Adapting to Structure and Using Structure to Adapt: Toward Explaining the Success of Modern Deep Learning

Stefani Karp

September 2024
CMU-ML-24-111

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Aarti Singh, Chair
Yuanzhi Li
Satyen Kale (Google Research)
Robert Nowak (University of Wisconsin-Madison)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported in part by Google.

Keywords: machine learning, deep learning, optimization, generalization, algorithmic stability, feature learning, kernel methods, convolutional neural networks, Transformers, language models

Abstract

This thesis studies the remarkable success of deep learning. It offers the perspective that, rather than developing black-box generalization bounds, one particularly fruitful way to understand the success of modern deep learning is through the careful interplay between neural networks' flexibility and structure in specific domains. In these domains, we can understand modern deep learning through its ability to (1) adapt to structure in data and (2) use its structures (architecture, pretrained initialization, etc.) to adapt. We build this perspective through a mix of theory and empirics. We begin by looking at traditional learning theory tools: generalization bounds. Specifically, we study algorithmic stability as a possible framework for explaining the performance of gradient descent in overparameterized neural networks. We provide empirical evidence that uniform stability does not appear with sufficient strength to explain the generalization performance of neural networks. Then, instead of focusing on taming deep learning's flexibility, we recast deep learning's flexibility as a powerful ability to adapt when just enough structure is present. In the remainder of the thesis, we carefully study three key settings - convolutional neural networks on image data, simple Transformers on basic algorithmic tasks, and pretrained language models on natural language data - that demonstrate the impressive ability of neural networks to adapt to structure in data and leverage their structures to quickly and flexibly adapt. Together, these three settings trace the evolution of training methods and paradigms over the past six years. Instead of the bleaker image painted by the more black-box approach to generalization that we began with, we use these settings to advocate for a more mechanistic and nuanced understanding of the interplay between neural networks' flexibility and structure in specific domains.

Acknowledgments

I would like to begin by thanking my advisor, Aarti Singh, for all her support throughout my PhD. From the very beginning, Aarti provided a safe, welcoming space to explore my research interests. She has listened thoughtfully to my perspective and ideas and encouraged me to trust myself and my judgment, even when I found it difficult. She has helped me grow as a researcher, at times by patiently digging into the details with me and at other times by helping me to take a step back, examine the bigger picture, and focus on what matters. And, regardless of the situation, she has always treated me with incredible kindness. I feel very grateful.

I would also like to thank my committee: Yuanzhi Li, Satyen Kale, and Rob Nowak. Yuanzhi Li has taught me so much about how to do meaningful research, particularly in the deep learning theory space. His intuition is really incredible, and I feel grateful to have learned from him throughout my PhD. Satyen Kale has been an amazing mentor at Google throughout these past few years. He has taught me a lot about optimization and generalization, both in classical settings and in deep learning. But beyond his technical wisdom, he has also been an especially caring mentor, providing thoughtful support throughout all aspects of the research process. It's hard to express how much this has meant. Rob Nowak has such a genuine love and curiosity for research, and all of my interactions with him have left me newly inspired. It has been such a gift to interact with him at various points throughout my PhD, and I'm really grateful for his participation in my committee.

I owe a big thanks to CMU SCS and especially the Machine Learning Department for so much support throughout my PhD. That begins with Diane Stidle, who makes magic happen within the department. Her help has been key on too many occasions to count. So many professors made CMU an amazing experience. Thank you for stimulating classes and the chance to teach too. I had really cherished TA experiences, and I owe a big thank you to the teaching teams and the students. Thank you to Mike Tarr for incredible generosity when I was exploring research questions at the start of my PhD and for welcoming me into group meetings. Thank you to Ryan Tibshirani for your leadership and support of the Wellness Network. Thank you to my peers, my classmates, and my wonderful officemates.

At Google, I owe a big thanks to the ML Theory team and especially to Mehryar Mohri, who took a chance on me as an early graduate student and invited me to join the team. I've learned so much from Mehryar, who's an incredible teacher with remarkably deep knowledge of so many aspects of machine learning theory. And to the rest of the ML Theory team - thank you! A special thanks to Pranjal Awasthi, for his thoughtful mentorship both in the deep learning theory space and more generally. Also, thanks to Phing Lee and Leslie Phillips for all your kindness and support, and thanks to the broader Google community, especially those in the NYC office.

I'm particularly grateful to all of my collaborators, who have made the research experience so much more enjoyable (listed here in chronological order, including mentors already mentioned): Behnam Neyshabur, Mehryar Mohri, Pranjal Awasthi,

Satyen Kale, Ezra Winston, Yuanzhi Li, Aarti Singh, Ziwei Ji, Kwangjun Ahn, Sashank Reddi, Sobhan Miryoosefi, Shankar Krishnan, Seungyeon Kim, Sanjiv Kumar, Kavya Ravichandran, Yatin Dandi, Francesca Mignacco, Aakash Lahoti, and Nikunj Saunshi.

I'm also grateful for the opportunities I've had to participate in intellectually-enriching programs beyond CMU and Google throughout my PhD, especially the Foundations of Deep Learning program at the Simons Institute in 2019 and the Les Houches summer school on Statistical Physics and Machine Learning in 2022. I'm really grateful to the organizers for creating such incredible programs and for inviting me to join. Through these programs, I met many collaborators and friends who deeply enriched my PhD experience - thank you!

Beyond PhD life, I feel immensely lucky to have so many wonderful friends who have provided support throughout this journey. I fear accidentally leaving someone out, so let me instead just say: thank you, all! You bring so much joy and meaning to, well, everything. You keep showing up and cheering me on, and I couldn't have done it without you.

Finally, a huge thank you to my family, including those who sadly didn't live long enough to be here today – my grandparents, uncle, and aunt. And to those I've been lucky enough to have throughout this journey, thank you for all your support these past six years, as always. Thank you, Aunt Donna. And thank you, Mom and Dad, for everything.

Contents

List of Figures

List of Tables

List of Algorithms

List of Theorems

Chapter 1

Introduction

Over the past decade, deep learning has transformed the computational landscape. In domains spanning computer vision, translation, natural language understanding, speech, robotics, biology, and many more, deep learning models have not only significantly outperformed prior machine learning techniques, but they have - in many cases - completely changed what seems possible. Deep learning models are the underlying technology behind many of the paradigm-shifting artificial intelligence applications that we see today, including the large-scale multimodal chatbots that have recently received considerable media attention.

However, it wasn't always obvious to the machine learning community that deep learning would one day see this remarkable fate. Researchers have been working on neural networks for decades (Rumelhart et al., 1986; LeCun and Bengio, 1995), often battling significant skepticism surrounding whether these models could ever truly work well. One common criticism was related to optimization. Neural networks are notorious for having nonconvex loss landscapes, which means that a local minimum of the training loss is not necessarily a global minimum, and thus little can be guaranteed about successful optimization in the worst case (Blum and Rivest, 1992; Bartlett and Ben-David, 2002). Moreover, since neural networks are often trained with many more parameters than data points (i.e., in the overparameterized regime), they seemed to lack the capacity control that had been the bedrock of classical machine learning theory (Vapnik and Chervonenkis, 1971; Bartlett and Mendelson, 2002). Therefore, even if global minima of the training loss *could* be found, it seemed difficult to expect these solutions to reliably perform well on unseen data. Nevertheless, neither of these obstacles has appeared as difficult in practice as theory suggested. Why?

This mystery of deep learning's success has attracted much attention within the machine learning theory community. Unlocking deep learning's secrets holds allure for multiple reasons. On the one hand, the failures of existing theory to explain the success of modern deep learning suggest that these theoretical frameworks have room to improve, which creates an excitement within the machine learning community akin to the search for physics beyond the standard model. Beyond this, understanding the mysteries of deep learning offers hope for designing even better systems and training algorithms. Working with deep learning models today has often been described as tinkering with a black box, and any progress made toward more principled

techniques promises to save researchers and engineers significant time and effort.

On the optimization side, various works over the years have been gradually developing the theory of optimization in the overparameterized regime. Specifically, one of the key insights was that, with sufficient width (i.e., number of neural network hidden units) in terms of the problem parameters, the neural network's parameters do not end up moving that much overall, which simplifies the dynamics of gradient descent (Du et al., 2019b). These initial results have been gradually improved over the years (Allen-Zhu et al., 2019b; Zou and Gu, 2019; Ji and Telgarsky, 2020; Zou et al., 2020; Chen et al., 2021).

On the generalization side, early work showed that, contrary to classical machine learning wisdom, test error does not increase with a neural network's size (i.e., number of hidden units), even after zero training error is achieved (Neyshabur et al., 2014). In response, the community asked: are certain reasonable notions of complexity naturally controlled by gradient descent when it is used to train deep neural networks? Or, what is gradient descent's "inductive bias"? Various norms and other complexity measures were explored, some with more success than others, but no single measure appeared to truly solve the puzzle (Neyshabur et al., 2017b; Bartlett et al., 2017a; Arora et al., 2018; Neyshabur et al., 2019; Long and Sedghi, 2020b). In fact, some norm-based measures were seen to correlate negatively with generalization performance (Jiang* et al., 2020), and Nagarajan and Kolter, 2019a even questioned and cast doubt on whether *any* uniform convergence technique (the focus of much of the aforementioned results) could explain deep learning's generalization performance, even after accounting for gradient descent's implicit bias.

Thus, at the time when this thesis started, there were far more questions than answers surrounding deep learning's remarkable performance.

1.1 Thesis Statement

This thesis therefore studies the question: Why do deep neural networks work so well?

Through a series of works completed over the past six years, we develop the following perspective:

Rather than developing black-box generalization bounds, one particularly fruitful way to understand the success of modern deep learning is through the careful interplay between neural networks' flexibility and structure in *specific domains*. In these domains, we can understand modern deep learning through its ability to (1) adapt *to* structure in data and (2) *use* its structures (architecture, pretrained initialization, etc.) to adapt.

We develop this perspective throughout this thesis as follows, employing a mix of theory and empirics.

We begin by looking at traditional learning theory tools: generalization bounds. As previously discussed, at the beginning of this thesis, traditional complexity-based generalization measures had been failing to adequately capture and explain the surprising generalization performance of overparameterized neural networks, and the search for "the right" complexity measure was in full swing. There was also the question of whether a traditional complexity-based approach would

ever prove adequate, or whether fundamentally new frameworks and tools would be needed. Much of the skepticism came from the role of gradient descent – it seemed like a full explanation should somehow account for the role of gradient descent specifically, but most traditional generalization bounds were not designed to take the optimizer into account. One notable exception to this was algorithmic stability (Bousquet and Elisseeff, 2002), a framework connecting an algorithm's generalization performance to its sensitivity to changes in the training data. Some theory researchers therefore wondered whether this could be the right framework to explain the remarkable performance of gradient descent in overparameterized neural networks. However, not much was known about whether this framework could really capture the performance of practical neural networks.

In On the Algorithmic Stability of SGD in Deep Learning (Chapter 2), we studied empirically whether neural networks demonstrate sufficient algorithmic stability to explain their generalization performance. We created the first comprehensive empirical study of uniform stability in deep learning and provided initial empirical evidence that uniform stability does not appear with sufficient strength to explain the generalization performance of neural networks. This work contributed to the growing sentiment within the theory community that traditional generalization bounds - even algorithm-dependent ones - were likely not the right toolkit for studying deep learning.

The failures of these generalization bounds stemmed from the capacity and flexibility of deep learning. Instead of trying so hard to tame its flexibility, could we instead focus on understanding how its flexibility contributed to its success? In the remaining chapters, this is the approach we take, recasting flexibility as a powerful ability to adapt when just enough structure is present. In Chapter 3, we study the ability of convolutional neural networks to adapt their filters to underlying structure in image data, amidst high degrees of background noise, and thus provably outperform corresponding kernel methods. We also present empirical evidence that this ability to adapt, which we call Local Signal Adaptivity, is a key separator between convolutional neural networks and their corresponding kernel methods. In Chapter 4, we study how the Transformer's flexible self-attention mechanism can adapt to an underlying decision-list-like structure, and we provide some of the first provable guarantees for softmax-based attention. Finally, in Chapter 5, we study how pretrained Transformer-based language models can grow in depth and subsequently adapt to their new parameters. The pretrained language models have sufficient structure to enable fast adaptation to the new Transformer blocks, which enables surprisingly early prediction of optimal growing strategies.

In this manner, through a mix of theory and empirics at different scales, we carefully study three key settings - convolutional neural networks on image data, simple Transformers on basic algorithmic tasks, and pretrained language models on natural language data - that demonstrate the impressive ability of neural networks to adapt to structure in data and leverage their structures (architecture, pretrained initializations) to quickly and flexibly adapt. Together, these three settings trace the evolution of training methods and paradigms over the past six years. Instead of the bleaker image painted by the more classical, black-box approach to generalization in Chapter 2, we use these settings to advocate for a more mechanistic and nuanced understanding of the interplay between neural networks' flexibility and structure in specific domains.

1.2 Overview of Chapters

Here, we provide a more detailed description of each chapter.

Chapter 2: Algorithmic Stability

This chapter is based on:

- Stefani Karp, Behnam Neyshabur, and Mehryar Mohri. On the Algorithmic Stability of SGD in Deep Learning. 2020.

In this chapter, we study *algorithmic stability* (Bousquet and Elisseeff, 2002) as a framework for explaining why gradient descent finds neural network parameters that generalize well. Other generalization bounds have failed to apply successfully to neural networks. Many such bounds either scale incorrectly with important problem parameters or are vacuous. In this work, we empirically analyze whether neural networks trained via gradient descent exhibit sufficient algorithmic stability to explain their generalization performance. Overall, we see a lack of stability empirically: just swapping one data point leads to quite different functions.

Where do we go from here? Propelled by the limitations of generalization bounds in explaining the success of gradient descent in modern deep learning, we consider alternative approaches. Specifically, the limitations of this black-box approach spur us to take a more mechanistic perspective. We look at three important threads, which together trace the evolution of training methods and paradigms over the past six years. Throughout, we focus on the role of structure: how neural networks adapt to structure in data and how their structures themselves enable this adaptation.

Chapter 3: Local Signal Adaptivity

This chapter is based on:

- Stefani Karp, Ezra Winston, Yuanzhi Li, and Aarti Singh. Local Signal Adaptivity: Provable Feature Learning in Neural Networks Beyond Kernels. In Advances in Neural Information Processing Systems, 2021.

In 2018, the Neural Tangent Kernel (Jacot et al., 2018b) was introduced as a possible explanation for the success of gradient-descent-trained neural networks. Specifically, this line of work showed that, under certain limiting conditions, neural networks trained via gradient descent reduce to kernel methods, where the specific kernel is determined by the neural network's architecture and its initialization. Mathematically, this perspective was incredibly appealing, as it permitted theoreticians to reuse the rich preexisting and mathematically rigorous literature on kernel methods to make mathematically-grounded predictions about modern-day neural networks. On the one hand, certain works showed impressive empirical similarity between the performance of neural networks and their corresponding kernel methods. However, on the other hand, these corresponding kernel methods could not consistently match the performance of gradient-descent-trained neural networks. And perhaps more importantly, something seemed

conceptually off: could it really be the case that features determined *before* seeing any training data at all were sufficient to explain why neural networks worked so well? Empirically, neural networks seemed to be using their training data to learn features. They seemed to be *adapting* their features to the data.

We explore this question throughout Chapter 3. Specifically, we study feature learning in convolutional neural networks. To do so, we introduce a toy model that is intended to capture what we conjecture to be a key property of natural image classification tasks: the presence of localized, label-determining features embedded within noisy backgrounds. In this simple model, we show that convolutional neural networks trained via gradient descent are able to adapt their filters to learn the underlying problem structure, whereas kernel methods succumb to the noise. We call this phenomenon Local Signal Adaptivity: the learned ability to find a small set of localized label-determining features embedded within a noisy background. Through theoretical and empirical analysis, we provide evidence that this adaptation is key to gradient-descent-trained convolutional neural networks' superiority over kernel methods.

Chapter 4: Transformer Theory

This chapter is based on:

- Stefani Karp, Pranjal Awasthi, and Satyen Kale. Provable Gradient-Descent-Based Learning of Decision Lists by Transformers. In DeepMath, 2023.

The Transformer architecture, introduced in 2017, significantly altered the landscape of natural language understanding and generation, as well as that of many other domains – ranging from speech to vision to biology to robotics. The Transformer architecture is suitable for any input data that can be represented as a sequence. Its key architectural component is self-attention, which allows each sequence element to dynamically "pay attention to" other sequence elements, enabling sophisticated computation across the sequence. The generality of this approach has been paradigm-shifting. Why is it the case that so many domains have fallen to the Transformer architecture? What is so powerful about the combination of sequence data, self-attention, and gradient descent?

Rigorously analyzing gradient descent dynamics for the Transformer architecture is notoriously difficult. The self-attention mechanism consists of a softmax applied over the input sequence, which significantly complicates the analysis. It is also far from obvious what assumptions one should make on the input data. What are the key interesting, common features of sequence data that have contributed to the Transformer's rise?

In Chapter 4, we explore this question through the design of a simple data distribution that significantly benefits from attention, and we provide a mathematically rigorous analysis of how gradient descent can optimize the parameters of a simple Transformer to solve this task. This provides one of the first examples of provably optimizing the softmax-based-attention using gradient descent. Our data distribution is similar to a decision list, with a few variations, and we put forth this simple data distribution as a basic example of the kind of domain-agnostic data structure to which self-attention's parameters can provably adapt.

Chapter 5: Landscape-Aware Growing

This chapter is based on:

- Stefani Karp*, Nikunj Saunshi*, Sobhan Miryoosefi, Sashank J. Reddi, and Sanjiv Kumar. Landscape-Aware Growing: The Power of a Little LAG. arXiv preprint arXiv:2406.02469, 2024,

where * denotes joint first authorship.

Language models have grown rapidly in size in recent years, and with this seeming explosion in model size, we have seen the concurrent rise of the "foundation model" paradigm: using and adapting large, pretrained models for various tasks. These pretrained models are adapted in myriad ways, ranging from supervised fine-tuning and various other fine-tuning techniques to model merging and growing. In this regime, the questions about gradient descent are slightly different. Rather than asking why gradient descent is able to optimize neural networks starting from random initializations, we ask how the structure contained in the pretrained network affects the dynamics and even predictability of gradient descent.

