For instance, in Chapter 3, we outperformed the commonly used k-fold cross-validation approach, which requires training k times; whereas our approach only requires one round of training. Although we acknowledge the negative environmental impact of our work, our aim is to ultimately contribute to the development of more environmentally friendly machine learning experiments.

We now discuss some directions for future research.

Testing generalizability. An important direction for future work is to test the generalizability of the methods developed in this thesis by exploring their applicability in other machine-learning tasks. For instance, natural language processing (NLP) has recently emerged as a promising area of research, with large language models dominating the field. To stay up to date with the latest trends, it would be interesting to investigate the performance of our metrics and methods in NLP tasks. However, it should be noted that some of the metrics may need to be tailored for these tasks. For instance, sensitivity, as defined in Eq. (2.2), may need to be adjusted to incorporate input perturbations, such as adding small noise into the word embeddings or replacing words with their closest neighbors. These adjustments should be studied before directly applying our metrics to NLP tasks.

Combination of techniques. This thesis primarily focused on studying supervised learning methods, which involve training models on labeled datasets to make predictions on new, unseen data. However, there are other techniques that have been developed to deal with small

datasets, such as semi-supervised learning and few-shot learning. Semi-supervised learning involves training a model on both labeled and unlabeled data to improve its performance on the target task, while few-shot learning aims to learn from a small number of examples, often just one or a few, for the target task. Exploring the combination of these techniques with our approach could provide interesting insights and improvements for addressing the challenges of small datasets. For instance, combining our supervised learning method with semi-supervised learning could help leverage the additional unlabeled data and potentially improve the performance of the model. It is important to note that incorporating these techniques does not make our research on basic supervised learning methods obsolete, as they are complementary and can be used in conjunction to further enhance performance. Moreover, such an exploration could open up new research avenues and lead to the development of more powerful and robust machine learning models. By leveraging the strengths of each method, we may achieve even better performance and generalizability.

Expanding applicability. Exploring the potential applications of the proposed metrics and methods in various contexts presents a fascinating avenue for future research. For instance, an exciting direction to investigate is whether sensitivity, as defined in Eq. (2.2), can serve as a regularizer. This would entail examining the effects of sensitivity as a regularization term on different machine learning models and tasks. Another investigation could involve incorporating gradient disparity, as defined in Eq. (3.6), into the optimization algorithm and developing a novel variant of stochastic gradient descent. Furthermore, susceptibility, as defined in Eq. (4.2), could potentially be utilized as a noisy-label detection approach, where the training procedure of different samples could be compared to susceptibility. Noisy-labeled samples might exhibit similar training changes in one step, as susceptibility, which measures loss changes on the loss of randomly-labeled samples. Such extensions would enhance the practical benefits of the proposed metrics and methods, making them more widely applicable across diverse fields.

Strengthening theoretical guarantees. Another essential direction for future research is to provide stronger theoretical guarantees for the proposed methods. While the empirical results of our metrics have been promising, it is crucial to establish theoretical guarantees to ensure their reliability and usefulness in practical applications. Currently, most of the claims made about the proposed methods are based on empirical evidence. Although we do have some theoretical claims, they often rely on assumptions that may not be close to real-world scenarios. Therefore, further research is needed to develop stronger theoretical frameworks that can provide more reliable and accurate predictions of the proposed methods' performance. By establishing stronger theoretical guarantees, we can enhance the credibility of the proposed methods and enable their widespread adoption in various fields.

Bibliography

Abdelfattah, M. S., Mehrotra, A., Dudziak, Ł., and Lane, N. D. (2021). Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*.

Aggarwal, P., Mishra, N. K., Fatimah, B., Singh, P., Gupta, A., and Joshi, S. D. (2022). Covid-19 image classification using deep learning: Advances, challenges and opportunities. *Computers in Biology and Medicine*, page 105350.

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430.

Algan, G. and Ulusoy, I. (2020). Label noise types and their effects on deep learning. *arXiv preprint arXiv:2003.10471*.

Allen-Zhu, Z., Li, Y., and Liang, Y. (2018). Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*.