In this chapter, we explore this question by studying the growing of pretrained language models. Specifically, we study how to increase the depth of pretrained Transformer-based language models. We conduct an extensive empirical analysis of various ways to increase the depth, and we see that there is generally sufficient structure in the existing network to enable fast adaptation to the new parameters via gradient descent. As a result, unlike in standard neural architecture search, where predicting the optimal architectures early on is fairly difficult, and unlike in the growing literature thus far, where the common wisdom has been to grow in a loss-preserving manner, we show that it is possible to identify near-optimal growing strategies after just a small amount of gradient-based training. In doing so, we highlight how the structure contained in pretrained weights enables efficient adaptation.

Chapter 2

Algorithmic Stability

This chapter is based on Karp et al., 2020:
Stefani Karp, Behnam Neyshabur, and Mehryar Mohri. On the Algorithmic Stability of SGD in Deep Learning. 2020.

2.1 Introduction

Despite the impressive empirical success of deep learning models, their ability to generalize well (on a significant set of data distributions) despite overparameterization has largely eluded the research community (Zhang et al., 2017; Neyshabur et al., 2017a). Various flavors of generalization bounds have been applied to neural networks, including various norm- and margin-based bounds (Bartlett et al., 2017b; Neyshabur et al., 2015; Elsayed et al., 2018; Liang et al., 2019; Long and Sedghi, 2020a), PAC-Bayes bounds (Dziugaite and Roy, 2017; Neyshabur et al., 2018), and VC-dimension-based bounds (Bartlett et al., 2019). However, many such bounds have been shown to be insufficiently-correlated with generalization as various model components are varied (e.g., number of parameters) (Jiang et al., 2020). Nagarajan and Kolter (2019b) demonstrated that some of these bounds can even increase with sample size in certain settings, underscoring the importance of *empirically* evaluating proposed bounds' behavior as a function of dataset size. Furthermore, Nagarajan and Kolter (2019b) suggest that all uniform-convergence-based approaches might inherently be unable to explain deep learning's generalization performance, even *after* uniform convergence is restricted to the smallest possible set of models determined by the implicit bias of the learning algorithm. If this is true, then what tools for proving deep learning generalization bounds remain? One such tool, as acknowledged by Nagarajan and Kolter (2019b), is *algorithmic stability*.

Algorithmic stability. Algorithmic stability typically refers to a sensitivity analysis of the algorithm itself; specifically, how much can swapping (or removing) one point in an m-item training set S change the output of an algorithm $\mathcal{A}(S)$? Bousquet and Elisseeff (2002) formalized and proved generalization bounds under various different flavors of algorithmic stability; since then, additional variants of algorithmic stability have been developed (Abou-Moustafa and

Szepesvári, 2019; Foster et al., 2019; Kutin and Niyogi, 2002; Liu et al., 2017). However, to this day, the main variant for obtaining bounds that hold with high probability over the random draw of the training set is *uniform stability*, the strictest of the requirements. Specifically, a learning algorithm $\mathcal{A}$ is called β- uniformly stable with respect to loss ℓ if:

$$\forall S \in \mathcal{Z}^m, \forall i \in \{1, \ldots, m\}, \forall z \in \mathcal{Z} : |\ell(\mathcal{A}(S), z) - \ell(\mathcal{A}(S^{\setminus i}), z)| \leq \beta,$$

where $S^{\setminus i}$ is S with element i removed. Often, uniform stability is expressed with respect to the *swapping* of one point, instead of the *removal*:

$$\forall S, S' \in \mathcal{Z}^m, \forall z \in \mathcal{Z} : |\ell(\mathcal{A}(S), z) - \ell(\mathcal{A}(S'), z)| \leq \beta,$$

where S and S' only differ at one index.

Algorithmic stability of stochastic gradient descent (SGD). Various works thus far have studied whether the framework of algorithmic stability can be applied to the analysis of stochastic gradient descent (typically including at least some extension to nonconvex loss landscapes) (Hardt et al., 2016; Feldman and Vondrak, 2019; Kuzborskij and Lampert, 2018). However, each of these results has some *subset* of the following weaknesses when applied to practical deep learning:

- The bound is only in *expectation* with respect to the draw of the sample S. In general, we ultimately seek bounds that will hold with high probability over the draw of the sample $S \sim \mathcal{D}^m$, although such bounds are generally more difficult to prove theoretically.
- The stability parameter β relies on smoothness parameters of the loss landscape that might not be particularly favorable for neural networks.
- The result heavily relies on a learning rate of $\mathcal{O}(1/t)$, where t is the parameter update (vs. epoch). This ensures that, in expectation over the algorithm's randomness, the learning rate has decayed more for larger samples by the time the swapped point is encountered. In contrast, in deep learning, the learning rate typically stays constant for at least the first epoch.
- The proof relies on controlling the (expected) distance between $\mathcal{A}(S)$ and $\mathcal{A}(S')$ in parameter space, which seems unlikely to decrease sufficiently with dataset size in practice (without the aforementioned $1/t$ learning rate schedule). We explore this in more detail in Section 2.4.

Our work. Inspired by the growing literature empirically analyzing the shortcomings of current deep learning generalization bounds and the anticipated algorithmic stability weaknesses discussed above, in this work we initiate a study of the following question: Does SGD empirically satisfy uniform stability in *practical* deep learning settings, in a manner sufficient to yield generalization bounds that hold with high probability (over the draw of the dataset and the algorithm's randomness)? Unfortunately, analyzing uniform stability *empirically* is incredibly challenging due to the many suprema in the definition (i.e., $\forall S, S', z$), and we thus do not claim that any empirical analysis can *definitively* answer whether or not SGD in deep learning is uniformly stable. However, to our knowledge, this is the most *extensive* empirical examination of uniform stability in deep learning to date. Our contributions are as follows:

- Discussion of challenges in the empirical evaluation of uniform stability, with suggested methodology for overcoming them. Crucially, we validate our methodology in the simpler setting of logistic regression.
- Evidence that uniform stability (with respect to the cross-entropy loss) does not decrease sufficiently with dataset size to fully explain generalization in deep learning.
- Evidence that $\|\mathcal{A}(S) - \mathcal{A}(S')\|_2$ (when the output of $\mathcal{A}$ is treated as a single vector of concatenated parameters) does not sufficiently decrease with dataset size in practical deep learning settings; in some cases, it can even increase despite strong generalization. We suggest that, if there is a form of algorithmic stability at play in deep learning, it does not stem from parameter closeness. We argue that any future theoretical attempts to prove stability of SGD in deep learning should proceed through a different key path.
- Discovery of settings with insufficient cross-entropy uniform stability to explain generalization but for which $\mathcal{A}(S)$ and $\mathcal{A}(S')$ are in the same basin of attraction (see Section 2.4 for a precise definition), suggesting that convex settings with large basins of attraction could also share these same failure modes and thus pave the way for more tractable analyses.

2.2 Methodology

Here, we describe the key aspects of our methodology for empirically evaluating uniform stability, with additional details in Appendix A.1. We first applied our methodology to logistic regression, which we used to help validate our methodology. We then applied our methodology to deep neural networks.

Throughout the paper, we use $\mathcal{A}(S)$ to denote to the output of the algorithm on dataset S. Although this object is really a function, we slightly abuse notation and treat it as a vector, i.e., with all of the model's parameters concatenated into a single vector. We occasionally use W_S instead to denote the parameters output by $\mathcal{A}$ on S, concatenated into a single vector.

Random seeds. Since we are seeking bounds that hold with high probability over the randomness of the algorithm, each plot we produce examines a *single* setting of the seed controlling initialization and the seed controlling SGD order. Thus, for each dataset/hyperparameter configuration, we present a single setting of the seeds in the main paper and defer our plots for other seeds to Appendix A.2.

Datasets: S and S'. We used MNIST for logistic regression (divided into two classes for binary classification: labels 0-4 and labels 5-9), and we used CIFAR-10 (Krizhevsky, 2009a) and SVHN (Street View House Numbers) (Netzer et al., 2011) for neural network training (10-class classification). In order to thoroughly study behavior (e.g., test/train error, various stability metrics, etc.) as a function of dataset size, we examined the following dataset sizes:

- $\{800, 1600, 3200, 6400, 12800\}$ for logistic regression, and
- $\{15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000\}$ for neural networks.

See Appendix A.1 for more details regarding this choice of dataset sizes.

We emphasize that we intentionally do *not* use data augmentation; we want to precisely measure behavior as a function of dataset size, and to our knowledge, there is no widely-accepted approach for calculating the *effective* dataset size with data augmentation.

Multiple trials per dataset size. To study the quantifier $\forall S, S' \in \mathcal{Z}^m$ *empirically* as thoroughly as possible, we sampled multiple (S, S') pairs per dataset size m. Each (S, S') pair was sampled as follows: we first randomly drew a subset of size m from the relevant training dataset (uniformly at random without replacement) to form S, we then uniformly sampled a single element of the relevant test set (call this element z), and finally we uniformly sampled an index $i \in \{1, \ldots, m\}$ of S in which to swap in z, thus forming S'. We then trained two models in parallel, one on training dataset S and one on S'. This procedure was repeated 90 times per dataset size for logistic regression and 40 times per dataset size for each neural network configuration (due to the higher cost of each run).

Crucially, the only difference between training on S and S' was the appearance of z in S' in a single batch per epoch. All other data points were the same and were visited in the same order. Furthermore, we explicitly disabled all sources of GPU nondeterminism to ensure that we were fully isolating the effect of swapping in z.

Models and training. The logistic regression model is a 784-dimensional linear classifier plus a bias term, and the neural networks are residual networks, specifically ResNet-20 (He et al., 2016).

The logistic regression models were trained via stochastic gradient descent (SGD) with learning rate 0.1, batch size 128, and no momentum.

On SVHN, we trained a ResNet-20 via SGD with learning rate 0.01, batch size 32, and no momentum. On CIFAR-10, we explored two different hyperparameter configurations: one without momentum and one with momentum 0.9. The other hyperparameters were the same across both configurations: a decaying learning rate schedule (starting at 0.1 and dividing by 10 at iterations 32,000 and 48,000) and batch size 128 (He et al., 2016).

Stopping criterion. We train each model for 100,000 iterations (i.e., parameter updates). See Appendix A.1 for a more detailed discussion of stopping criteria.

Uniform stability with respect to the cross-entropy loss. In the uniform stability definition, instead of a supremum over the domain, we calculate a max over the test set. A priori, it might not be clear how effective this would be, and we thus validate our methodology via logistic regression in Section 2.3 before proceeding to deep learning.

Plots and curve fitting. Many of the quantities examined in this paper take the form of $g(m) = \sup_{S \in \mathcal{Z}^m} f(S)$ or $g(m) = \sup_{S,S' \in \mathcal{Z}^m} f(S, S')$ for some function f, and we expect $g(m)$ to have the form $g(m) = am^b$ for some constants a, b. Thus, for these quantities, we use the following plotting motif: all trials per dataset size are displayed as blue dots, the maximum value per dataset size is a red dot, and a green curve of the form $g(m) = am^b$ is fit to the *red* dots (see Appendix A.1 for curve fitting details). To emphasize b, the rate of decrease (or occasionally increase) with m, these plots display both the x- and y-axes in log scale.

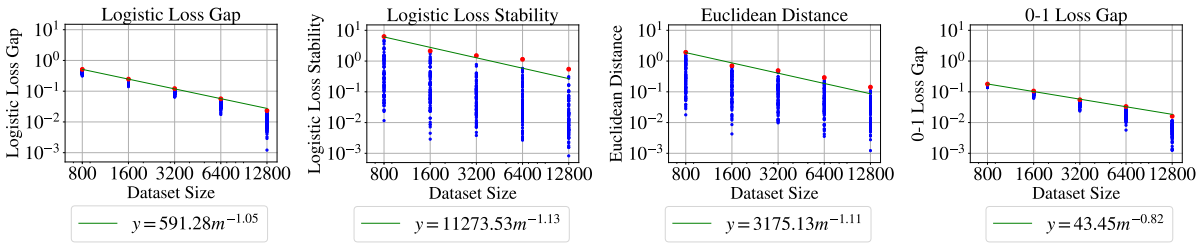


Figure 2.1: Main logistic regression results. From left to right: logistic loss generalization gap as a function of dataset size, logistic loss stability as a function of dataset size, Euclidean distance between parameters found using S vs. S' as a function of dataset size, and 0-1 loss generalization gap as a function of dataset size. There are 90 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, the logistic loss stability has a dependence on m comparable to that of the logistic loss generalization gap.

2.3 Uniform Stability and Generalization

For many years, obtaining useful generalization bounds via uniform stability required $\beta = \mathcal{O}(1/m)$, but Feldman and Vondrak (2019) (followed by Bousquet et al. (2019)) recently derived tighter bounds of the form: with probability at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$,

$$R_{\mathcal{D}}(\mathcal{A}(S)) \leq \widehat{R}_S(\mathcal{A}(S)) + c\left(\beta \log(m) \log(m/\delta) + \frac{\sqrt{\log(1/\delta)}}{\sqrt{m}}\right) \tag{2.1}$$

for some constant c. Here, $R_{\mathcal{D}}(\mathcal{A}(S))$ is the expected loss over the true distribution, and $\widehat{R}_S(\mathcal{A}(S))$ is the empirical loss evaluated on S. This bound suggests that $R_{\mathcal{D}}(\mathcal{A}(S)) - \widehat{R}_S(\mathcal{A}(S))$ is bounded by $\tilde{\mathcal{O}}(\max\{\beta, 1/\sqrt{m}\})$, hiding logarithmic dependencies inside the $\tilde{\mathcal{O}}$. Thus, if $R_{\mathcal{D}}(\mathcal{A}(S)) - \widehat{R}_S(\mathcal{A}(S))$ empirically decays more slowly than $1/\sqrt{m}$, providing empirical evidence that $R_{\mathcal{D}}(\mathcal{A}(S)) - \widehat{R}_S(\mathcal{A}(S))$ and β decay *similarly* with m would suggest that uniform stability has sufficient strength to explain generalization.

In this section, we present the results of our uniform stability experiments for both logistic regression and neural networks. In both sections, we also carefully estimate $R(\mathcal{A}(S)) - \widehat{R}_S(\mathcal{A}(S))$ as a function of dataset size, under both the 0-1 loss and the logistic or cross-entropy loss, to understand to what degree our uniform stability results are able to capture the strength of generalization. For convenience, we use the phrase "generalization gap" or "loss gap" to denote this difference in test and train loss.

2.3.1 Logistic regression

As there are obvious challenges in the empirical investigation of uniform stability with respect to the cross-entropy loss, we began by analyzing logistic regression, which presents many of the

ID	Model	Dataset	Learning rate	Batch size	Momentum
1a	ResNet-20	SVHN	0.01 (constant)	32	0.0
2a	ResNet-20	CIFAR-10	0.1, 0.01 at 32k, 0.001 at 48k	128	0.0
2b	ResNet-20	CIFAR-10	0.1, 0.01 at 32k, 0.001 at 48k	128	0.9

Table 2.1: Details of the 3 deep neural network settings studied.

same challenges (e.g., the logistic loss, how to analyze the suprema over the domain, etc.) but provides a much simpler and better-understood testbed in which to explore our methodology.

Results. In Figure 2.1, we plot the logistic loss generalization gap, our empirical estimate of the logistic loss uniform stability, the Euclidean distance between the final parameters of $\mathcal{A}(S)$ and $\mathcal{A}(S')$, and the 0-1 loss generalization gap. We fit a curve to the maximum value per dataset size, as described in detail in Section 2.2 and Appendix A.1, and we compare the dependence on m of our curves. Among the first three plots, we see a very similar dependence on m, ranging from $m^{-1.05}$ to $m^{-1.13}$. The dependence on m in the 0-1 loss generalization gap plot is a bit weaker ($m^{-0.82}$), but we include this primarily for completeness and as a frame of reference; we are more interested in whether *logistic loss* stability can explain the strength (with respect to m) of generalization with respect to the *logistic loss*.

Conclusions. These plots demonstrate the potential of our methodology to capture, via uniform stability with a *finite* maximum over the test set, the dependence on m of the Euclidean distance between parameters and, most importantly, the logistic loss generalization gap. Thus, although there are obvious differences between the suprema in the definition of uniform stability and our empirical evaluation with finite maxima, our results suggest that there is nevertheless some promise of obtaining informative empirical results.

2.3.2 Deep learning

After validating our methodology in the simpler setting of logistic regression, we now extend our methodology to the three deep learning configurations described in Section 2.2.

Results. Figure 2.2 displays the generalization and stability results for our three neural network settings. In contrast with logistic regression, we postpone examining the parameters themselves until Section 2.4, in which we conduct an analysis more targeted to deep learning's nonconvex loss landscape.

Most significantly, we compare the cross-entropy loss generalization gap to the uniform stability curve. For the ResNet-20 on SVHN, the stability curve displays a mild decrease with m (specifically, $m^{-0.09}$), compared to $m^{-0.18}$ for the cross-entropy loss generalization gap. For the ResNet-20 on CIFAR-10 *without* momentum, the stability curve does *not* decrease with m, despite the cross-entropy loss generalization gap having a dependence of $m^{-0.48}$. For the ResNet-20 on CIFAR-10 *with* momentum, the stability curve displays a mild decrease with m (specifically, $m^{-0.17}$), compared to $m^{-0.35}$ for the cross-entropy loss generalization gap.

To provide a frame of reference, we also compare the cross-entropy loss generalization gap

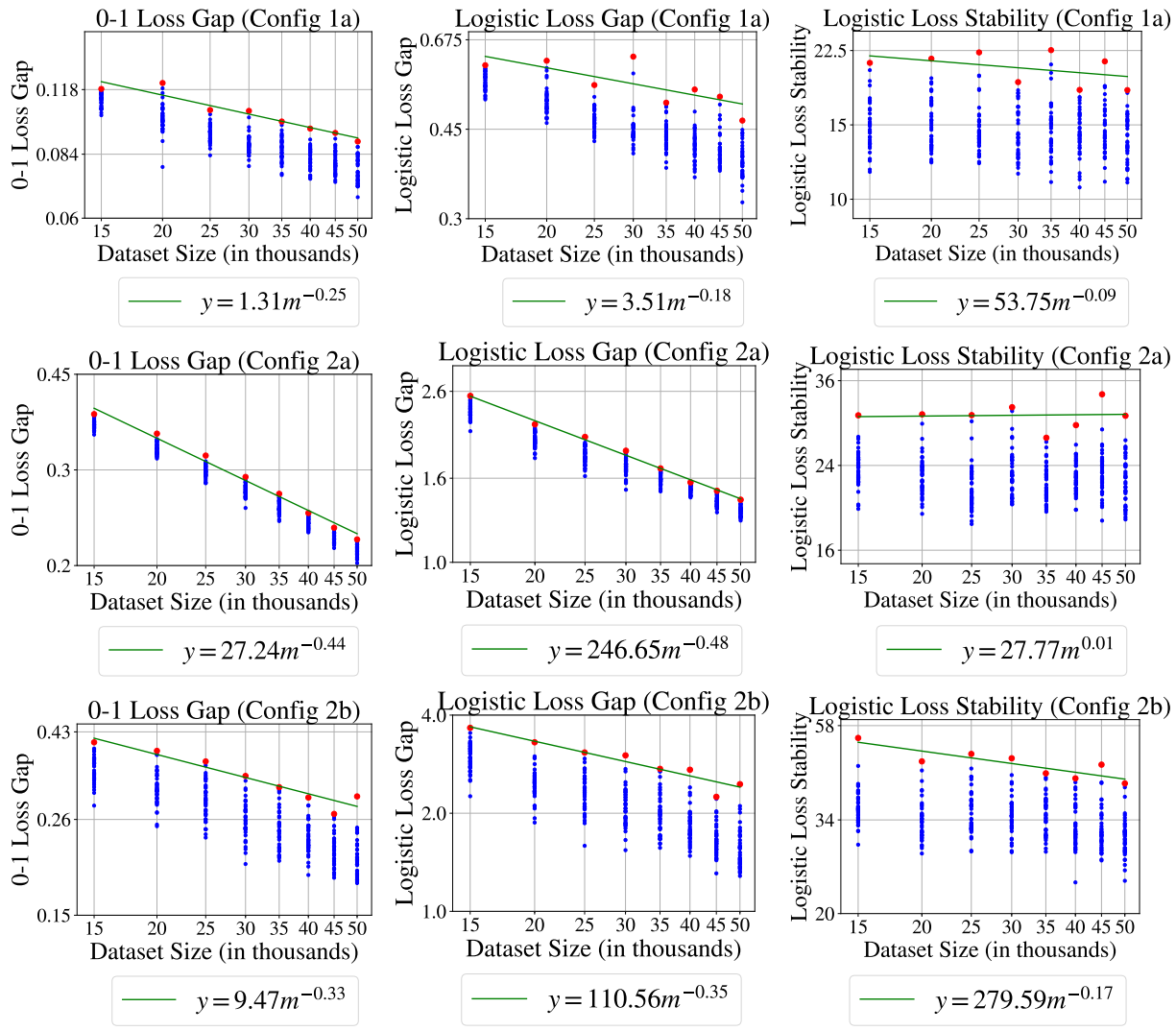


Figure 2.2: Main neural network results. Each row is a different neural network configuration. Per row, from left to right: 0-1 loss generalization gap as a function of dataset size, cross-entropy loss generalization gap as a function of dataset size, cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's dependence on m. Note: We use "logistic loss" and "cross-entropy" loss interchangeably here; all models in this figure were trained and evaluated with the cross-entropy loss.

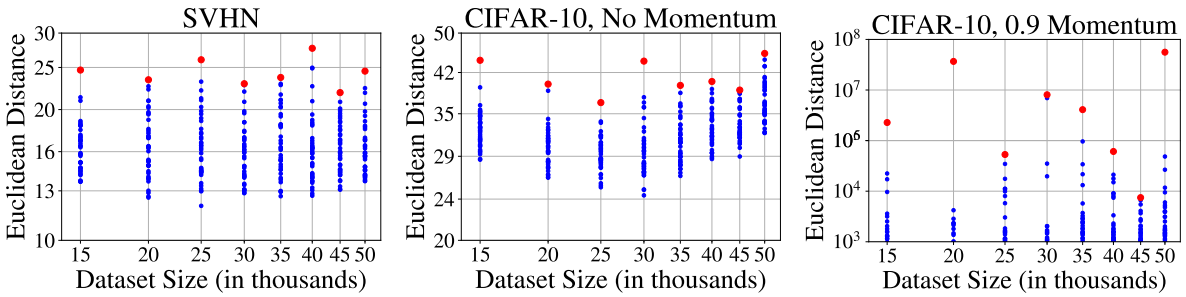


Figure 2.3: $\|\mathcal{A}(S) - \mathcal{A}(S')\|_2$ at $t = 100,000$, for each of the 3 neural network configurations. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, and the maximum value per dataset size m is plotted as a red dot.

to the 0-1 loss generalization gap and note that, at least for these particular configurations, attempting to explain the rate of decrease with m of the cross-entropy loss generalization gap does not leave us too far from the 0-1 generalization gap either.

Appendix A.2 includes the same experiments repeated with more seeds (for initialization and SGD data order) and includes plots at other stopping points (other than 100,000 iterations).

Conclusions. Overall, in our deep learning experiments, uniform stability with respect to the cross-entropy loss does not appear with sufficient strength to explain observed generalization with respect to the cross-entropy loss.

2.4 Behavior of Parameters

In this section, we analyze the behavior of the underlying parameters to try to disentangle the effect of the cross-entropy loss and the supremum over the domain (estimated via the max over the test set) from the learned models themselves in parameter space.

2.4.1 Euclidean distance

As mentioned in Section 2.1, we are further interested in studying the Euclidean distance between the final learned parameters to help understand whether the key proof strategy introduced by Hardt et al. (2016) extends to practical deep learning settings. Since this paper, most proofs of the stability of SGD (even in nonconvex settings) proceed by bounding the Euclidean distance in parameter space between $\mathcal{A}(S)$ and $\mathcal{A}(S')$ and then appealing to the Lipschitzness of the loss. However, if the Euclidean distance between $\mathcal{A}(S)$ and $\mathcal{A}(S')$ does not decrease with dataset size in our trained models, this suggests that this proof strategy might not be sufficient for obtaining generalization bounds in practical deep learning settings that hold with high probability (over the random draw of the dataset and the random initialization and SGD data order of the algorithm).

Results. Figure 2.3 presents $\|\mathcal{A}(S) - \mathcal{A}(S')\|_2$ for our three neural network configurations. We see that, from $m = 15k$ to $m = 50k$, the distances do not decrease with dataset size at a sufficient rate to explain generalization and actually even increase in some dataset size ranges.

Conclusions. These results suggest that a decrease in Euclidean distance of the parameters with dataset size is likely not a viable path through which to prove stability in *practical* deep learning settings.

One might ask whether the nondecreasing Euclidean distance we observe here is caused by the norms in parameter space *themselves* growing with dataset size. We first emphasize that this question does not impact our conclusions, as the proofs to which we have referred invoke the raw Euclidean distance between the parameters. However, for completeness, we refer the interested reader to Appendix A.3 for an extensive analysis of norms and normalized Euclidean distances.

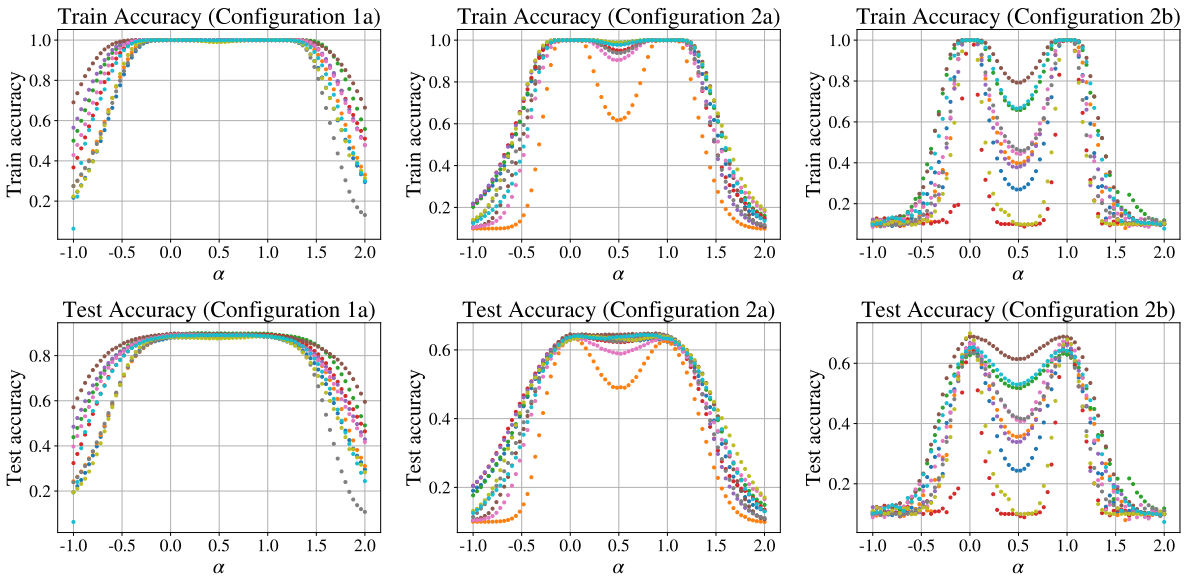
2.4.2 Linear mode connectivity

We now ask the question: Is nonconvexity causing optimization on S and S' to diverge to different basins of attraction, thus thwarting efforts to extend analyses from convex settings to deep learning's nonconvex setting (as is done by Hardt et al. (2016) and follow-up works)? Here, we use *basin of attraction* to mean a convex set of solutions (in parameter space) all with comparable training and/or test loss.

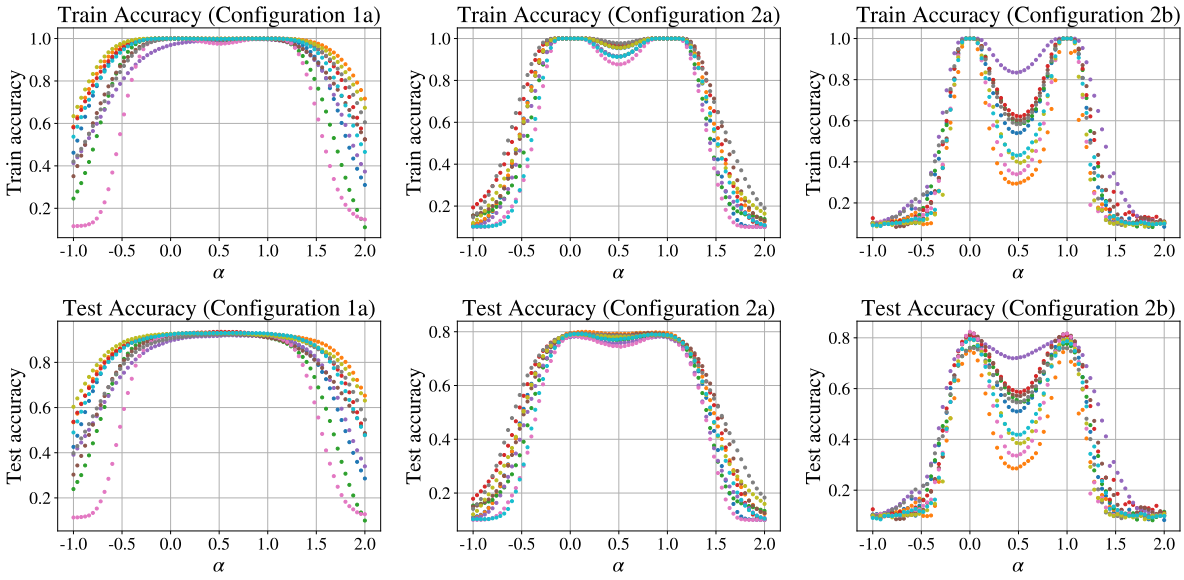
To make this more precise, we invoke the *linear mode connectivity* framework of Frankle et al. (2020) to study this question. Specifically, linear mode connectivity asks whether, at all networks along the linear path between two candidate networks (in parameter space), the training and/or test error does not increase. In our setting, W_S and $W_{S'}$ qualify as linearly connected modes if, for all $\alpha \in [0, 1]$, the (test or train) accuracy of the model with parameters $\alpha W_S + (1 - \alpha)W_{S'}$ is not significantly below that of W_S or $W_{S'}$ (roughly 2%, per Frankle et al. (2020)).

Results. We plot the train accuracy (on S) and test accuracy at $\alpha W_S + (1 - \alpha)W_{S'}$ at 76 equally-spaced values of $\alpha \in [-1, 2]$. The learned parameters are at $\alpha = 0$ and $\alpha = 1$, but we include additional values of α on either end as a frame of reference. We randomly select 10 trials among the 40 trials described in Section 2.2 and, for each trial, we plot the train and test accuracy for each value of α. Figure 2.4a has results for $m = 15,000$ and Figure 2.4b has results for $m = 50,000$. The ResNet-20 on SVHN has nondecreasing accuracy when linearly interpolating between W_S and $W_{S'}$, the ResNet-20 on CIFAR-10 without momentum has slightly decreasing accuracy when linearly interpolating between W_S and $W_{S'}$, and the ResNet-20 on CIFAR-10 with momentum has significantly decreasing accuracy when interpolating.

Conclusions. A priori, it is not obvious what one should expect when linearly interpolating, and it is thus perhaps surprising that our three configurations largely span the space of possibilities. Thus, in order to further study the weaknesses of uniform stability in practical deep learning settings, we suggest that moving to a regime such as a ResNet-20 on CIFAR-10 *with* momentum, in which the solutions are *not* connected by a path of nondecreasing accuracy, might not be immediately necessary from a scientific standpoint. Perhaps the limitations of uniform stability



(a) Training dataset size $m = 15,000$.



(b) Training dataset size $m = 50,000$.

Figure 2.4: Linear mode connectivity at $t = 100,000$. Each column has a different neural network configuration. In each plot, the x-axis is α, ranging from -1.0 to 2.0, and the y-axis is the train or test accuracy evaluated at the parameters $\alpha W_S + (1-\alpha)W_{S'}$ for some (S, S') pair. Specifically, each color represents a different (S, S') pair from among the 40 trials described in Section 2.2; each plot includes 10 such pairs randomly selected from among the 40. Figure 2.4a has results for training dataset size $m = 15,000$ and Figure 2.4b has results for training dataset size $m = 50,000$. Overall, the ResNet-20 on SVHN has nondecreasing accuracy when linearly interpolating between W_S and $W_{S'}$, the ResNet-20 on CIFAR-10 without momentum has slightly decreasing accuracy when linearly interpolating between W_S and $W_{S'}$, and the ResNet-20 on CIFAR-10 with momentum has significantly decreasing accuracy when interpolating.

can be explored and better understood with a configuration such as our ResNet-20 on SVHN (without momentum). Notably, the SVHN interpolation results suggest that nonconvexity might not be necessary at all to investigate the particular weaknesses of algorithmic stability experienced by deep learning; rather, *convex* settings with large enough basins of attraction (defined for our purposes as convex sets of parameters yielding approximately equal training and/or test loss) to host a reasonable degree of functional diversity might actually be subject to these same weaknesses. Thus, our findings might open the door to the study of more tractable, convex settings in which one can study the same limitations of algorithmic stability that appear in deep learning.

2.5 Conclusion

In this work, we have initiated the challenging endeavor of empirically studying the uniform stability of deep learning. Although we freely admit that no reasonable empirical results could *definitively* rule out uniform stability (due to its formulation as several maxima over the domain), we believe that our results present compelling evidence that (a) uniform stability (with respect to the cross-entropy loss) might not be present in practical deep learning with sufficient strength to explain generalization, and (b) that typical theoretical approaches based on parameter distance decreasing with dataset size are likely not the driving force behind any form of algorithmic stability that nevertheless might exist in deep learning. Ultimately, if some form of algorithmic stability (perhaps weaker than uniform stability) is at play in deep learning, we suspect that it will stem from a function-space view that appropriately handles divergence to different basins of attraction after swapping one data point (as seen in Section 2.4.2, Configuration 2b). However, in the meantime, we present compelling evidence that many of the weaknesses of uniform stability can already be seen empirically in simpler, perhaps even convex, settings. Formalizing and further investigating these more tractable settings presents an interesting direction for future work.

Chapter 3

Local Signal Adaptivity

This chapter is based on Karp et al., 2021:
Stefani Karp, Ezra Winston, Yuanzhi Li, and Aarti Singh. Local Signal Adaptivity: Provable Feature Learning in Neural Networks Beyond Kernels. In Advances in Neural Information Processing Systems, 2021.

3.1 Introduction

Recently, deep learning (using multi-layer, *non-linear* neural networks) has demonstrated superior performance over traditional linear learners in many machine learning tasks. These achievements have bred much *theoretical* investigation into whether neural networks are, in fact, superior - and why. On the one hand, the Neural Tangent Kernel (NTK) and derivative works show that, under certain (limiting) conditions, a gradient-descent-trained neural network reduces to a kernel method with a specific architecture- and initialization-determined kernel (Jacot et al., 2018a; Du et al., 2019a). However, this does *not* seem to be the full story, as it fails to capture the feature learning aspect of neural network training. This distinction between a *fixed* feature representation and a *data-adaptive* feature representation has been studied from a variety of perspectives, including the *lazy* vs. *active* regime framework (Chizat et al., 2019; Woodworth et al., 2020; Moroshko et al., 2020; Geiger et al., 2020; Wang et al., 2020). Building on these insights, there has been increasing interest in now showing a *provable gap* between the performance of neural networks and kernel methods in various settings (Ghorbani et al., 2019; Ghorbani et al., 2020; Allen-Zhu and Li, 2019; Allen-Zhu and Li, 2020a; Li et al., 2020b; Malach et al., 2021; Kamath et al., 2020; Refinetti et al., 2021; Daniely and Malach, 2020; Chen et al., 2020; Domingo-Enrich et al., 2021).

In this work, we extend this theoretical investigation into the superiority of neural networks over linear learners and, inspired by practical settings, propose a new line of reasoning that we refer to as "Local Signal Adaptivity". In particular, we explore the power of convolutional neural networks (CNNs) in image classification, compared to linear functions over prescribed feature mappings.

Our setting: We study a simple data distribution that captures one key property of natural image classification tasks: a small set of localized "label-determining" features embedded within a "noisy" background. We formally prove that even when such background occupies a rather large fraction of an image, a CNN can be quite effective at locating the label-determining signal and thus ignoring background information that is mostly irrelevant to the true label, to achieve high accuracy.

Our result: In the formal setting of our simple data distribution (presented in Section 3.3), we ask whether a particular two-layer CNN trained via stochastic gradient descent can provably acquire this "signal-finding" ability and how this compares to its associated convolutional neural tangent kernel (CNTK). We answer with a separation result between our CNN and its CNTK: *We formally prove that a small CNN, trained using standard SGD from random initialization, can efficiently learn to find the "signal" and threshold out the noise*, whereas the corresponding CNTK requires *a comparatively much larger model* (i.e., with more features) in order to accomplish this.