Andreassen, A., Bahri, Y., Neyshabur, B., and Roelofs, R. (2021). The evolution of out-of-distribution robustness throughout fine-tuning. *arXiv preprint arXiv:2106.15831*.

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79.

Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. (2019). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018). Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR.

Athalye, A., Carlini, N., and Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR.

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Bai, Y. and Liu, T. (2021). Me-momentum: Extracting hard confident examples from noisily labeled data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9312–9321.

Baldock, R., Maennel, H., and Neyshabur, B. (2021). Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249.

Beery, S., Liu, Y., Morris, D., Piavis, J., Kapoor, A., Joshi, N., Meister, M., and Perona, P. (2020). Synthetic examples improve generalization for rare classes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 863–873.

Bellido, I. and Fiesler, E. (1993). Do backpropagation trained neural networks have normal weight distributions? In *International Conference on Artificial Neural Networks*, pages 772–775. Springer.

Bengio, E., Pineau, J., and Precup, D. (2020). Interference and generalization in temporal difference learning. *arXiv preprint arXiv:2003.06350*.

Bengio, Y. and Delalleau, O. (2011). On the expressive power of deep architectures. In *International Conference on Algorithmic Learning Theory*, pages 18–36. Springer.

Bengio, Y. and Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of machine learning research*, 5(Sep):1089–1105.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Bien, N., Rajpurkar, P., Ball, R. L., Irvin, J., Park, A., Jones, E., Bereket, M., Patel, B. N., Yeom, K. W., Shpanskaya, K., et al. (2018). Deep-learning-assisted diagnosis for knee magnetic resonance imaging: development and retrospective validation of mrnet. *PLoS medicine*, 15(11):e1002699.

Bietti, A. and Mairal, J. (2019). On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*.

Brown, G., Bun, M., Feldman, V., Smith, A., and Talwar, K. (2021). When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 123–132.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

Chang, H.-S., Learned-Miller, E., and McCallum, A. (2017). Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30.

Chatterjee, S. (2020). Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *International Conference on Learning Representations*.

Chatterji, N., Neyshabur, B., and Sedghi, H. (2020). The intriguing role of module criticality in the generalization of deep networks. In *International Conference on Learning Representations*.

Chatterji, N. S., Neyshabur, B., and Sedghi, H. (2019). The intriguing role of module criticality in the generalization of deep networks. *arXiv preprint arXiv:1912.00528*.

Chen, P., Liao, B. B., Chen, G., and Zhang, S. (2019). Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pages 1062–1070. PMLR.

Chen, P., Ye, J., Chen, G., Zhao, J., and Heng, P.-A. (2021a). Robustness of accuracy metric and its inspirations in learning with noisy labels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11451–11461.

Chen, W., Gong, X., and Wang, Z. (2021b). Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*.

Chen, Y., Meng, G., Zhang, Q., Xiang, S., Huang, C., Mu, L., and Wang, X. (2018). Reinforced evolutionary neural architecture search. *arXiv preprint arXiv:1808.00193*.

Cordeiro, F. R. and Carneiro, G. (2020). A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations? In *2020 33rd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 9–16. IEEE.

Cormack, G. V. and Kolcz, A. (2009). Spam filter evaluation with imprecise ground truth. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 604–611.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Deng, Z., He, H., and Su, W. J. (2020). Toward better generalization bounds with locally elastic stability. *arXiv preprint arXiv:2010.13988*.

Dimopoulos, Y., Bourret, P., and Lek, S. (1995). Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters*, 2(6):1–4.

Domingos, P. (2000). A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238.

Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

Dziugaite, G. K., Hsu, K., Gharbieh, W., and Roy, D. M. (2020). On the role of data in pac-bayes bounds. *arXiv preprint arXiv:2006.10929*.

Dziugaite, G. K. and Roy, D. M. (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017.

Feldman, V. (2020). Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959.

Feldman, V. and Zhang, C. (2020). What neural networks memorize and why: Discovering the long tail via influence estimation. *arXiv preprint arXiv:2008.03703*.