Empirical justification: While we pick a simple data distribution in our work to illustrate the main idea and obtain a formal proof, we point out that our setting is very natural in real images: in many image classification tasks, the label-determining feature only occupies a small fraction of the image, and most other parts are background noise (Figure 3.1). Furthermore, in neural networks trained on natural images, it is generally accepted that activation patterns become increasingly sparse throughout training, effectively zeroing out the activations of low-magnitude noise and locating the true signal (suggestive of the denoising/LSA phenomenon studied in this work). For completeness, we have included such an experiment in Figure 3.2, illustrating how the average percentage of active neurons per instance decreases throughout training (details are provided in Appendix B.3). Finally, to empirically study our theoretical results, we create a new dataset by embedding CIFAR-10 images within either random Gaussian or ImageNet backgrounds. Our experiments show that, as the intensity of the background noise grows and thus the "denoising task" becomes harder, the performance of the neural network stays relatively stable, while the performance of the corresponding NTK does, in fact, degrade significantly (Section 3.6, Figure 3.4).

Based on our theorem and experiments, we therefore believe that this per-instance "signal finding within noisy backgrounds" ability of convolutional neural networks, which we dub "Local Signal Adaptivity" (LSA), is one key component of the superiority of SGD-trained convolutional neural networks over fixed feature mappings.

3.2 Related Work

Learning neural networks using SGD. There is a long line of work proving that various concept classes can be learned efficiently by SGD-trained neural networks. Many of the theoretical works taking a "beyond-NTK" perspective and emphasizing the non-linearity of neural networks are only proved under Gaussian inputs (Kawaguchi, 2016; Soudry and Carmon, 2016; Xie et al., 2016; Ge et al., 2017; Soltanolkotabi et al., 2017; Tian, 2017; Brutzkus and Globerson, 2017; Zhong et al., 2017; Li and Yuan, 2017; Boob and Lan, 2017; Li et al., 2018; Vempala and Wilmes, 2018; Ge

Figure 3.1: Example images from the ImageNet2012 dataset, illustrating background noise in natural image classification tasks. The labels are: bicycle (top left), canoe (top right), school bus (bottom left), balloon (bottom right). In each example, the label-determining objects are only one part of the whole image.

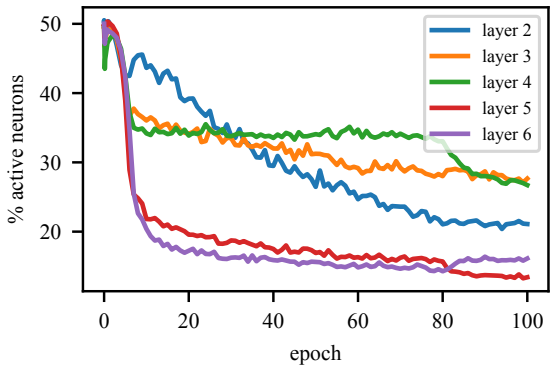


Figure 3.2: Sparsity of intermediate WRN layers during training on CIFAR-10. The x-axis is the epoch number, and the y-axis is the average percentage of active neurons per instance. Full training details are provided in Appendix B.3. The plot illustrates how activation patterns become increasingly sparse throughout training, effectively zeroing out low-magnitude noise, as in the LSA phenomenon.

et al., 2018; Bakshi et al., 2018; Oymak and Soltanolkotabi, 2019; Yehudai and Shamir, 2019; Li and Yuan, 2017; Li and Liang, 2017; Li et al., 2016; Li and Dou, 2020; Li et al., 2020a). In our work, we consider a more realistic setting in which there is a highly-structured signal hidden within random background noise.

We now give a more detailed comparison of our work with prior theoretical results on the separation between neural networks and kernels.

Representation power of neural networks. Many existing works focus on separating the representation power of neural networks from that of other models (Ghorbani et al., 2020; Refinetti et al., 2021; Gribonval et al., 2020; Suzuki, 2019; Suzuki and Nitanda, 2019). However, the fact that a function can be represented efficiently by a neural network does *not* mean that it can be efficiently learned. Only a subset of such works prove efficient learnability, including Suzuki and Akiyama (2020) and Daniely and Malach (2020). In our work, we focus on the set of functions that can be *efficiently learned* by neural networks, in particular learned via stochastic gradient descent over the standard training objective starting from random initialization.

Classification vs. regression. Many existing works only focus on separating the power of neural networks from that of other learners in a regression setting (Ghorbani et al., 2020; Suzuki and Akiyama, 2020; Allen-Zhu and Li, 2020a; Allen-Zhu and Li, 2019; Li et al., 2020a; Wei et al., 2018; Yehudai and Shamir, 2019). In this case, both the neural network and the other learning methods are required to fit the exact label. Although Daniely and Malach (2020) presents one of the few results in a classification setting, their final separation result is still in terms of hinge loss instead of 0-1 loss. In our result, we focus on the *binary classification setting*, where the neural network and (significantly) the linear learner are only required to fit the sign of the label, instead

of the exact labeling function. Our final result is therefore more natural in image classification settings than prior works.

Learning a hidden subspace. Many of the existing works showing how neural networks are better than other learners, in particular kernel methods, focus on the case where the concept class is of the form $f(x) = \phi(Wx)$, where W is a rank-deficient matrix (Ghorbani et al., 2020; Wei et al., 2018). Thus, the learning process can be divided into two phases: (1) Identifying the hidden subspace of W; (2) Learning the function ϕ over this hidden subspace. We point out that to the best of our knowledge, all the cited works only shows the superior power of neural networks in (1). This means that, if W were known, then a neural network would have the same sample/time complexity as kernels when performing (2). As we will show, in our work, the underlying concept class does not have a fixed subspace, rather the signal can drift between different patches, and the neural network needs to identify the signal patch while ignoring the noise patches. Our explanation of how neural networks outperform kernels is therefore fundamentally different from the cited works.

Kernel lower bounds. Most of the existing theoretical works demonstrating the power of neural networks focus on showing how neural networks are better than kernel methods or linear functions over prescribed feature mappings. There are two related lines of work: (1) Showing that neural networks are better than any kernel method or any linear learner (Allen-Zhu and Li, 2020a; Daniely and Malach, 2020); (2) Showing that neural networks are better than a *particular* set of linear learners, most notably, the NTKs or finite-width NTKs of the corresponding neural network (Yehudai and Shamir, 2019; Refinetti et al., 2021; Wei et al., 2018). Our work belongs to the second line. Although a result along the first line gives a much stronger separation, we point out that in the *classification* setting, *such a separation is known to be extremely challenging in computational complexity theory* (O'Donnell and Servedio, 2003; Sherstov and Wu, 2019).

3.3 Problem Setup

We now formally state our data distribution and the learning problem in our paper.

Relevant problem parameters. We consider the input $X = (X_1, \cdots, X_{k+1})$ to have $k+1$ patches; each patch is associated with a vector $X_i \in \mathbb{R}^d$. We can think of X as either the input image or the output of some intermediate layer of the convolutional neural network. It is convenient to think of X as a matrix with $k+1$ rows and d columns, i.e., $X \in \mathbb{R}^{(k+1)\times d}$. We treat k as "large" and d as polynomial in k, enabling us to simplify certain results by expressing them entirely in terms of k. We consider an unknown signal vector $\mathbf{w}_\star \in \mathbb{R}^d$ with ℓ_2-norm $\|\mathbf{w}_\star\|_2 = 1$, which determines how each (image) X is generated (described below). We also set $\sigma = \log k/\sqrt{k}$, where σ determines the intensity of the noise in our problem (described below).

Data distribution. We sample each (image, label) pair $(X, y) \sim \mathcal{D}$. The marginal distribution $\mathcal{D}_y$ is a uniform distribution over $\{-1, 1\}$, i.e., both classes occur with equal probability. We first sample y from $\mathcal{D}_y$ and then sample $X \sim \mathcal{D}_{X|y}$ as follows: (a) draw $i^\star$ from $\{1, \ldots, k+1\}$ arbitrarily and put $y\mathbf{w}_\star$ in row $i^\star$ (we call this the "signal" row); (b) in each of the k remaining

rows (we call these the "noise" rows), put an independently-drawn vector $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I_{d\times d})$. We use $X_i \in \mathbb{R}^d$ to denote row i of X, for $i \in \{1, \ldots, k+1\}$. Formally, we therefore have:

$$X_i = \begin{cases} y\mathbf{w}_\star & \text{if } i = i^\star \\ \epsilon_i & \text{otherwise} \end{cases}$$

Our data distribution shares many similarities with the distribution studied in Yu et al. (2019). However, perhaps most crucially among the *differences*, the noise magnitude in our setting is *significantly* higher; in Yu et al. (2019), the noise magnitude is low enough to maintain *linear* separability, which is insufficient to show any separation with linear learners.

Remark. We consider the simplest setting where there is only one feature $\mathbf{w}_\star$, though our result trivially extends to the case when there are multiple orthogonal features (as many as d). For example, the signal patch can contain features of the form $\sum_i \alpha_i \mathbf{w}_i^\star$, where the label is determined by $\sum_i \alpha_i$, as in Allen-Zhu and Li, 2020b.

How to learn this concept class. In our concept class, one of the rows X_i is associated with the true signal $y\mathbf{w}_\star$, and all the others are random Gaussian noise. Since this row can appear arbitrarily in one of the $k+1$ rows, the most naive way to learn this function is to use a convolutional linear classifier:

$$l(X) = \sum_{i=1}^{k+1} \langle \mathbf{w}_\star, X_i \rangle.$$

However, we argue that this linear classifier is actually very bad and cannot be used to classify the labels correctly. Note that in our problem setup, the noise rows are sampled according to $\mathcal{N}(0, \sigma^2 I_{d\times d})$, which means that for each noise row i, we have $\langle \mathbf{w}_\star, X_i \rangle \sim \mathcal{N}(0, \sigma^2)$. Hence, the *total accumulated noise* over k noise rows would be $\mathcal{N}(0, k\sigma^2)$. By our choice of $\sigma = \log k/\sqrt{k}$, this means that the total noise is much bigger than the signal $|y| = 1$. Hence, the linear classifier fails to classify the labels correctly with at least probability 0.49.

The above argument suggests that, to learn the concept class, the model cannot naively sum up $\langle \mathbf{w}_\star, X_i \rangle$. Rather, the model has to identify the signal row and ignore the noise rows. In other words, the model needs to distinguish between the case when $\langle \mathbf{w}_\star, X_i \rangle$ is large (y) or small ($\mathcal{N}(0, \sigma^2)$). As we will show, this can be *efficiently* learned by a neural network with ReLU activations, trained using standard stochastic gradient descent from random initialization.

Convolutional neural network (CNN) architecture. Our goal is to learn the optimal parameters $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ of the following simple CNN:

$$f_{\mathbf{w},b}(x) = \sum_{i=1}^{k+1} \phi_b(\langle \mathbf{w}, X_i \rangle) = \sum_{i=1}^{k+1} \Big[\mathrm{ReLU}(\langle \mathbf{w}, X_i \rangle + b) - \mathrm{ReLU}(-\langle \mathbf{w}, X_i \rangle + b) \Big].$$

This CNN can be understood from *either* of two equivalent perspectives:

(1) A single convolutional filter $\mathbf{w} \in \mathbb{R}^d$ with stride d is applied to the image. Then the *soft-thresholding* activation function $\phi_b(x) = \mathrm{ReLU}(x+b) - \mathrm{ReLU}(-x+b)$ is applied pointwise

over the resulting $(k+1)$-dimensional vector. Finally, the second (non-trainable) layer of the CNN simply sums up the $k+1$ entries (i.e., the second layer is a fully-connected layer mapping from $\mathbb{R}^{k+1}$ to $\mathbb{R}$, with non-trainable weights all equal to 1).

(2) *Two* convolutional filters, each with stride d, are applied to the image. One filter is $\mathbf{w} \in \mathbb{R}^d$ and the other is $-\mathbf{w} \in \mathbb{R}^d$. After each filter is applied, the activation function $\varphi_b(x) = \mathrm{ReLU}(x+b)$ is applied pointwise to all $2k+2$ pre-activation values (i.e., a $k+1$ "vector" with *2 channels*). A non-trainable fully-connected layer is then applied, mapping $\mathbb{R}^{2k+2}$ to $\mathbb{R}$; in this layer, each of the $k+1$ weights applied to the the first channel (i.e., from filter $\mathbf{w}$) are all 1, and each of the weights applied to the second channel (i.e., from filter $-\mathbf{w}$) are all -1.

The key in both perspectives is how the two ReLU activation functions work together to implement a soft-thresholding function, which enables denoising. Throughout, we refer to our CNN as having one filter, since it only has $d+1$ trainable parameters, regardless of which perspective is taken.

Training algorithm. We initialize b deterministically at 0. We initialize $\mathbf{w}$ randomly by drawing from $\mathcal{N}(0, \sigma_0^2 I_{d\times d})$, where σ_0 is $1/\mathrm{poly}(k)$.

We train the above CNN using mini-batch stochastic gradient descent (SGD) with the logistic loss, where the logistic loss ℓ is defined as $\ell(f_{\mathbf{w},b}(X), y) := \log\left(1 + e^{-y f_{\mathbf{w},b}(X)}\right)$.

Specifically, at each iteration of SGD, we use $\mathrm{poly}(k)$ fresh samples from $\mathcal{D}$. This will allow us to argue that, at each iteration, the empirical gradient is very close to the true population loss gradient.

To simply the proof, we use a slightly smaller learning rate for b (denoted η_b) than for $\mathbf{w}$ (denoted $\eta_{\mathbf{w}}$). Specifically, we adopt a $1/\mathrm{poly}(k)$ learning rate for $\mathbf{w}$, and we set $\eta_b/\eta_{\mathbf{w}} = 1/k$. With a bit more technical care, our proofs can be extended to cover the setting where $\eta_b = \eta_{\mathbf{w}}$ but we use this simpler version to illustrate the main idea of the learning process, as we will state in the next section.

To avoid any ambiguity, we define this procedure explicitly as Algorithm 3.1.

Algorithm 3.1 Mini-batch SGD

Initialization and learning rate $b^{(0)} \leftarrow 0$; $\mathbf{w}^{(0)} \leftarrow \mathcal{N}(0, \sigma_0^2 I_{d\times d})$; $\eta_b \leftarrow \eta/k$; $\eta_{\mathbf{w}} \leftarrow \eta$
for $t \leftarrow 1 \dots T$ **do**
  Sample a mini-batch of examples of size $n = \mathrm{poly}(k)$: $\mathcal{Z} \leftarrow \mathcal{D}^n$
  Compute the stochastic gradient: $g_b = \frac{1}{n}\sum_{z\in\mathcal{Z}} \nabla_b \ell(z)$, $g_{\mathbf{w}} = \frac{1}{n}\sum_{z\in\mathcal{Z}} \nabla_{\mathbf{w}} \ell(z)$
  Update using stochastic gradient descent: $b^{(t)} \leftarrow b^{(t-1)} - \eta_b g_b$, $\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t-1)} - \eta_{\mathbf{w}} g_{\mathbf{w}}$
end for
Return $b^{(T)}, \mathbf{w}^{(T)}$

Additional notation. We will occasionally use the shorthand $f_t(\cdot)$ to denote $f_{\mathbf{w}^{(t)},b^{(t)}}(\cdot)$, the network at iteration t. We use the standard big-O notation and its variants: $\mathcal{O}(\cdot), o(\cdot), \Theta(\cdot), \Omega(\cdot), \omega(\cdot)$, where k is the problem parameter that becomes large. Occasionally, we use the symbol $\widetilde{\mathcal{O}}(\cdot)$ (and analogously with the other four variants) to hide $\log k$ factors.

Our results will hold with high probability over the random initialization of $\mathbf{w}$ and the randomness of the mini-batches, where "high probability" here means a failure probability

super-polynomially small in k. Unless specified otherwise, *w.h.p.* can be taken to mean: with probability at least $1 - e^{-\Omega(\log^2 k)}$.

3.4 Neural Network Results

We now present and briefly describe our main result on the provably *efficient* learning of the CNN described in Section 3.3.

Theorem 3.1: Main result

There exists an absolute constant k_0 such that, for every $k \geq k_0$, using $\text{poly}(k)$ samples from $\mathcal{D}$, learning rate $\eta = 1/\text{poly}(k)$, and $T = \text{poly}(k)$ iterations, w.h.p. over the randomness of the initialization and the samples, we have $\Pr_{(X,y)\sim\mathcal{D}}[\text{sign}(f_T(X)) \neq y] \leq 0.01$, for the final network f_T returned by Algorithm 3.1.

Therefore, in contrast to its corresponding Neural Tangent Kernel (Section 3.5), our CNN trained via SGD *provably* achieves high accuracy *efficiently*. This stands in contrast to some of the prior works discussed in Section 3.2, many of which do not prove efficient learnability.

We defer our detailed proofs to Appendix B.1 and summarize the key ideas here.

Empirical vs. population gradients. Our general proof technique involves analyzing the gradient of the *population loss* at each iteration (we call this the *true gradient*): $\nabla_{\mathbf{w}}\mathbb{E}_{(X,y)\sim\mathcal{D}}[\ell(f_{\mathbf{w},b}(X), y)]$ and $\nabla_b\mathbb{E}_{(X,y)\sim\mathcal{D}}[\ell(f_{\mathbf{w},b}(X), y)]$. Then, with $\text{poly}(k)$ samples per mini-batch, we argue that *w.h.p.* the empirical gradient concentrates around the true gradient, and over $T = \text{poly}(k)$ steps, the accumulated error is fairly small. This argument is made rigorous in Appendix B.1. Therefore, in the remainder of this section, to illustrate the key idea of the proof, we limit our discussion to the *true gradient* as just defined (and use the term *gradient* or *derivative* without further qualification).

Learning the signal direction. As training progresses, $\mathbf{w} \cdot \mathbf{w}_\star$ grows from its small-magnitude initialization to a relatively large, but still $\mathcal{O}(1)$, *positive* value. Specifically, at each iteration, the gradient with respect to $\mathbf{w}$ has a component parallel to $\mathbf{w}_\star$, with two *opposing* forces determining the sign/magnitude of this component: in expectation over $(X, y) \sim \mathcal{D}$,

1. the k "noise" rows of X push $\mathbf{w} \cdot \mathbf{w}_\star$ to *shrink* and
2. the "signal" row pushes $\mathbf{w} \cdot \mathbf{w}_\star$ to *grow*.

The "signal" row's contribution has a magnitude of $\Theta(1)$, and the k "noise" rows have a total contribution of magnitude at most $\mathcal{O}\left(|\mathbf{w} \cdot \mathbf{w}_\star| \cdot k \log k \cdot \sigma^2\right) = \mathcal{O}\left(|\mathbf{w} \cdot \mathbf{w}_\star| \cdot \log^3 k\right)$. Therefore, as long as $|\mathbf{w} \cdot \mathbf{w}_\star| = o(1/\log^3 k)$, the "signal" row's contribution overpowers the "noise" rows' contribution, causing $\mathbf{w} \cdot \mathbf{w}_\star$ to grow. This means that, within $\Theta((\eta_{\mathbf{w}} \log^4 k)^{-1}) = \text{poly}(k)$ iterations, $\mathbf{w} \cdot \mathbf{w}_\star$ rises from its small-magnitude initialization to $\Theta(1/\log^4 k)$. After $\mathbf{w} \cdot \mathbf{w}_\star$ rises to $\Theta(1/\log^4 k)$, we no longer track its dynamics explicitly, as it must stay somewhere between $\Omega(1/\log^4 k)$ and $\mathcal{O}(1)$ throughout the rest of training, and this turns out to be sufficient for the remainder of the argument. The main lemma we prove in Appendix B.1 regarding the growth of $\mathbf{w} \cdot \mathbf{w}_\star$ is a slightly more formal version of the following:

Lemma 3.1: Sketched

For any $t \le T$, if $|\mathbf{w}^{(t)} \cdot \mathbf{w}_\star| = o\left(\frac{1}{\log^3 k}\right)$, then $\nabla_{\mathbf{w}} \mathbb{E}[\ell(f_t(X), y)] \cdot \mathbf{w}_\star = -\Theta(1)$.

Bounded growth along non-signal directions. As training progresses, we also track $\|\mathbf{w} - (\mathbf{w} \cdot \mathbf{w}_\star)\mathbf{w}_\star\|_2$, the magnitude of $\mathbf{w}$'s projection onto the orthogonal complement of $\mathrm{span}\{\mathbf{w}_\star\}$. Although the projection's direction can change, we show that its norm remains very small, allowing $\mathbf{w} \cdot \mathbf{w}_\star$ to dominate.

Specifically, letting $\mathbf{w}_\perp$ denote $\mathbf{w} - (\mathbf{w} \cdot \mathbf{w}_\star)\mathbf{w}_\star$ at the start of iteration t, we show that $\nabla_{\mathbf{w}} \mathbb{E}[\ell(f_t(X), y)] \in \mathrm{span}\{\mathbf{w}_\star, \mathbf{w}_\perp\}$. Further, $\nabla_{\mathbf{w}} \mathbb{E}[\ell(f_t(X), y)] \cdot \mathbf{w}_\perp / \|\mathbf{w}_\perp\|_2$ mirrors the "noise" contribution above: it has magnitude at most $\mathcal{O}\left(\|\mathbf{w}_\perp\|_2 \cdot k \log k \cdot \sigma^2\right) = \mathcal{O}\left(\|\mathbf{w}_\perp\|_2 \cdot \log^3 k\right)$ and pushes $\|\mathbf{w}_\perp\|_2$ to *shrink*. Thus, if we were actually using the *true gradient*, $\mathbf{w}_\perp$ would maintain its direction, and its norm would shrink. The effect of the *stochastic* gradient is that $\mathbf{w}_\perp$ *can* change direction slightly, and its norm can grow a bounded amount per iteration; however, as long as the mini-batch size is large enough ($\mathrm{poly}(k)$ suffices), its norm stays small enough throughout training. The main lemma we prove in Appendix B.1 regarding $\|\mathbf{w}_\perp\|_2$ is therefore a slightly more formal version of the following:

Lemma 3.2: Sketched

For any $t \le T$, let $g_\perp^{(t)}$ denote $\nabla_{\mathbf{w}} \mathbb{E}[\ell(f_t(X), y)] \cdot \mathbf{w}_\perp^{(t)} / \|\mathbf{w}_\perp^{(t)}\|_2$. Then $g_\perp^{(t)} \ge 0$ and $g_\perp^{(t)} = \mathcal{O}\left(\|\mathbf{w}_\perp^{(t)}\|_2 \cdot \log^3 k\right)$.

Learning to threshold out the noise. The derivative with respect to b similarly has two *opposing* forces determining its sign/magnitude: in expectation over $(X, y) \sim \mathcal{D}$,

1. the k "noise" rows of X push b to decrease and
2. the "signal" row pushes b to increase.

The "signal" row's contribution has a magnitude of $\Theta(1)$. Thus, we can only guarantee that b is decreasing if the "noise" rows' total contribution is $\omega(1)$. Roughly, the "noise" rows' contribution is determined by (i) the scalar projection of each noise vector ϵ_i onto $\mathbf{w}$ and (ii) (when $b < 0$) how much of these scalar projections survive "thresholding". As $\|\mathbf{w}\|_2$ grows throughout training (primarily due to the growth of $\mathbf{w} \cdot \mathbf{w}_\star$, discussed above), (i) becomes larger. However, as b decreases (i.e, $|b|$ grows, for $b < 0$), (ii) becomes smaller (i.e., much of the noise does not survive "thresholding"). Therefore, the crux of the proof is to show that, as long as the probability of misclassification is still > 0.01, despite b already "thresholding out" a fair amount of the noise, the "noise" rows' total contribution will remain $\omega(1)$ and thus cause b to further decrease. The main lemma we prove in Appendix B.1 regarding b is therefore a slightly more formal version of the following:

Lemma 3.3: Sketched

For any $t \le T$, if $\mathbf{w}^{(t)} \cdot \mathbf{w}_\star = \omega(1/k^{1/8})$, then $\nabla_b \mathbb{E}[\ell(f_t(X), y)] = \Omega(1)$.

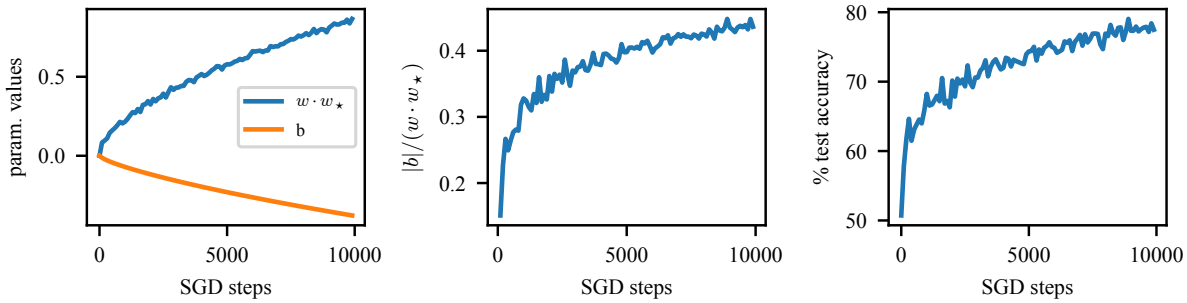


Figure 3.3: Training our model on synthetic data, using $k = 1000$, $d = 10$, $\sigma = 1$, $\eta_w = 0.1$, $\eta_b = 0.1/1000$, and minibatch size 500 (with new i.i.d. samples generated for each batch). In each plot, the x-axis is the SGD step number. The left plot shows the values of b and $\mathbf{w} \cdot \mathbf{w}_\star$, the center plot shows the ratio $|b|/(\mathbf{w} \cdot \mathbf{w}_\star)$, and the right plot shows the test accuracy. Overall, as training progresses, $\mathbf{w} \cdot \mathbf{w}_\star$ increases, b decreases, the ratio $|b|/|\mathbf{w} \cdot \mathbf{w}_\star|$ increases, and the test accuracy increases correspondingly, tightly aligning with the theory.

We note that the requirement $\mathbf{w}^{(t)} \cdot \mathbf{w}_\star = \omega(1/k^{1/8})$ is satisfied by the $\Omega(1/\log^4 k)$ lower bound presented above, thus connecting the two phases of training.

Comment on actual dynamics. The elegance of this approach is that it largely allows us to ignore b's behavior prior to $\mathbf{w} \cdot \mathbf{w}_\star$ reaching the $\Omega(1/\log^4 k)$ regime. In reality, there is a short phase at the beginning of training during which b *grows* (i.e., becomes increasingly *positive*); this occurs because, even though none of the noise is being "thresholded out" at this point, $\|\mathbf{w}\|_2$ is not yet large enough for the "noise" rows' contribution to dominate the "signal" row's $\Theta(1)$ contribution (which itself does *not* scale with $\|\mathbf{w}\|_2$). Then, at some point *prior* to $\mathbf{w} \cdot \mathbf{w}_\star$ reaching $\Omega(1/\log^4 k)$, b begins to decrease rather than increase - and, as we prove, eventually continues to decrease throughout the rest of training. However, for our performance guarantee, it does not matter exactly *when* this transition from increasing to decreasing actually occurs. The $\eta_b/\eta_{\mathbf{w}} = 1/k$ ratio ensures that $|b|/|\mathbf{w} \cdot \mathbf{w}_\star|$ never exceeds $\Theta(1/k)$ while $b > 0$, which means that $b = \mathcal{O}(1/k)$ throughout its *positive* phase.

Provable efficiency. We argue that if $\Pr_{(X,y)\sim\mathcal{D}}[\text{sign}(f_T(X)) \neq y] \geq 0.01$, then we must have $|\mathbf{w} \cdot \mathbf{w}_\star| = \mathcal{O}(1)$. This, along with the rest of the argument, is made fully rigorous in Appendix B.1. We sum up the total number of iterations as follows. First, there are $\mathcal{O}((\eta_{\mathbf{w}} \log^4 k)^{-1})$ iterations before b begins to decrease. Then, because $|\mathbf{w} \cdot \mathbf{w}_\star| = \mathcal{O}(1)$, $\nabla_b \mathbb{E}[\ell(f_t(X), y)] = \Omega(1)$, and the classification accuracy is controlled by the ratio $|b|/|\mathbf{w} \cdot \mathbf{w}_\star|$, we conclude that there are $\text{poly}(k)$ iterations before $|b|$ becomes large enough to yield $\Pr_{(X,y)\sim\mathcal{D}}[\text{sign}(f_T(X)) \neq y] \leq 0.01$. Thus, $T = \text{poly}(k)$, and with $\text{poly}(k)$ samples per mini-batch, we have total sample complexity $\text{poly}(k)$.

Illustration of training dynamics with synthetic data. Figure 3.3 illustrates these training dynamics on synthetic data (with $k = 1000$, $d = 10$, $\sigma = 1$, $\eta_w = 0.1$, $\eta_b = 0.1/1000$). As can be seen in the figure, as training progresses, $\mathbf{w} \cdot \mathbf{w}_\star$ increases and b decreases (i.e., $b < 0$, with increasing magnitude). Furthermore, as the ratio $|b|/|\mathbf{w} \cdot \mathbf{w}_\star|$ grows (and thus more and more noise is "thresholded out"), the test accuracy increases as well, tightly aligning with the theory.

3.5 Kernel Results

In this section, we compare the function learned by our simple convolution neural network to its corresponding finite-width CNTK (Allen-Zhu et al., 2019a). We define the finite-width CNTK as:

$$k_{\mathbf{w}}(X) = \sum_{i\in[k]}\sum_{j\in[m]} \langle \mathbf{w}_j, X_i\rangle 1_{|\langle \mathbf{w}_j^{(0)}, X_i\rangle|+b_j\geq 0},$$

which is a linear function over the gradient of $f_{\mathbf{w},b}(X)$ at initialization. We show that this NTK is unable to classify the target function correctly, unless the total number of channels m is large. For simplicity we consider the case when $d = \frac{1}{\sigma^2}$.

Theorem 3.2: Kernel lower bound

As long as $m = O(1)$, w.p. at least 0.999 over the random initialization $\{\mathbf{w}_j^{(0)}\}_{j\in[m]}$ where each $\mathbf{w}_j^{(0)}$ i.i.d. $\sim \mathcal{N}(0, \sigma_0^2 I)$, we have that for every set of weights $\mathbf{w}$ and for every set of biases $\{b_j\}_{j\in[m]}$,

$$\Pr_{X,y\sim\mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y] \geq 0.1.$$

Compared to our convolutional neural network, which only requires $m = 1$ neurons and can learn the target function correctly, the corresponding finite-width CNTK needs $\omega(1)$ neurons in order to even represent the target function. This is due to the fact that the features $\mathbf{w}_j^{(0)}$ *are prescribed at random initialization*, instead of trained. Thus, even with arbitrary tuned bias b_j, the neural tangent kernel still fails to adapt to the local signal and perform denoising.

3.6 Experiments

We now provide concrete empirical evidence that the LSA phenomenon does explain the performance gap between CNNs and kernels in real-world datasets. To gain more insight, we construct several CIFAR-10 (Krizhevsky, 2009b) variants with various forms of structured noise. These datasets capture a key property of real-world image classification tasks, that the signal is localized to patch amidst a large amount of background noise. Although real-world images also exhibit this property (as in Figure 3.1), we wanted the ability to easily vary the intensity of the background noise, while leaving the signal intact. On each dataset, we compare the performance of a 10-layer WideResNet (with widening factor of 10) to that of its finite-width NTK (Zagoruyko and Komodakis, 2016). We vary the intensity of the noise and observe the resulting degradation in WideResNet (WRN) and NTK performance. Details of the architectures and training procedures are given in Appendix B.3.

Each dataset is constructed by scaling a CIFAR image to 16x16 pixels and placing it onto a 32x32 noise background. In some instances described below, we used ImageNet backgrounds (Deng et al., 2009). Experiments on larger-sized images and different image placements are described in Appendix B.4. Here we highlight experiments on four such datasets:

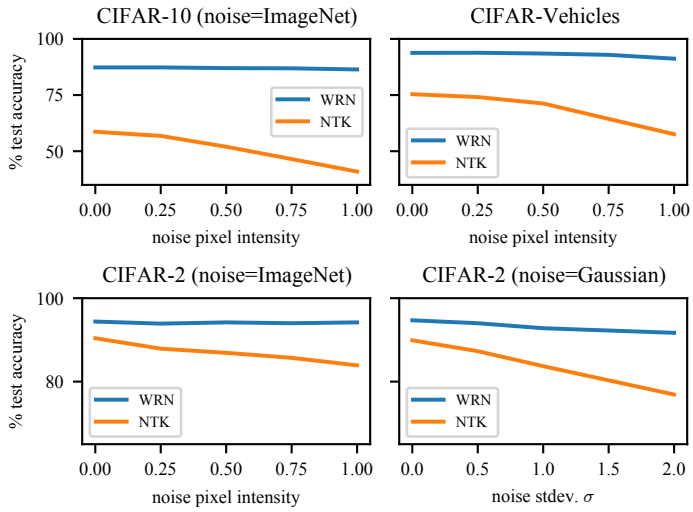


Figure 3.4: Each plot shows how the WRN and its corresponding NTK respond to increasing the intensity of the background noise, in various image settings: (top left) CIFAR-10, ImageNet noise, (top right) CIFAR-Vehicles, (bottom left) CIFAR-2, ImageNet noise, (bottom right) CIFAR-2, Gaussian noise, with more details in Section 3.6. The x-axis is the scale of the background noise or the standard deviation of the background noise, and the y-axis is the test accuracy. The WRN (blue) retains most of its performance as the noise intensity increases, whereas the corresponding NTK (orange) has degraded performance.

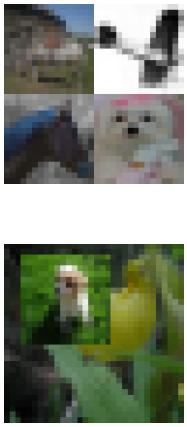


Figure 3.5: Examples from CIFAR-Vehicles (top) and CIFAR-10 with ImageNet noise (bottom), both for background noise pixel intensity scaled to 0.75.

- **CIFAR-10, ImageNet noise**. CIFAR-10 images are placed at a random location onto random background image chosen from the ImageNet Plants synset. The Plants synset was chosen for its visual similarity to backgrounds in real images. In our experiments, we scale the background pixel intensity in a range between zero (black background) and one (original ImageNet background). An example is shown in Figure 3.5 (bottom).
- **CIFAR-2, ImageNet noise**. Same as above, except the CIFAR-10 classes are grouped into animals and vehicles and the classification task is now binary. In contrast to CIFAR-10, on CIFAR-2 the NTK performs nearly as well as the CNN in the zero-noise setting.
- **CIFAR-2, Gaussian noise**. Also using the CIFAR-2 task, but with standard Gaussian noise in the background, for a range of standard deviations σ.
- **CIFAR-Vehicles**. The task is to classify between the four vehicle classes from CIFAR-10. The vehicle appears in a random corner of the image, and the other three corners are filled with random images from the four CIFAR-10 animal classes. See Figure 3.5 (top). Unlike the relatively uniform ImageNet plants backgrounds, this dataset is designed to capture a common type of background noise which consists of "distractor" signals which are not predictive of the true image class. For example, irrelevant people, bicycles, and birds could all occur in the background of a real-world vehicle-classification task.

Observations. As seen in Figure 3.4, as the scale of the noise increases, NTK performance decreases significantly while WRN performance is relatively unaffected. Table 3.1 gives the percent of test accuracy retained by each model as noise intensity increases from 0 to 1 (or from $\sigma = 0$ to $\sigma = 2$ in the Gaussian-noise case); that is, the percentage $100 \cdot \frac{\text{test acc. at max noise level}}{\text{test acc. at zero noise}}$. The WRN always retains over 96% of its zero-noise performance, while the NTK in one case degrades below 70%.

While these datasets capture key attributes of real images, their synthetic construction does limit how realistic the images can be. We constructed the dataset in this way in order to allow tunable noise levels, thus providing the x-axis of the plots in Figure 3.4. In fact, the scaling of the noise by pixel intensity is another somewhat-unrealistic aspect, since in real images we might expect the background and signal to have similar intensities. One could imagine alternative ways of creating tunable noise levels, such as increasing the size of the background or diversity of distractor images; these are all viable avenues for future experimentation.

Table 3.1: The percentage of the zero-noise test accuracy still retained at the maximum noise level, for each of the image types described in Section 3.6. Specifically, each number in this table is $100 \cdot \frac{\text{test acc. at max noise level}}{\text{test acc. at zero noise}}$. Overall, the WRN always retains over 96% of its zero-noise performance, while the NTK in one case degrades below 70%.