Forouzesh, M., Salehi, F., and Thiran, P. (2021). Generalization comparison of deep neural networks via output sensitivity. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7411–7418. IEEE.

Forouzesh, M., Sedghi, H., and Thiran, P. (2023). Leveraging unlabeled data to track memorization. In *The Eleventh International Conference on Learning Representations*.

Forouzesh, M. and Thiran, P. (2021). Disparity between batches as a signal for early stopping. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pages 217–232. Springer.

Fort, S., Nowak, P. K., Jastrzebski, S., and Narayanan, S. (2019). Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*.

Fortmann-Roe, S. (2012). Understanding the bias-variance tradeoff. http://scott.fortmann-roe.com/docs/BiasVariance.html.

Frénay, B. and Verleysen, M. (2013). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Fu, J., Liu, P., Zhang, Q., and Huang, X. (2020). Rethinking generalization of neural models: A named entity recognition case study. *arXiv preprint arXiv:2001.03844*.

Fu, L. and Chen, T. (1993). Sensitivity analysis for input vector in multilayer feedforward neural networks. In *Neural Networks, 1993., IEEE International Conference on*, pages 215–218. IEEE.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136.

Garg, S., Balakrishnan, S., Kolter, J. Z., and Lipton, Z. C. (2021a). Ratt: Leveraging unlabeled data to guarantee generalization. *arXiv preprint arXiv:2105.00303*.

Garg, S., Balakrishnan, S., Lipton, Z. C., Neyshabur, B., and Sedghi, H. (2021b). Leveraging unlabeled data to predict out-of-distribution performance. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Google (2017). Web traffic time series forecasting competi- tion. www.kaggle.com/c/web-traffic-time-series-forecasting.

Gu, J. and Tresp, V. (2019). Neural network memorization dissection. *arXiv preprint arXiv:1911.09537*.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377.

Guo, C., Gardner, J., You, Y., Wilson, A. G., and Weinberger, K. (2019). Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Haghifam, M., Negrea, J., Khisti, A., Roy, D. M., and Dziugaite, G. K. (2020). Sharpened generalization bounds based on conditional mutual information and an application to noisy, iterative algorithms. *arXiv preprint arXiv:2004.12983*.

Han, B., Yao, Q., Liu, T., Niu, G., Tsang, I. W., Kwok, J. T., and Sugiyama, M. (2020). A survey of label-noise representation learning: Past, present and future. *arXiv preprint arXiv:2011.04406*.

Hanin, B. and Rolnick, D. (2018). How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*, pages 571–581.

Harutyunyan, H., Reing, K., Ver Steeg, G., and Galstyan, A. (2020). Improving generalization by controlling label-noise information in neural network weights. In *International Conference on Machine Learning*, pages 4071–4081. PMLR.

He, H. and Su, W. (2020). The local elasticity of neural networks. In *International Conference on Learning Representations*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Heckel, R. and Yilmaz, F. F. (2020). Early stopping in deep networks: Double descent and how to eliminate it. *arXiv preprint arXiv:2007.10099*.

Hesamian, M. H., Jia, W., He, X., and Kennedy, P. (2019). Deep learning techniques for medical image segmentation: achievements and challenges. *Journal of digital imaging*, 32:582–596.

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, J., Qu, L., Jia, R., and Zhao, B. (2019). O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3326–3334.

Hui, L. and Belkin, M. (2020). Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ipeirotis, P. G., Provost, F., and Wang, J. (2010). Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*.

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K., and Geras, K. (2020). The break-even point on optimization trajectories of deep neural networks. *arXiv preprint arXiv:2002.09572*.

Jia, S., Jiang, S., Lin, Z., Li, N., Xu, M., and Yu, S. (2021). A survey: Deep learning for hyperspectral image classification with few labeled samples. *Neurocomputing*, 448:179–204.

Jiang, A. Q. and Gal, L. S. C. L. Y. (2021). Can network flatness explain the training speed-generalisation connection?

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018a). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR.