	CIFAR-10, ImNet noise	**CIFAR-Vehicles**	**CIFAR-2**, ImNet noise	**CIFAR-2**, Gauss. noise
WRN	98.97	97.25	99.79	96.83
NTK	69.75	76.37	92.81	85.54

3.7 Conclusion

We have considered a simple, noisy data distribution that captures some of the key structure seen in natural images. We have (1) provably shown that a particular two-layer CNN trained via SGD can *efficiently* (in time and sample complexity) achieve high accuracy and (2) that its corresponding linear model (the finite-width NTK) requires a much larger network size (i.e., more features). Overall, our results shed light on a new mechanism through which neural networks are provably better than their corresponding kernels: in particular, when there is a signal hidden within background noise, a neural network is able to simultaneously adapt to the local signal and perform "denoising". We provide empirical justification showing that our theoretical framework does coincide with the superior power of neural networks over linear learners in practice.

One avenue for future work involves extending the noise distribution: increasing σ beyond $\log k/\sqrt{k}$, which can provide an even stronger separation with linear learners, and extending to other noise models beyond Gaussian. Another possible extension is to drop the restriction that the same filter is used in both parts of the activation function and study whether this soft-thresholding behavior is recovered automatically upon training the final-layer weights. We could also consider more general models in which the signal "patch" and the CNN filter are not perfectly aligned. Another possibility is to develop a limitation result for all kernel methods instead of the CNTK corresponding to our CNN; however, as discussed in Section 3.2, this would

likely be a regression result instead of a classification result, which is weaker in other ways. Finally, it would be interesting to extend this theoretical analysis to deeper networks and thus more practical CNNs (perhaps "hierarchical denoising").

Chapter 4

Transformer Theory

This chapter is based on Karp et al., 2023:
Stefani Karp, Pranjal Awasthi, and Satyen Kale. Provable Gradient-Descent-Based Learning of Decision Lists by Transformers. In DeepMath, 2023.

4.1 Introduction

Despite the incredible success of Transformers (Vaswani et al., 2017), there is much that remains to be understood about the attention mechanism and its learning via gradient-based algorithms. The theoretical studies of gradient-based optimization in Transformers are rather limited and paint a far-from-complete picture.

One key line of understanding-focused work studies in-context learning, the Transformer's impressive ability to learn from examples of a task provided in its prompt (Brown et al., 2020). Elhage et al., 2021 and Olsson et al., 2022 introduce and formalize induction heads and present them as a possible candidate for the main mechanism underlying in-context learning. Other works attempting to understand in-context learning include Garg et al., 2022; Oswald et al., 2023; Ahn et al., 2023; Fu et al., 2024. On the theoretical side, most rigorous proofs here are limited to linear attention in order to make the analysis more tractable.

A more general line of work, beyond in-context learning, seeks to provide a mechanistic understanding of how gradient descent works in simple Transformer models (Jelassi et al., 2022; Bietti et al., 2023; Tian et al., 2023; Li et al., 2023b). These works have made valuable progress in understanding some of the core mechanisms underlying gradient-based learning in Transformer models. However, most of these works do not provide full gradient descent convergence guarantees under softmax-based attention starting from uniform attention.

Another challenge these analyses face is how to design significant but theoretically-tractable sequence distributions. What types of distributions capture meaningful characteristics of the sequence-to-sequence tasks solved by self-attention?

In this chapter, we present some of the first convergence proofs for gradient descent on the parameters of a one-layer Transformer with softmax-based attention, on a simple data

distribution where the Transformer's attention plays a *key role* in solving the task. Specifically, we introduce and formalize a simple decision-list-like data distribution and a slight generalization thereof, serving as a theoretically simple, near-minimal example of a data distribution for which learning the attention head is crucial in solving the task. We study our simple Transformer's ability to learn this data distribution, and we prove efficient gradient-descent-based learning under the population loss.

We first focus on just training the parameters of the Transformer *inside* the softmax-based-attention and prove that gradient descent on the population loss can efficiently learn these parameters. We then present a generalization of our data distribution and prove a corresponding *local* convergence result, in which the Transformer's value matrix is trained too, beginning from a "good enough" initialization.

4.2 Setup

4.2.1 Simplified Transformer model

Let L denote the vocabulary size, n denote the sequence length, and d denote the embedding dimension. For a sequence of tokens of length n, let $X \in \mathbb{R}^{d\times n}$ be the matrix formed by concatenating the embeddings of the n tokens. A *Transformer* model is composed of attention layers which have $d \times d$ parameter matrices W_Q, W_K, and W_V and computes new embeddings for the sequence via the function:

$$X \mapsto \sigma((W_V X)\sigma((W_K X)^\top (W_Q X))),$$

where for any matrix M, $\sigma(M)$ is the softmax function applied to the columns of M.

In the following, we consider a simplified Transformer model where the product $W_K^\top W_Q$ is replaced by a single $d \times d$ parameter matrix W, and for simplicity we use V to denote the matrix W_V:

$$f_{W,V}(X) := \sigma(VX\sigma(X^\top WX)), \tag{4.1}$$

Further, we assume that the embedding dimension d equals the vocabulary size L, and the input embeddings are simple one-hot embeddings, i.e. token i is embedded as the standard basis vector e_i, which is 0 in all coordinates except i, where it is 1. Thus, a sequence $(i_1, i_2, \ldots, i_n) \in [L]^n$ is represented by $X = [e_{i_1} \mid e_{i_2} \mid \ldots | e_{i_n}]$.

We assume also that the output vocabulary size is L. For a given input sequence represented by X, let $Y = (y_1, y_2, \ldots, y_n) \in [L]^n$ be the label sequence. The loss of the transformer model $f_{W,V}$ is given by the loss on a single sequence (X, Y) is now defined as

$$\mathcal{L}((W, V); (X, Y)) = \frac{1}{n}\sum_{j=1}^{n} -\log(f_{W,V}(X)_{y_j,j}). \tag{4.2}$$

We analyze the dynamics of gradient flow on the above loss function on the sequence-to-sequence mapping described in the next section.

4.2.2 Sequence-to-sequence mapping

We consider an arbitrary *permutation function* $\pi : [L] \to [L]$, and we let P denote the $L \times L$ permutation *matrix* associated with the permutation *function* π. The sequence-to-sequence mapping of interest is computed as follows. Let M denote a positive integer assumed to be much less than L. For every token $a \in [L]$, we associate a list of tokens $L_a = (a_1, a_2, \ldots, a_m)$ of length $m \leq M$, such that $a_m = a$. Given a sequence $s \in [L]^n$, the output label y for any token $a \in s$ is equal to $\pi(a_j)$, where j is the smallest index such that $a_j \in s$. We call a_j the *label-determining token* for a in s, since it *determines* the label: $\pi(a_j)$. Note that since $a_m = a \in s$, j always exists. We call this a decision-list-like data distribution because this mapping can be realized by a *decision list* associated with each token a.

4.2.3 Data distribution

We make several key assumptions on the data distribution.

Assumption 4.1 (No duplicate tokens)**.** *Each token in the vocabulary appears at most once per input sequence.*

We also make the following assumption about the relationship between the input data distribution (over length-n sequences) $\mathcal{D}$ and the parameters at any time step:

Assumption 4.2 (Minimum misclassification event probability)**.** *Suppose some token a is misclassified in at least one sequence. Let $k^\star$ denote the first label-determining token (according to the ordering in L_a) for which there is such a misclassification event. Let S denote the set of sequences s in which the following all occur: a is in s, a is misclassified in s, and $k^\star$ is a's label-determining token in s. For any such set S for which $\Pr_{s\sim\mathcal{D}}[s \in S] > 0$, we assume that $\Pr_{s\sim\mathcal{D}}[s \in S] \geq \delta$.*

We will consider two settings. In one setting, the matrix V is fixed at cP. In this setting, we can achieve Assumption 4.2 through the following parameter-independent assumption (i.e., only an assumption on the data distribution).

Assumption 4.3 (Minimum probability per sequence-dependent triple)**.** *For any token triple (a, b, c) such that $b \in L_a$ and c either appears after b in L_a or is not in L_a, if $\Pr_{s\sim\mathcal{D}}[a, b, c \in s$ and b is a's label-determining token in $s] > 0$, then*

$$\Pr_{s\sim\mathcal{D}}[a, b, c \in s \text{ and } b \text{ is } a\text{'s label-determining token in } s] \geq \delta.$$

This assumption is sufficient to imply Assumption 4.2 because, in order for a to be misclassified when b is its label-determining token, there must be some other token c whose weight in a's column of W is higher than b's weight in this column. As long as c appears in the sequence when b is a's label-determining token, then a will be misclassified.

In the second setting, however, when V is not fixed at cP, it is no longer guaranteed that a misordering of weights in W will lead to misclassification, since the application of matrix V can subtly affect the final order of the logits. Therefore, in this setting, for a given V, whether or not a particular misordered pair of weights in W actually leads to a misclassification depends on the exact post-softmax-attention values, which are affected by exactly which other tokens

appear simultaneously in the sequence. As a result, in this setting, it is more difficult to achieve Assumption 4.2 through a parameter-independent assumption. One simple way is to ensure that every set of tokens appears with a minimum probability.

Assumption 4.4 (Minimum probability per token set)**.** *For any size-n set of tokens $\mathcal{S} \in \{1, \dots, L\}^n$, if $\Pr_{s\sim\mathcal{D}}[i \in s \ \ \forall s \in \mathcal{S}] > 0$, then*

$$\Pr_{s\sim\mathcal{D}}[i \in s \ \ \forall s \in \mathcal{S}] \geq \delta.$$

However, this assumption is stronger, and it can lead to a much smaller value of δ.

For generality, we will work with Assumption 4.2 throughout the rest of this chapter and merely present Assumption 4.3 and Assumption 4.4 as more interpretable, parameter-free alternatives at the cost of a possibly smaller value of δ.

4.3 Main Results

In this section, we present our two main results. In the first, the matrix V is fixed at cP for sufficiently large c. In the second, the matrix V is initialized near cP for sufficiently large c and is trained alongside W.

We assume that the data are generated according to the distributional assumptions and the sequence-to-sequence mapping given in the previous sections. We provide a convergence analysis for gradient descent on the population loss. The population loss is defined as

$$\mathcal{L}(W, V) = \mathbb{E}_{(X,Y)}[\mathcal{L}((W, V); (X, Y)).$$

For simplicity, we will sometimes drop the parameters in $\mathcal{L}(W, V)$ and just write $\mathcal{L}$ when our algebra does not depend on the particular parameters.

Gradient descent on the population loss is defined by the following update equations:

$$W_{t+1} = W_t - \eta\nabla_W\mathcal{L}(W_t, V_t)$$
$$V_{t+1} = V_t - \eta\nabla_V\mathcal{L}(W_t, V_t)$$

with some given initialization W_0, V_0.

We let $\mathcal{L}_t$ denote the loss at time t, i.e., $\mathcal{L}_t := \mathcal{L}(W_t, V_t)$.

Throughout, we will consider two versions: one where we train W only and one where we train both W and V. In the former, we will often drop the V, as in $\mathcal{L}(W)$ vs. $\mathcal{L}(W, V)$.

We will use $\nabla\mathcal{L}$ to denote $\nabla_W\mathcal{L}$ when training just W and $\nabla\mathcal{L}$ to denote $\nabla_{W,V}\mathcal{L}$ when training both W and V.

We will use $\mathcal{L}(W, V; s)$ and related variations to denote the loss on a particular sequence s.

4.3.1 Informal theorem statements

Theorem 4.1: Training only W (Informal)

Fix $V = cP$ for sufficiently large c, and initialize $W_0 = 0$. Run gradient descent on the population loss with sufficiently small learning rate η. Gradient descent will find a W_t that achieves perfect classification on all sequences within a small number of steps.

Theorem 4.2: Training W and V (Informal)

Initialize $W_0 = 0$ and V_0 close to cP, for sufficiently large c. Run gradient descent on the population loss with sufficiently small learning rate η. Gradient descent will find a W_t, V_t that achieve perfect classification on all sequences within a small number of steps.

4.3.2 Proof sketch

The rough idea of the proof is that misclassification implies a large gradient. Therefore, as long as gradient descent is run long enough, the gradient will become small enough to rule out misclassification. Here, we illustrate this in the simplified setting where π is the identity mapping and the matrix V is fixed to cI for some sufficiently large c.

Due to the 1-hot encodings, we can reason about each column of W (each token in the vocabulary) separately, so let us consider an arbitrary token a. For simplicity, let w denote token a's column in W. We have $L_a = (a_1, \ldots, a_m)$, where $a_m = a$. As long as the order of the weights in w matches the order in L_a, i.e., $w_{a_1} > \cdots > w_{a_m} > w_b$ for all $b \notin L_a$, then token a will be classified perfectly in all sequences.

Consider an arbitrary sequence s. If k is a's true label in sequence s, thenwe can show that

$$\frac{\partial \mathcal{L}(W; s)}{\partial w_k} \leq 0,$$

i.e., the gradient with respect to a's true label in w is negative (or 0). We can further show that if a is not correctly classified as k, then this gradient is also large in magnitude.

Unfortunately, though, this is not enough. When k appears in sequences s where a's true label is not k, then $\frac{\partial \mathcal{L}(W;s)}{\partial w_k}$ can instead be positive. This makes it hard to argue about the population loss over all sequences $\frac{\partial \mathcal{L}(W)}{\partial w_k}$. Is there enough structure to argue about how things balance out?

It turns out that we can instead argue about a certain directional derivative (defined in the following subsections) that reduces to $\frac{\partial \mathcal{L}(W;s)}{\partial w_k}$ whenever k is the true label and is always ≤ 0 otherwise. This circumvents the issue with $\frac{\partial \mathcal{L}(W;s)}{\partial w_k}$ sometimes being positive.

Using this directional derivative, we can then show that misclassification implies a large gradient. Therefore, with sufficiently many steps of gradient descent, we can ensure perfect classification.

4.3.3 Formal theorem statements and proofs

Theorem 4.3: Training only W (Formal version of Theorem 4.1)

Fix $V = cP$, for $c = \frac{4n^2 \log L}{\delta}$. Starting from $W_0 = 0$, run gradient descent on the population loss with learning rate $\eta = \frac{1}{\lambda_{\max}}$, where $\lambda_{\max}$ is the maximum eigenvalue of the Hessian of $\mathcal{L}(W)$. Then, there exists a $t \leq \frac{32M\lambda_{\max}}{\log L}$ such that W_t, V achieve perfect classification on all sequences.

Proof. By contradiction.

Without loss of generality, suppose some token a is misclassified. Among all sequences in which a is misclassified, let $k^\star$ denote the label-determining token that appears *earliest* in L_a. Let S denote the set of sequences where this misclassification pattern occurs - specifically, a is misclassified and $k^\star$ is its label-determining token.

Let p denote the column of $X\sigma(X^\top WX)$ corresponding to a, let w denote the column of matrix W updated due to a, let y denote a's true label, and let $\hat{y}$ denote a's predicted label.

Then, for any $i \in s$:

$$\frac{\partial}{\partial w_k} p_i = \begin{cases} p_i(1-p_i) = -p_i p_k + p_k & k = i \\ -p_i p_k & k \in s, k \neq i \\ 0 & k \in [L] \setminus s. \end{cases}$$

Via the chain rule, we obtain:

$$\frac{\partial \mathcal{L}}{\partial w_k} = p_k \left[\sum_{i \in s} V_{y,i} p_i - V_{y,k} + \sum_{l \in [L]} \sigma_l \left(V_{l,k} - \sum_{i \in s} V_{l,i} p_i \right) \right].$$

We will now examine the directional derivative $\langle \nabla_w \mathcal{L}(W; s), v \rangle$, where $\nabla_w \mathcal{L}(W; s)$ denotes the gradient of the loss on sequence s with respect to column w and we define v as the length-L vector of all 0s other than 1s for the tokens that *precede* $k^\star$ in L_a (as well as $k^\star$ itself).

For any $s \in S$, we have: $\langle \nabla_w \mathcal{L}(W; s), v \rangle \leq -\frac{c}{2} p_{k^\star}(1 - p_{k^\star})$. This comes from $\sigma_y \leq 1/2$ and

the following calculation:

$$\begin{aligned}
\langle \nabla_w \mathcal{L}(W; s), v \rangle &= \frac{1}{n} p_{k^\star} \left[\sum_{i \in s} V_{y,i} p_i - V_{y,k^\star} + \sum_{l \in [L]} \sigma_l \left(V_{l,k^\star} - \sum_{i \in s} V_{l,i} p_i \right) \right] \\
&= \frac{1}{n} p_{k^\star} \left[\sum_{l \in [L]} \sigma_l \left[(z_y - z_l) - (V_{y,k^\star} - V_{l,k^\star}) \right] \right] \\
&= \frac{1}{n} p_{k^\star} \left[\sum_{l \in [L] \setminus \{y\}} \sigma_l \sum_{i \in s \setminus \{k^\star\}} p_i [(\underbrace{V_{y,i} - V_{l,i}}_{\Delta_{l,i}}) - (\underbrace{V_{y,k^\star} - V_{l,k^\star}}_{\Delta_{l,k^\star}})] \right] \\
&\leq \frac{1}{n} p_{k^\star} \left[\sum_{l \in [L] \setminus \{y\}} \sigma_l \sum_{i \in s \setminus \{k^\star\}} p_i (-c) \right] \\
&\leq -\frac{c}{n} p_{k^\star} (1 - \sigma_y)(1 - p_{k^\star}) \\
&\leq -\frac{c}{2n} p_{k^\star} (1 - p_{k^\star}).
\end{aligned}$$

For any $s \notin S$, we have: $\langle \nabla_w \mathcal{L}(W; s), v \rangle \leq 0$. This comes from the following calculation, where we let s' denote the tokens in s which have 1s in v:

$$\begin{aligned}
\langle \nabla_w \mathcal{L}(W; s), v \rangle &= \frac{1}{n} c p_{\pi^{-1}(y)} \left(\sum_{k \in s'} p_k - 1 \right) + \sum_{k \in s'} p_k \sigma_{\pi(k)} c - \sum_{k \in s'} p_k \sum_{t \in s} \sigma_{\pi(t)} c p_t \\
&= \frac{1}{n} c p_{\pi^{-1}(y)} \left(\sum_{k \in s'} p_k - 1 \right) + \sum_{k \in s'} \sum_{t \in s \setminus s'} c p_k p_t \left(\sigma_{\pi(k)} - \sigma_{\pi(t)} \right) \\
&\leq \frac{1}{n} c p_{\pi^{-1}(y)} \left(\sum_{k \in s'} p_k - 1 \right) + \sum_{k \in s'} \sum_{t \in s \setminus s'} c p_k p_t \sigma_{\pi(k)} \\
&= -\frac{1}{n} c p_{\pi^{-1}(y)} \sum_{t \in s \setminus s'} p_t + c \sum_{k \in s'} p_k \sigma_{\pi(k)} \sum_{t \in s \setminus s'} p_t \\
&\leq -\frac{1}{n} c p_{\pi^{-1}(y)} \sum_{t \in s \setminus s'} p_t + c p_{\pi^{-1}(y)} \left(\sum_{k \in s'} \sigma_{\pi(k)} \right) \sum_{t \in s \setminus s'} p_t \\
&\leq 0.
\end{aligned}$$

We will put these parts together to upper bound $\langle \nabla_w \mathcal{L}(W), v \rangle$, the directional derivative of the population loss. However, we first need a lower bound on $p_{k^\star}(1 - p_{k^\star})$. We know that $p_{k^\star} \leq \frac{1}{2}$; otherwise, token a would have been correctly classified. Therefore, we do not have to consider $p_{k^\star}$ being close to 1; we only have to consider $p_{k^\star}$ being close to 0 (i.e., we only have to worry about lower bounding $p_{k^\star}$). Rather than a lower bound that holds for all sequences, we upper bound the total mass where $p_{k^\star}$ can be very small, as follows.