Jiang, Y., Foret, P., Yak, S., Roy, D. M., Mobahi, H., Dziugaite, G. K., Bengio, S., Gunasekar, S., Guyon, I., and Neyshabur, B. (2020). Neurips 2020 competition: Predicting generalization in deep learning. *arXiv preprint arXiv:2012.07976*.

Jiang, Y., Krishnan, D., Mobahi, H., and Bengio, S. (2018b). Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. (2019). Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12.

Kaissis, G. A., Makowski, M. R., Rückert, D., and Braren, R. F. (2020). Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311.

Kandel, I. and Castelli, M. (2020). Transfer learning with convolutional neural networks for diabetic retinopathy image classification. a review. *Applied Sciences*, 10(6):2021.

Karimi, D., Dou, H., Warfield, S. K., and Gholipour, A. (2020). Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical image analysis*, 65:101759.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Kim, S. and Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 32(3):669–679.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kishida, I. and Nakayama, H. (2019). Empirical study of easy and hard examples in cnn training. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part IV 26*, pages 179–188. Springer.

Kline, D. M. and Berardi, V. L. (2005). Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4):310–318.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Krueger, D., Ballas, N., Jastrzebski, S., Arpit, D., Kanwal, M. S., Maharaj, T., Bengio, E., Fischer, A., and Courville, A. (2017). Deep nets don't learn via memorization.

Kyriakides, G. and Margaritis, K. (2020). An introduction to neural architecture search for convolutional networks. *arXiv preprint arXiv:2005.11074*.

Lam, C. P. and Stork, D. G. (2003). Evaluating classifiers by means of test data with noisy labels. In *IJCAI*, volume 3, pages 513–518. Citeseer.

Lara-Benítez, P., Carranza-García, M., and Riquelme, J. C. (2021). An experimental review on deep learning architectures for time series forecasting. *International journal of neural systems*, 31(03):2130001.

Le, Y. and Yang, X. (2015). Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583.

Li, J., Socher, R., and Hoi, S. C. (2020a). Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*.

Li, J., Wong, Y., Zhao, Q., and Kankanhalli, M. S. (2019a). Learning to learn from noisy labeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5051–5059.

Li, J., Zhang, M., Xu, K., Dickerson, J., and Ba, J. (2021). How does a neural network's architecture impact its robustness to noisy labels? *Advances in Neural Information Processing Systems*, 34.

Li, J., Zhang, M., Xu, K., Dickerson, J. P., and Ba, J. (2020b). Noisy labels can induce good representations. *arXiv preprint arXiv:2012.12896*.

Li, M., Soltanolkotabi, M., and Oymak, S. (2020c). Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4313–4324. PMLR.

Li, S., Song, W., Fang, L., Chen, Y., Ghamisi, P., and Benediktsson, J. A. (2019b). Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6690–6709.

Liang, X., Yao, L., Liu, X., and Zhou, Y. (2022). Tripartite: Tackle noisy labels by a more precise partition. *arXiv preprint arXiv:2202.09579*.

Liu, J., Bai, Y., Jiang, G., Chen, T., and Wang, H. (2020a). Understanding why neural networks generalize well through gsnr of parameters. In *International Conference on Learning Representations*.

Liu, S., Niles-Weed, J., Razavian, N., and Fernandez-Granda, C. (2020b). Early-learning regularization prevents memorization of noisy labels. *arXiv preprint arXiv:2007.00151*.

Liu, V., Yao, H., and White, M. (2019). Toward understanding catastrophic interference in value-based reinforcement learning. *Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*.

Liu, Y., Starzyk, J. A., and Zhu, Z. (2008). Optimized approximation algorithm in neural networks without overfitting. *IEEE transactions on neural networks*, 19(6):983–995.

Lodwich, A., Rangoni, Y., and Breuel, T. (2009). Evaluation of robustness and performance of early stopping rules with multi layer perceptrons. In *2009 international joint conference on Neural Networks*, pages 1877–1884. IEEE.

Lopes, V., Alirezazadeh, S., and Alexandre, L. A. (2021). Epe-nas: Efficient performance estimation without training for neural architecture search. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*, pages 552–563. Springer.

Loshchilov, I. and Hutter, F. (2015). Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Lu, Y. and He, W. (2022). Selc: Self-ensemble label correction improves learning with noisy labels. *arXiv preprint arXiv:2205.01156*.

Lyle, C., Schut, L., Ru, B., Gal, Y., and van der Wilk, M. (2020). A bayesian perspective on training speed and model selection. *arXiv preprint arXiv:2010.14499*.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131.

Ma, X., Huang, H., Wang, Y., Romano, S., Erfani, S., and Bailey, J. (2020). Normalized loss functions for deep learning with noisy labels. In *ICML*.

Maennel, H., Alabdulmohsin, I., Tolstikhin, I., Baldock, R. J., Bousquet, O., Gelly, S., and Keysers, D. (2020). What do neural networks learn when trained with random labels? *arXiv preprint arXiv:2006.10455*.

Mahsereci, M., Balles, L., Lassner, C., and Hennig, P. (2017). Early stopping without a validation set. *arXiv preprint arXiv:1703.09580*.

Malach, E. and Shalev-Shwartz, S. (2017). Decoupling" when to update" from" how to update". *Advances in neural information processing systems*, 30.

McAllester, D. (2003). Simplified pac-bayesian margin bounds. In *Learning theory and Kernel machines*, pages 203–215. Springer.

McAllester, D. A. (1999a). Pac-bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 164–170.

McAllester, D. A. (1999b). Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

Mehta, H., Cutkosky, A., and Neyshabur, B. (2020). Extreme memorization via scale of initialization. *arXiv preprint arXiv:2008.13363*.

Mehta, P., Bukov, M., Wang, C.-H., Day, A. G., Richardson, C., Fisher, C. K., and Schwab, D. J. (2019). A high-bias, low-variance introduction to machine learning for physicists. *Physics reports*.

Mellor, J., Turner, J., Storkey, A., and Crowley, E. J. (2021). Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR.

Mhaskar, H., Liao, Q., and Poggio, T. (2017). When and why are deep networks better than shallow ones? In *Thirty-First AAAI Conference on Artificial Intelligence*.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932.

Nagarajan, V. and Kolter, J. Z. (2019). Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 11615–11626.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003.

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., and Mitliagkas, I. (2018). A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*.

Negrea, J., Haghifam, M., Dziugaite, G. K., Khisti, A., and Roy, D. M. (2019). Information-theoretic generalization bounds for sgld via data-dependent estimates. In *Advances in Neural Information Processing Systems*, pages 11013–11023.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017a). Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2017b). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*.

Neyshabur, B., Sedghi, H., and Zhang, C. (2020). What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523.

Ng, A. (2012). Learning Thoery Lecture Notes. http://cs229.stanford.edu/notes/cs229-notes4.pdf.

Nguyen, D. T., Mummadi, C. K., Ngo, T. P. N., Nguyen, T. H. P., Beggel, L., and Brox, T. (2019). Self: Learning to filter noisy labels with self-ensembling. *arXiv preprint arXiv:1910.01842*.

Nigam, N., Dutta, T., and Gupta, H. P. (2020). Impact of noisy labels in learning techniques: a survey. In *Advances in data and information sciences*, pages 403–411. Springer.

NNGC (2008). Nn5 time series forecasting competition for neural networks. http://www.neural-forecasting-competition.com/NN5.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*.

Papernot, N., McDaniel, P., and Goodfellow, I. (2016a). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016b). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Patel, D. and Sastry, P. (2021). Memorization in deep neural networks: Does the loss function matter? In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 131–142. Springer.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. (2017). Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952.

Pedraza, A., Deniz, O., and Bueno, G. (2021). On the relationship between generalization and robustness to adversarial examples. *Symmetry*, 13(5):817.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Philipp, G. and Carbonell, J. G. (2018). The nonlinearity coefficient-predicting overfitting in deep neural networks. *arXiv preprint arXiv:1806.00179*.

Piche, S. W. (1995). The selection of weight accuracies for madalines. *IEEE Transactions on Neural Networks*, 6(2):432–445.