For every $s \in S$ such that $p_{k^\star} \leq \frac{1}{2n}$, there must exist some $l \in s$ such that $p_l \geq 1/n$. This means that, in such cases, the loss on token a is lower bounded:

$$\begin{aligned}
-\log \sigma_y &= -\log \frac{e^{z_y}}{\sum_{t\in[L]} e^{z_t}} \\
&\geq -\log \frac{e^{z_y}}{e^{z_{\pi(l)}}} \\
&= z_{\pi(l)} - z_y \\
&= cp_l - cp_{k^*} \\
&\geq \frac{c}{n} - \frac{c}{2n} \\
&= \frac{c}{2n}.
\end{aligned}$$

However, since we are running gradient descent with sufficient small learning rate, the population loss cannot increase – it is upper bounded by the loss at time $t = 0$. This allows us to upper bound $\delta' := \Pr[s \in S, p_{k^\star} \leq \frac{1}{2n}]$ as follows:

$$\begin{aligned}
\frac{1}{n} \cdot \frac{c}{2n} \cdot \delta' &\leq \mathcal{L}_0 \\
\delta' &\leq \frac{2n^2 \log(L)}{c} \\
\delta' &\leq \frac{2n^2 \log(L)}{\frac{4n^2\log(L)}{\delta}} \\
\delta' &\leq \delta/2.
\end{aligned}$$

We now analyze the directional derivative of the population loss as follows:

$$\langle \nabla_w \mathcal{L}(W), v\rangle = (1) + (2) + (3),$$

where

$$\begin{aligned}
(1) &= \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W; s), v\rangle | s \in S, p_{k^\star} > \frac{1}{2n}\right] \Pr\left[s \in S, p_{k^\star} > \frac{1}{2n}\right] \\
(2) &= \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W; s), v\rangle | s \in S, p_{k^\star} \leq \frac{1}{2n}\right] \Pr\left[s \in S, p_{k^\star} \leq \frac{1}{2n}\right] \\
(3) &= \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W; s), v\rangle | s \notin S\right] \Pr[s \notin S].
\end{aligned}$$

We have already established that (1), (2) and (3) are all ≤ 0, so we can upper bound $\langle \nabla_w \mathcal{L}(W), v\rangle$ by dropping (2) and (3).

We obtain

$$\begin{aligned}\langle \nabla_w \mathcal{L}(W), v\rangle &\leq \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W; s), v\rangle | s \in S, p_{k^\star} > \frac{1}{2n}\right] \Pr\left[s \in S, p_{k^\star} > \frac{1}{2n}\right] \\ &< -\frac{\delta}{2} \cdot \frac{c}{2n} \cdot \frac{1}{2n}\left(1 - \frac{1}{2n}\right) \\ &= -\frac{\delta}{2} \cdot \frac{\frac{4n^2 \log(L)}{\delta}}{2n} \cdot \frac{1}{2n}\left(1 - \frac{1}{2n}\right) \\ &\leq -\frac{\log L}{4}.\end{aligned}$$

We therefore have

$$\frac{\log L}{4} < |\langle \nabla_w \mathcal{L}(W), v\rangle| \leq \|\langle \nabla_w \mathcal{L}(W)\rangle\|\|v\| \leq \|\langle \nabla_W \mathcal{L}(W)\rangle\|\|v\| \leq \sqrt{M}\|\langle \nabla_W \mathcal{L}(W)\rangle\|.$$

By a standard gradient descent analysis, running for T iterations with $\eta = \frac{1}{\lambda_{\max}}$ implies that there exists a $t \in [T]$ such that

$$\|\nabla \mathcal{L}_t\|_2^2 \leq \frac{2\lambda_{\max}\mathcal{L}_0}{T}.$$

When $T = \frac{32M\lambda_{\max}}{\log L}$, this means that there exists a $t \in [T]$ such that

$$\|\nabla \mathcal{L}_t\|_2^2 \leq \frac{\log^2 L}{16M}.$$

However, misclassification implies that

$$\|\nabla \mathcal{L}_t\|_2^2 > \frac{\log^2 L}{16M},$$

which is a contradiction.

Therefore, after $T = \frac{32M\lambda_{\max}}{\log L}$ iterations, there must exist a $t \in [T]$ with perfect classification.

□

Theorem 4.4: Training W and V (Formal version of Theorem 4.2)

Initialize $W_0 = 0$ and V_0 such that $\|V_0 - cP\|_\infty \leq \frac{\epsilon}{2}$ for some $\epsilon > 0$, where $c > \left(4\bar{\mathcal{L}}\sqrt{\frac{M}{\epsilon}} + \frac{6\epsilon}{n}\right)\frac{10n^2}{\delta} + 4\epsilon$, $\epsilon \leq 16M$, and $\bar{\mathcal{L}} := \max\{1, \mathcal{L}_0\}$. Run gradient descent on the population loss with learning rate $\eta = \frac{1}{\lambda_{\max}}$, where $\lambda_{\max}$ is the maximum eigenvalue of the Hessian of $\mathcal{L}(W, V)$. Then, there exists a $t \leq \frac{\lambda_{\max}\epsilon}{2\bar{\mathcal{L}}}$ such that W_t, V_t achieve perfect classification on all sequences.

Proof. By contradiction.

Without loss of generality, suppose some token a is misclassified. Among all sequences in which a is misclassified, let $k^\star$ denote the label-determining token that appears *earliest* in L_a. Let S denote the set of sequences where this misclassification pattern occurs - specifically, a is misclassified and $k^\star$ is its label-determining token.

Let p denote the column of $X\sigma(X^\top WX)$ corresponding to a, let w denote the column of matrix W updated due to a, let y denote a's true label, and let $\hat{y}$ denote a's predicted label.

Then, for any $i \in s$:

$$\frac{\partial}{\partial w_k} p_i = \begin{cases} p_i(1-p_i) = -p_i p_k + p_k & k = i \\ -p_i p_k & k \in s, k \neq i \\ 0 & k \in [L] \setminus s. \end{cases}$$

Via the chain rule, we obtain:

$$\frac{\partial \mathcal{L}}{\partial w_k} = p_k \left[\sum_{i \in s} V_{y,i} p_i - V_{y,k} + \sum_{l \in [L]} \sigma_l \left(V_{l,k} - \sum_{i \in s} V_{l,i} p_i \right) \right].$$

Let v_j denote row j of V. Then

$$\begin{aligned} \nabla_{v_y} \mathcal{L} &= -p + \sigma_y p = (\sigma_y - 1)p \\ (l \neq y) \quad \nabla_{v_l} \mathcal{L} &= \sigma_l p. \end{aligned}$$

We will now examine the directional derivative $\langle \nabla_w \mathcal{L}(W, V; s), v \rangle$, where $\nabla_w \mathcal{L}(W, V; s)$ denotes the gradient of the loss on sequence s with respect to column w and we define v as the length-L vector of all 0s other than 1s for the tokens that *precede* $k^\star$ in L_a (as well as $k^\star$ itself).

For any $s \in S$, we have: $\langle \nabla_w \mathcal{L}(W, V; s), v \rangle \leq -\frac{1}{2n} p_{k^\star}(1 - p_{k^\star})(c - 4\epsilon)$. This comes from Lemma 4.1, $\sigma_y \leq 1/2$, and the following calculation:

$$
\begin{aligned}
\langle \nabla_w \mathcal{L}(W, V; s), v \rangle &= \frac{1}{n} p_{k^\star} \left[\sum_{i \in s} V_{y,i} p_i - V_{y,k^\star} + \sum_{l \in [L]} \sigma_l \left(V_{l,k^\star} - \sum_{i \in s} V_{l,i} p_i \right) \right] \\
&= \frac{1}{n} p_{k^\star} \left[\sum_{l \in [L]} \sigma_l \left[(z_y - z_l) - (V_{y,k^\star} - V_{l,k^\star}) \right] \right] \\
&= \frac{1}{n} p_{k^\star} \left[\sum_{l \in [L] \setminus \{y\}} \sigma_l \sum_{i \in s \setminus \{k^\star\}} p_i [(\underbrace{V_{y,i} - V_{l,i}}_{\Delta_{l,i}}) - (\underbrace{V_{y,k^\star} - V_{l,k^\star}}_{\Delta_{l,k^\star}})] \right] \\
&\leq \frac{1}{n} p_{k^\star} \left[\sum_{l \in [L] \setminus \{y\}} \sigma_l \sum_{i \in s \setminus \{k^\star\}} p_i (-c + 4\epsilon) \right] \\
&\leq \frac{1}{n} p_{k^\star} (1 - \sigma_y)(1 - p_{k^\star})(-c + 4\epsilon) \\
&\leq -\frac{1}{2n} p_{k^\star} (1 - p_{k^\star})(c - 4\epsilon).
\end{aligned}
$$

For any $s \notin S$, we have: $\langle \nabla_w \mathcal{L}(W, V; s), v \rangle \leq 6\epsilon$. This comes from the following calculation,

where we let s' denote the tokens in s which have 1s in v:

$$\begin{aligned}
\langle \nabla_w \mathcal{L}(W, V; s), v \rangle &= \frac{1}{n} \Bigg[c p_{\pi^{-1}(y)} \left(\sum_{k \in s'} p_k - 1 \right) + \sum_{k \in s'} p_k \sigma_{\pi(k)} c - \sum_{k \in s'} p_k \sum_{t \in s} \sigma_{\pi(t)} c p_t \\
&\quad + \sum_{k \in s'} p_k \langle \epsilon_{y,:}, p \rangle - \sum_{k \in s'} p_k \epsilon_{y,k} + \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \epsilon_{t,k} - \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \langle \epsilon_{t,:}, p \rangle \Bigg] \\
&= \frac{1}{n} \Bigg[c p_{\pi^{-1}(y)} \left(\sum_{k \in s'} p_k - 1 \right) + \sum_{k \in s'} \sum_{t \in s \setminus s'} c p_k p_t \left(\sigma_{\pi(k)} - \sigma_{\pi(t)} \right) \\
&\quad + \sum_{k \in s'} p_k \langle \epsilon_{y,:}, p \rangle - \sum_{k \in s'} p_k \epsilon_{y,k} + \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \epsilon_{t,k} - \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \langle \epsilon_{t,:}, p \rangle \Bigg] \\
&\leq \frac{1}{n} \Bigg[c p_{\pi^{-1}(y)} \left(\sum_{k \in s'} p_k - 1 \right) + \sum_{k \in s'} \sum_{t \in s \setminus s'} c p_k p_t \sigma_{\pi(k)} \\
&\quad + \sum_{k \in s'} p_k \langle \epsilon_{y,:}, p \rangle - \sum_{k \in s'} p_k \epsilon_{y,k} + \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \epsilon_{t,k} - \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \langle \epsilon_{t,:}, p \rangle \Bigg] \\
&= -\frac{1}{n} \Bigg[c p_{\pi^{-1}(y)} \sum_{t \in s \setminus s'} p_t + c \sum_{k \in s'} p_k \sigma_{\pi(k)} \sum_{t \in s \setminus s'} p_t \\
&\quad + \sum_{k \in s'} p_k \langle \epsilon_{y,:}, p \rangle - \sum_{k \in s'} p_k \epsilon_{y,k} + \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \epsilon_{t,k} - \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \langle \epsilon_{t,:}, p \rangle \Bigg] \\
&\leq -\frac{1}{n} \Bigg[c p_{\pi^{-1}(y)} \sum_{t \in s \setminus s'} p_t + (c p_{\pi^{-1}(y)} + 2\epsilon) \left(\sum_{k \in s'} \sigma_{\pi(k)} \right) \sum_{t \in s \setminus s'} p_t \\
&\quad + \sum_{k \in s'} p_k \langle \epsilon_{y,:}, p \rangle - \sum_{k \in s'} p_k \epsilon_{y,k} + \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \epsilon_{t,k} - \sum_{k \in s'} p_k \sum_{t \in [L]} \sigma_t \langle \epsilon_{t,:}, p \rangle \Bigg] \\
&\leq 6\epsilon/n.
\end{aligned}$$

We will put these parts together to upper bound $\langle \nabla_w \mathcal{L}(W, V), v \rangle$, the directional derivative of the population loss. However, we first need a lower bound on $p_{k^\star}(1 - p_{k^\star})$ (which requires upper and lower bounding $p_{k^\star}$).

We know that $p_{k^\star} \leq \frac{1}{2} + \frac{\epsilon}{c}$; otherwise, token a would have been correctly classified.

To lower bound $p_{k^\star}$, we further divide S into two subsets: a set where $p_{k^\star} \leq 1/(2n)$, and a set where $p_{k^\star} > 1/(2n)$. We will ultimately ignore the set where $p_{k^\star} \leq 1/(2n)$ (i.e., drop it when upper bounding). Therefore, we focus on lower bounding $\Pr_{s\sim\mathcal{D}}[s \in S, p_{k^\star} > 1/(2n)]$. We can see this as follows:

For every $s \in S$ such that $p_{k^\star} \leq \frac{1}{2n}$, there must exist some $l \in s$ such that $p_l \geq 1/n$. We then have:

$$\begin{aligned}
-\log \sigma_y &= -\log \frac{e^{z_y}}{\sum_{t\in[L]} e^{z_t}} \\
&\geq -\log \frac{e^{z_y}}{e^{z_{\pi(l)}}} \\
&= z_{\pi(l)} - z_y \\
&= cp_l + \langle \epsilon_{\pi(l),:}, p\rangle - cp_{k^*} - \langle \epsilon_{y,:}, p\rangle \\
&\geq \frac{c}{n} - \epsilon - \frac{c}{2n} - \epsilon \\
&= \frac{c}{2n} - 2\epsilon.
\end{aligned}$$

Since we are running gradient descent with a small enough learning rate, the loss is decreasing at every time step. Therefore, $\mathcal{L}_t \leq \mathcal{L}_0$, so at every time step t, we have

$$\begin{aligned}
\frac{1}{n}\left(\frac{c}{2n} - 2\epsilon\right) \Pr\left[s \in S, p_{k^\star} \leq \frac{1}{2n}\right] &\leq \mathcal{L}_0 \\
\Pr\left[s \in S, p_{k^\star} \leq \frac{1}{2n}\right] &\leq \frac{n\mathcal{L}_0}{\frac{c}{2n} - 2\epsilon}
\end{aligned}$$

Plugging in $c = \left(4\bar{\mathcal{L}}\sqrt{\frac{M}{\epsilon}} + \frac{6\epsilon}{n}\right)\frac{10n^2}{\delta} + 4\epsilon$ from the theorem statement, we obtain:

$$\begin{aligned}
\Pr\left[s \in S, p_{k^\star} \leq \frac{1}{2n}\right] &\leq \frac{n\mathcal{L}_0}{\frac{c}{2n} - 2\epsilon} \\
&< \frac{n\mathcal{L}_0}{\frac{\left(4\bar{\mathcal{L}}\sqrt{\frac{M}{\epsilon}} + \frac{6\epsilon}{n}\right)\frac{10n^2}{\delta} + 4\epsilon}{2n} - 2\epsilon} && \text{(plugging in } c\text{)} \\
&= \frac{n\mathcal{L}_0}{\frac{20\bar{\mathcal{L}}n}{\delta}\sqrt{\frac{M}{\epsilon}} + \frac{30\epsilon}{\delta} + \frac{2\epsilon}{n} - 2\epsilon} && \text{(simplifying)} \\
&\leq \frac{n\mathcal{L}_0}{\frac{20\bar{\mathcal{L}}n}{\delta}\sqrt{\frac{M}{\epsilon}}} && (\delta \leq 1) \\
&= \frac{\delta}{20} \cdot \frac{\mathcal{L}_0}{\bar{\mathcal{L}}} \cdot \sqrt{\frac{\epsilon}{M}} && \text{(simplifying)} \\
&\leq \frac{\delta}{5} && (\epsilon \leq 16M).
\end{aligned}$$

We thus conclude that $\Pr\left[s \in S, p_{k^\star} > \frac{1}{2n}\right] \geq \frac{4\delta}{5}$.

We now analyze the directional derivative of the population loss as follows:

$$\langle \nabla_w \mathcal{L}(W, V), v \rangle = (1) + (2) + (3),$$

where

$$\begin{aligned}
(1) &= \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W, V; s), v\rangle | s \in S, p_{k^\star} > \frac{1}{2n}\right] \Pr\left[s \in S, p_{k^\star} > \frac{1}{2n}\right] \\
(2) &= \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W, V; s), v\rangle | s \in S, p_{k^\star} \leq \frac{1}{2n}\right] \Pr\left[s \in S, p_{k^\star} \leq \frac{1}{2n}\right] \\
(3) &= \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W, V; s), v\rangle | s \notin S\right] \Pr[s \notin S].
\end{aligned}$$

We note that, in (1), we have

$$p_{k^*}(1 - p_{k^*}) > \min\left\{\frac{1}{2n}\left(1 - \frac{1}{2n}\right), \left(\frac{1}{2} + \frac{\epsilon}{c}\right)\left(\frac{1}{2} - \frac{\epsilon}{c}\right)\right\},$$

where the first part follows from the lower bound on $p_{k^\star}$ and the second part follows from the fact that we must have $p_{k^\star} \leq \frac{1}{2} + \frac{\epsilon}{c}$; otherwise, token a would have been correctly classified.