Piotrowski, A. P. and Napiorkowski, J. J. (2013). A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. *Journal of Hydrology*, 476:97–111.

Pleiss, G., Zhang, T., Elenberg, E., and Weinberger, K. Q. (2020a). Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056.

Pleiss, G., Zhang, T., Elenberg, E. R., and Weinberger, K. Q. (2020b). Detecting noisy training data with loss curves.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Pulastya, V., Nuti, G., Atri, Y. K., and Chakraborty, T. (2021). Assessing the quality of the datasets by identifying mislabeled samples. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 18–22.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Qian, X. and Klabjan, D. (2020). The impact of the mini-batch size on the variance of gradients in stochastic gradient descent. *arXiv preprint arXiv:2004.13146*.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. (2020). Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. (2019). On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR.

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449.

Razzak, M. I., Naz, S., and Zaib, A. (2018). Deep learning for medical image processing: Overview, challenges and the future. *Classification in BioApps: Automation of Decision Making*, pages 323–350.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*.

Roh, Y., Heo, G., and Whang, S. E. (2019). A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*.

Rolnick, D., Veit, A., Belongie, S., and Shavit, N. (2017). Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Ru, R., Lyle, C., Schut, L., Fil, M., van der Wilk, M., and Gal, Y. (2021). Speedy performance estimation for neural architecture search. *Advances in Neural Information Processing Systems*, 34.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., and Goldstein, T. (2019). The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. *arXiv preprint arXiv:1904.06963*.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shifat-E-Rabbi, M., Yin, X., Fitzgerald, C. E., and Rohde, G. K. (2020). Cell image classification: a comparative overview. *Cytometry Part A*, 97(4):347–362.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sokolić, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. (2017). Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280.

Song, H., Kim, M., and Lee, J.-G. (2019). SELFIE: Refurbishing unclean samples for robust deep learning. In *ICML*.

Song, H., Kim, M., Park, D., and Lee, J.-G. (2020a). Prestopping: How does early stopping help generalization against label noise?

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. (2022). Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Song, J., Dauphin, Y., Auli, M., and Ma, T. (2020b). Robust and on-the-fly dataset denoising for image classification. In *European Conference on Computer Vision*, pages 556–572. Springer.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. (2018). The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Štěpnička, M. and Burda, M. (2016). Computational intelligence in forecasting (cif). https://irafm.osu.cz/cif.

Stone, J. V. (2019). *Artificial intelligence engines: a tutorial introduction to the mathematics of deep learning*. Sebtel Press.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133.

Stutz, D., Hein, M., and Schiele, B. (2019). Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987.

Sun, Z., Hua, X.-S., Yao, Y., Wei, X.-S., Hu, G., and Zhang, J. (2020). Crssc: salvage reusable samples from noisy data for robust learning. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 92–101.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. (2020). Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR.

Tartaglione, E., Lepsøy, S., Fiandrotti, A., and Francini, G. (2018). Learning sparse neural networks via sensitivity-driven regularization. In *Advances in Neural Information Processing Systems*, pages 3878–3888.

Telgarsky, M. (2016). Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*.

Thulasidasan, S., Bhattacharya, T., Bilmes, J., Chennupati, G., and Mohd-Yusof, J. (2019). Combating label noise in deep learning using abstention. *arXiv preprint arXiv:1905.10964*.

Tibshirani, R. (1996). *Bias, variance and prediction error for classification rules*. Citeseer.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*.

Van Horn, G. and Perona, P. (2017). The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*.

Wang, T. E., Gu, J., Mehta, D., Zhao, X., and Bernal, E. A. (2018). Towards robust deep neural networks. *arXiv preprint arXiv:1810.11726*.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. (2019). Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 322–330.

Wang, Y., Peng, T., Duan, J., Zhu, C., Liu, J., Ye, J., and Jin, M. (2020). Pathological image classification based on hard example guided cnn. *IEEE Access*, 8:114249–114258.

Wei, J., Zhu, Z., Cheng, H., Liu, T., Niu, G., and Liu, Y. (2021). Learning with noisy labels revisited: A study using real-world human annotations. *arXiv preprint arXiv:2110.12088*.

Wei, J., Zhu, Z., Cheng, H., Liu, T., Niu, G., and Liu, Y. (2022). Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*.

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., and Feris, R. (2018). Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8817–8826.

Wu, Z., Shen, C., and Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133.

Xia, X., Liu, T., Han, B., Gong, C., Wang, N., Ge, Z., and Chang, Y. (2021). Robust early-learning: Hindering the memorization of noisy labels. In *International Conference on Learning Representations*.

Xia, X., Liu, T., Han, B., Wang, N., Gong, M., Liu, H., Niu, G., Tao, D., and Sugiyama, M. (2020). Part-dependent label noise: Towards instance-dependent label noise. *Advances in Neural Information Processing Systems*, 33:7597–7610.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. (2015). Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Xu, Z.-Q. J., Zhang, Y., Luo, T., Xiao, Y., and Ma, Z. (2019a). Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*.

Xu, Z.-Q. J., Zhang, Y., and Xiao, Y. (2019b). Training behavior of deep neural network in frequency domain. In *International Conference on Neural Information Processing*, pages 264–274. Springer.

Yang, J., Zeng, X., Zhong, S., and Wu, S. (2013). Effective neural network ensemble approach for improving generalization performance. *IEEE transactions on neural networks and learning systems*, 24(6):878–887.

Yang, Y.-Y., Rashtchian, C., Zhang, H., Salakhutdinov, R., and Chaudhuri, K. (2020). A closer look at accuracy vs. robustness. *arXiv preprint arXiv:2003.02460*.

Yao, Y., Rosasco, L., and Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315.

Yin, D., Kannan, R., and Bartlett, P. (2019). Rademacher complexity for adversarially robust generalization. In *International conference on machine learning*, pages 7085–7094. PMLR.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. (2017). Gradient diversity: a key ingredient for scalable distributed learning. *arXiv preprint arXiv:1706.05699*.

Yu, F., Wang, D., Shelhamer, E., and Darrell, T. (2018). Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. (2019). How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173. PMLR.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zeng, X. and Yeung, D. S. (2001). Sensitivity analysis of multilayer perceptron to input and weight perturbations. *IEEE Transactions on Neural Networks*, 12(6):1358–1366.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016a). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021a). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Zhang, C., Bengio, S., and Singer, Y. (2019a). Are all layers created equal? *arXiv preprint arXiv:1902.01996*.

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

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. (2019b). Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR.

Zhang, J., Barr, E., Guedj, B., Harman, M., and Shawe-Taylor, J. (2019c). Perturbed model validation: A new framework to validate model relevance.

Zhang, J., Wu, X., and Sheng, V. S. (2016b). Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576.

Zhang, X., Hou, P., Zhang, X., and Sun, J. (2021b). Neural architecture search with random labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10907–10916.

Zhang, X., Wu, D., Xiong, H., and Dai, B. (2021c). Optimization variance: Exploring generalization properties of {dnn}s.

Zhang, X., Xiong, H., and Wu, D. (2020a). Rethink the connections among generalization, memorization and the spectral bias of dnns. *arXiv preprint arXiv:2004.13954*.

Zhang, Y., Lu, Y., Han, B., Cheung, Y.-m., and Wang, H. (2022). Combating noisy-labeled and imbalanced data by two stage bi-dimensional sample selection. *arXiv preprint arXiv:2208.09833*.

Zhang, Z., Zhang, H., Arik, S. O., Lee, H., and Pfister, T. (2020b). Distilling effective supervision from severe label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9294–9303.

Zhu, C., Chen, W., Peng, T., Wang, Y., and Jin, M. (2021a). Hard sample aware noise robust learning for histopathology image classification. *IEEE Transactions on Medical Imaging*, 41(4):881–894.

Zhu, Z., Dong, Z., Cheng, H., and Liu, Y. (2021b). A good representation detects noisy labels. *arXiv preprint arXiv:2110.06283*.

Ziyin, L., Chen, B., Wang, R., Liang, P. P., Salakhutdinov, R., Morency, L.-P., and Ueda, M. (2020). Learning not to learn in the presence of noisy labels. *arXiv preprint arXiv:2002.06541*.

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Mahsa Forouzesh

Departement of Computer and Communication Sciences
Ecole Polytechnique Federale De Lausanne (EPFL)
Mobile: +41787181640
Email: mahsa.forouzesh93@gmail.com
Personal Website

Research Interests

- Deep Learning
- Generalization in Neural Networks
- Supervised Learning with Limited and Noisy Data

Education

Sept.2017-present **PhD in Computer and Communication sciences**, EPFL, Lausanne, Switzerland
Information and Network Dynamics Group, Advisor: Prof. Patrick Thiran
Thesis Title: ***Deep-Learning Generalization with Limited and Noisy Labels***

Selected Courses:

- Machine Learning (Fall 2017)
- Optimization for Machine Learning (Spring 2018)
- Deep Learning for Natural Language Processing (Fall 2019)
- Distributed Information Systems (Fall 2020)

2012-2016 **B.Sc. in Electrical Enginnering**
Telecommunication Engineering Minor, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

- GPA: 18.22/20 (3.82/4) — GPA of last 2 years: 18.87/20 (3.96/4)

Experiences/Services

2022 **Software engineering internship** at Google, Zurich, Switzerland;
Building offensive image classifiers using image embeddings and OCRs resulting in a headroom reduction of **38.8**%.

2020-2022 **Supervised** the semester projects "Implementing Variants of MeProp", "NAS without Training using Sensitivity", and "Early Stopping for Time-series Applications"

2021-23 Served as a **reviewer** at NeurIPS 2022, ICLR 2022 (selected as a highlighted reviewer), ICML 2022, and ICML 2023

2018-22 **Teaching assistant** for the courses Machine learning with over 500 students, Dynamical system theory for engineers, Probability and statistics, Stochastic models in communication, and Mise à niveau (MAN)

2022-present Served as a volunteer **counsellor** at EPFL Peer2Peer counselling center

2020 Volunteered at ICLR 2020, ICML 2020 and NeurIPS 2020 conferences

Publications

Leveraging Unlabeled Data to Track Memorization (ICLR 2023)
Mahsa Forouzesh, Hanie Sedghi, Patrick Thiran

Disparity Between Batches as a Signal for Early Stopping (ECML/PKDD 2021)
Mahsa Forouzesh, Patrick Thiran

Generalization Comparison of Deep Neural Networks via Output Sensitivity (Oral Presentation at ICPR 2020)
Mahsa Forouzesh, Farnood Salehi, Patrick Thiran

Honors and Awards

Sept. 2017 Fellowship Program in Computer and Communication Sciences, EPFL, Switzerland

2016 M.Sc. Admission from ECE Department without entrance exam, University of Tehran, Tehran, Iran

2016 Ranked 11th among 150 EE students in B.Sc., University of Tehran, Tehran, Iran

2014-2015 FOE (Faculty of Engineering) Award as the 3rd ranked student, awarded by Department of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

2012 Ranked 42nd among more than 300,000 participants in the nationwide university entrance exam (Konkor) and membership to INEF-Iran National Elites Foundation

Selected Projects

Fall 2019 **Proposed an extension of the CMOW (continuous multiplication of words) model for Sentence embedding**
Modified the optimization procedure of the CBOW-CMOW hybrid model by applying an explore-exploit algorithm using Pytorch

Spring 2018 **Implemented MeProp, a sparsified back propagation algorithm, using Pytorch**

Fall 2017 **Designed and implemented a recommender system using Tensorflow**

Fall 2017 **Implemented the elastic averaging SGD algorithm using Tensorflow**

Computer and Technical Skills

Python, Pytorch, Tensorflow, Pandas, MATLAB, C++, C, SQL, Flume, Verilog, Assembly, Qt, SIMULINK, Multisim, HSpice, NS2, Arduino, Wireshark