Further, we have already established that (2) is ≤ 0 and (3) is $\leq 6\epsilon/n$, so we can upper bound $\langle \nabla_w \mathcal{L}(W, V), v \rangle$ as follows:

$$\begin{aligned}
\langle \nabla_w \mathcal{L}(W, V), v \rangle &\leq \mathbb{E}\left[\langle \nabla_w \mathcal{L}(W, V; s), v\rangle | s \in S, p_{k^\star} > \frac{1}{2n}\right] \Pr\left[s \in S, p_{k^\star} > \frac{1}{2n}\right] + 0 + \frac{6\epsilon}{n} \\
&< -\frac{1}{2n} \min\left\{\frac{1}{2n}\left(1 - \frac{1}{2n}\right), \left(\frac{1}{2} + \frac{\epsilon}{c}\right)\left(\frac{1}{2} - \frac{\epsilon}{c}\right)\right\}(c - 4\epsilon)\frac{4\delta}{5} + \frac{6\epsilon}{n} \\
&< -\frac{1}{2n} \cdot \frac{1}{4n} \cdot (c - 4\epsilon)\frac{4\delta}{5} + \frac{6\epsilon}{n} \\
&\leq -\frac{1}{2n} \cdot \frac{1}{4n} \cdot \left(\left(4\bar{\mathcal{L}}\sqrt{\frac{M}{\epsilon}} + \frac{6\epsilon}{n}\right)\frac{10n^2}{\delta} + 4\epsilon - 4\epsilon\right)\frac{4\delta}{5} + \frac{6\epsilon}{n} \\
&= -4\bar{\mathcal{L}}\sqrt{\frac{M}{\epsilon}}.
\end{aligned}$$

This implies that

$$4\bar{\mathcal{L}}\sqrt{\frac{M}{\epsilon}} < |\langle \nabla_w \mathcal{L}(W, V), v\rangle| \leq \|\nabla_w \mathcal{L}(W, V)\|\|v\| \leq \|\nabla_w \mathcal{L}(W, V)\|\sqrt{M}$$

and therefore

$$\|\nabla_w \mathcal{L}(W, V)\|^2 > \frac{16\bar{\mathcal{L}}^2}{\epsilon}.$$

By a standard gradient descent analysis, running for T iterations with $\eta = \frac{1}{\lambda_{\max}}$ implies that there exists a $t \in [T]$ such that

$$\|\nabla \mathcal{L}(W, V)\|^2 \leq \frac{2\lambda_{\max}\mathcal{L}_0}{T}.$$

This means that, if we run for $T = \frac{\lambda_{\max}\epsilon}{2\bar{\mathcal{L}}}$ iterations, we have

$$\|\nabla \mathcal{L}(W, V)\|^2 \leq \frac{4\mathcal{L}_0\bar{\mathcal{L}}}{\epsilon}.$$

However, misclassification implies

$$\frac{16\bar{\mathcal{L}}^2}{\epsilon} < \|\nabla_w \mathcal{L}(W, V)\|^2 \leq \|\nabla \mathcal{L}(W, V)\|^2 \leq \frac{4\mathcal{L}_0\bar{\mathcal{L}}}{\epsilon},$$

which is a contradiction.

Therefore, after $T = \frac{\lambda_{\max}\epsilon}{2\bar{\mathcal{L}}}$ iterations, we must have perfect classification. □

Lemma 4.1: Invariant on V

For all $t \in [T]$, we have $\|V_t - cP\|_\infty \leq \epsilon$.

Proof. By the gradient calculations, each gradient update can change each entry of V by at most η. Thus, over T time steps, each entry of V can change by at most

$$T\eta = \frac{\lambda_{\max}\epsilon}{2\bar{\mathcal{L}}}\frac{1}{\lambda_{\max}} = \frac{\epsilon}{2\bar{\mathcal{L}}} \leq \frac{\epsilon}{2}.$$

Thus, for any $t \in [T]$, we have:

$$\|V_t - cP\|_\infty \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

□

4.3.4 Experiments

Although our results for training W and V together are currently limited to a local convergence result, where V begins near cP, we show here that the cP structure can - at least in some cases - be picked up fairly easily empirically. Figure 4.1 shows a visualization of matrix V after just one step of gradient descent (with a large learning rate), beginning from random initialization of V (and zero initialization of W). In this setting, we use $L = 200$ and $n = 25$. We use learning rate $\eta = L^2$. We initialize V as follows: $V_0[i, j] \overset{i.i.d.}{\sim} \mathcal{N}(0, 10^{-8})$, and we initialize W at 0: $W_0 = 0$. Our sequences are drawn at random as follows: $s_{\text{in}} \sim U(\{n\text{-permutations of } [L]\})$.

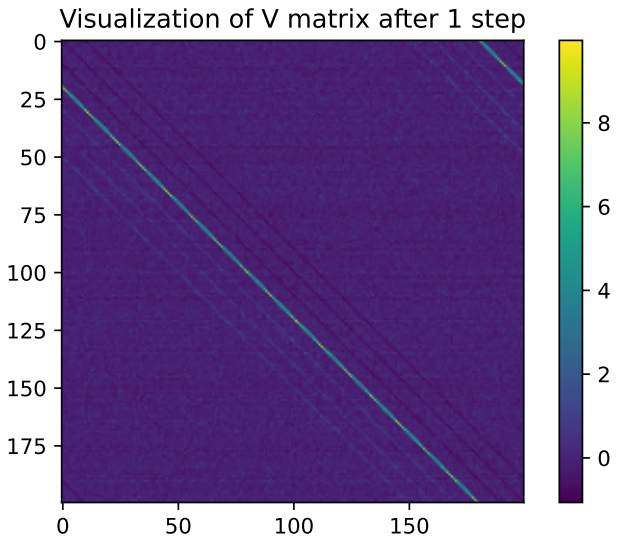


Figure 4.1: A visualization of matrix V after just one step of gradient descent (with a large learning rate), using $L = 200$, $n = 25$, and learning rate $\eta = L^2$. V is initialized at random: $V_0[i, j] \overset{i.i.d.}{\sim} \mathcal{N}(0, 10^{-8})$. W is initialized at 0. The input sequences are drawn at random: $s_{\text{in}} \sim U(\{n\text{-permutations of } [L]\})$. Using the notation $f(i) := i \mod (L + 1)$, the mapping π is $f(i + 20)$, and each decision list is $L_i = (f(i+10), f(i+20), f(i+30), i)$. The visualization shows that, at least in some cases, the structure we want in V can be picked up fairly easily empirically under uniform attention (i.e., $W = 0$).

The ground-truth sequence-to-sequence mapping is designed as follows. Let $f(i) := i \mod (L + 1)$ for notational convenience. The mapping π is then chosen to be $f(i + 20)$, and each decision list is chosen to be $L_i = (f(i + 10), f(i + 20), f(i + 30), i)$.

The takeaway from Figure 4.1 is that the structure we want in V can be picked up under uniform attention (i.e., $W = 0$). This suggests that bridging the gap from a local convergence result to a full convergence result might be tractable in future work.

4.4 Conclusion

In this chapter, we have presented convergence results for a one-layer Transformer on a simple decision-list-inspired data distribution, which crucially requires the Transformer's token-to-token attention in order to solve the task. Although the local convergence result requires the value matrix to be initialized within a "good enough" region in parameter space, empirical results indicate that uniform attention (corresponding to $W = 0$) might be enough to already push V in the direction of P. Therefore, with more work, it might be possible to widen the radius of the local convergence result. Other possible extensions include more complicated data distributions, and we are hopeful that some of the ideas in this document might prove useful in developing such extensions.

Chapter 5

Landscape-Aware Growing

This chapter is based on Karp et al., 2024:
Stefani Karp*, Nikunj Saunshi*, Sobhan Miryoosefi, Sashank J. Reddi, and Sanjiv Kumar. Landscape-Aware Growing: The Power of a Little LAG. arXiv preprint arXiv:2406.02469, 2024.
* denotes joint first authorship.

5.1 Introduction

Large language models with hundreds of billions of parameters have undoubtedly changed the landscape of NLP and AI. However, training such large models is incredibly resource-intensive, motivating the development of efficient training paradigms. One standard way to save resources is to design more efficient architectures and optimization algorithms. Another approach recently gaining popularity is knowledge transfer via the *growing of models*. Here, rather than training a large model from scratch, the idea is to use a much smaller existing pretrained model to initialize the parameters of the larger model (Chen et al., 2015; Wang et al., 2022). This has been shown to accelerate the training of large models compared to training from random initialization. Recently, such ideas were used to train a 100B-parameter-scale model by growing from a 16B-parameter model (Li et al., 2023a). A related approach is stagewise training, where the goal is to train a target model by gradually growing its size. Methods like progressive and gradual stacking (Gong et al., 2019; Reddi et al., 2023) have shown success in efficient stagewise training of BERT models by using layers from the previous stage to initialize the next stage.

These approaches for growing and stacking share a crucial design choice: the particular *growth operator* used to initialize the larger model from the smaller model. In the context of growing in depth, the fundamental design question becomes: *how should we grow an L-layer model into an $(L + k)$-layer model for further training?*

Numerous strategies have been proposed in prior work for growing in depth as well as width (see Section 5.5 for a literature survey). These strategies are largely heuristic or based on the principle of *loss or function preservation* – a growing strategy should ensure that the loss value/functional behavior of the grown model is the same or very similar to that of the smaller

model used to initialize it. At first sight, this seems like a desirable property, because we already know that the small model is a decent pretrained model. Thus, growth operators that maintain the loss value or functional properties will guarantee a good initialization for the larger model, which can be further improved upon by continued training. While this seems intuitive, to the best of our knowledge there is no systematic study of the efficacy of this approach. Thus, we study the following questions:

What are good guiding principles to select the best growing strategy?
Is loss/function preservation a good heuristic?

We explore the above questions in the context of growing in depth[1]. In particular, through extensive empirical analysis, we argue that loss preservation is not necessarily a good strategy. Instead we identify that the training dynamics and *landscape properties* afforded by an initialization play a much bigger role in the success of growing. In this context, we highlight the following contributions:

- For a pool of depth-growing strategies inspired by prior work, we conduct an extensive empirical analysis and find that the loss value at initialization does not correlate well with the final performance of the model (i.e., Pearson correlation of -0.51 and Spearman correlation of -0.42).
- Instead, we propose an alternative view based on the landscape induced by an initialization through the following key observation: while initial loss can be misleading, the loss after a relatively small number of steps (roughly 5000) correlates very strongly with the final performance (i.e., Pearson correlation of 0.98 and Spearman correlation of 0.99).
- We take this a step further and find that good predictions for the best strategy can be made even earlier, after a few hundred steps, and that this corresponds with a measurable phase transition.
- Based on the above empirical observation, we propose a selection strategy called Landscape-Aware Growing (LAG). We validate our hypothesis by testing LAG on 1-stage growing for both BERT and UL2 pretraining. LAG is shown to recover a strategy that is very close to the optimal strategy in hindsight and is also better than many previously considered static strategies. Furthermore, we apply LAG to the setting of gradual stacking by applying LAG to each stacking stage. This improves BERT pretraining loss compared to vanilla gradual stacking, thus further validating the efficacy of LAG.

5.2 Problem Setup: Growing and Stacking

We describe the problem of growing in general and how it can be applied to stacking. This section also sets up the notation for the rest of the paper.

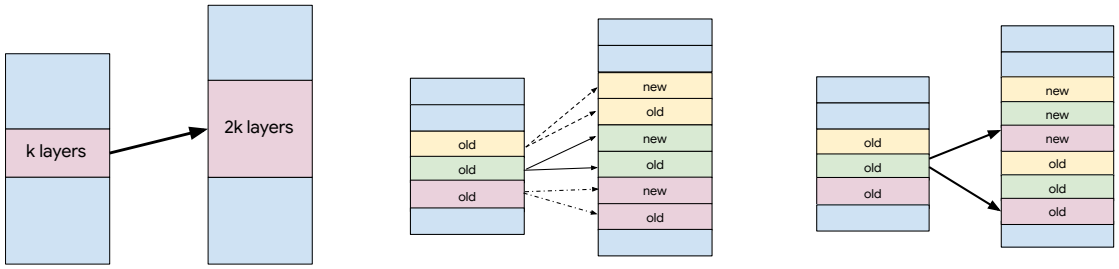


Figure 5.1: Illustration of growing a network. Left: Generic growth of k layers into $2k$ layers. Middle: Example with $L = 6, k = 3, i = 2, b = 1$, parameter duplication (interleaving). Right: Example with $L = 6, k = 3, i = 2, b = 3$, parameter duplication (single-block copying).

5.2.1 Growing

The basic idea of growing is to use one pretrained model (typically a smaller one) to construct an initialization for another model. This growing step can be formalized as a growth operator that maps parameters from a smaller architecture class $\mathcal{F}_1$ to a larger one $\mathcal{F}_2$. We let $G : \mathcal{F}_1 \to \mathcal{F}_2$ denote the growth operator. Given a pretrained checkpoint for a smaller model M_1, the grown checkpoint $M_2 = G(M_1)$ expresses every parameter in the new checkpoint as a function of parameters of the old checkpoint. For Transformer-based architectures, the small and large model classes $\mathcal{F}_1$ and $\mathcal{F}_2$ can potentially vary in either the depth dimension (number of layers) or in width (model dimension, number of attention heads). In this work, for simplicity of analysis, we focus on growing in the depth dimension, i.e., $\mathcal{F}_1$ and $\mathcal{F}_2$ only vary in the depth of the model. However, we note that the ideas being discussed in the paper are general and also apply to other growth dimensions.

We consider growing an L-layer network into an $(L + k)$-layer network in the following manner: choose one set of k consecutive layers and grow it into $2k$ consecutive layers (in various ways). Motivated by prior work on depth growing (Gong et al., 2019; Reddi et al., 2023; Wang et al., 2022), we consider the following design space, parameterized by *index*, *block size*, and *initialization scheme*:

- *Index (i)*: When choosing one set of k consecutive layers in the original L-layer network, where should this set of k layers begin? Throughout, we will let i denote the start index for this set of k layers (1-indexed), so that the feasible set of values for i is $\{1, 2, \ldots, L-k+1\}$.
- *Block size (b)*: How should we divide k into smaller blocks when growing? For simplicity, we divide k into equal-sized blocks and consider all valid block sizes b such that k is divisible by b. At one extreme, when $b = 1$, new and old layers are interleaved. At the other extreme, when $b = k$, the k old layers end up consecutive and the k new layers end up consecutive in the $(L + k)$-layer network.
- *Initialization scheme*: How should the new layers be initialized? We consider both options of random initialization and parameter duplication of the old layers.

To make this more concrete with an example, let us consider a 6-layer network that we wish

[1]Note that one could also grow models in width, but here we restrict our attention to depth-wise growing.

to grow to 9 layers. In this case, $L = 6$ and $k = 3$. The feasible set of start indices is $\{1, 2, 3, 4\}$, and the feasible set of block sizes is $\{1, 3\}$. To help illustrate how the growth operators work, some examples in the design space are as follows, with the new layers indicated in bold.

1. $i = 2, b = 1$, duplication: $[1, 2, 3, 4, 5, 6] \rightarrow [1, 2, \mathbf{2}, 3, \mathbf{3}, 4, \mathbf{4}, 5, 6]$
2. $i = 2, b = 3$, duplication: $[1, 2, 3, 4, 5, 6] \rightarrow [1, 2, 3, 4, \mathbf{2}, \mathbf{3}, \mathbf{4}, 5, 6]$
3. $i = 2, b = 1$, random: $[1, 2, 3, 4, 5, 6] \rightarrow [1, 2, \mathbf{random}, 3, \mathbf{random}, 4, \mathbf{random}, 5, 6]$
4. $i = 2, b = 3$, random: $[1, 2, 3, 4, 5, 6] \rightarrow [1, 2, 3, 4, \mathbf{random}, \mathbf{random}, \mathbf{random}, 5, 6]$

Figure 5.1 (middle, right) illustrates rows 1, 2 above. After the new layers are added, all $L + k$ layers of the resulting model are jointly trained.

5.2.2 Stacking as iterated growing

Above, we discuss a single-step growing operation to transition from L to $L + k$ layers. Throughout, as in Gong et al., 2019 and Reddi et al., 2023, we use "stacking" to refer to the iterated application of this "grow, then train" strategy, where training starts with a shallow model, and at the end of each stage the model depth is grown by a certain amount until the desired depth is achieved. Thus, any growth operator can be converted into a corresponding stage-wise pretraining approach. In this work, we consider the gradual post-stacking framework from Reddi et al., 2023, corresponding to the repeated application of growing with start index $L - b + 1$ and block size b. We use this perspective to extend our growing results in Section 5.3 to stacking in Section 5.4.2.

5.3 Understanding Growing in Depth

5.3.1 Pitfalls of loss-preservation-based growing

As discussed in Section 5.1, a common idea for growing in prior work is based on loss or function preservation as a guiding principle – a growth operator is constructed such that it maintains the same loss value or functional behavior as the original smaller model. The intuition is that this can provide a good initialization for the model in terms of the loss, and that hopefully translates to the final performance. Our work challenges the idea of loss preservation for growing in depth. To put this to the test, we consider a list of depth-growing strategies inspired by prior work (discussed in Section 5.2.1) and measure the correlation between the initial and final loss values across these strategies.

Growing BERT. We begin by pretraining BERT-Base for 500,000 steps. See Appendix C.1 for training details. After 500,000 steps, we grow the model from 12 layers to 16 layers. We consider the abstract design space formalized in Section 5.2.1, instantiated in this particular setting. Specifically, we consider indices 0, 2, 4, 6, and 8 (skipping odd indices simply due to compute limitations – odd indices are equally valid choices), block sizes 1, 2, and 4, and initialization schemes *random* and *parameter duplication*. This results in a search space of 30 different growth operators.

Correlation of initial loss with final loss. For each of the 30 growth operators G, we apply G to the 12-layer model M_1 to initialize a 16-layer model M_2. We then continue training M_2 for 100,000 steps, ensuring that each 16-layer model sees data in the same order, to avoid artifacts of data order as best as possible. We load the optimizer state from the previous stage and maintain a constant learning rate of 0.0001. We measure the validation loss upon initialization of M_2 (i.e., at step 500,000), as well as after 100,000 steps of training M_2 (i.e., at step 600,000). In Figure 5.2 (left), for each growth operator, we plot the validation loss at step 600,000 vs. the validation loss at step 500,000. Visually, we can see that the loss immediately after applying the growth operator is not well-correlated with the loss after continued training for 100,000 steps. Numerically, the Pearson correlation between the validation losses at step 600,000 and the validation losses at step 500,000 is -0.515, and the corresponding Spearman correlation is -0.418.

At step 500,000, the 12-layer model has a validation loss of 1.823. Therefore, although our search space does not include pure loss-preserving growth operators, some of the growth operators do come quite close (e.g., loss approximately 1.825 vs. 1.823), and others only raise the validation loss a bit in comparison with other growth operators in the search space. From this, we can see that approximate loss preservation does not appear to correlate with final performance.

Implications. Given that loss preservation is not a good heuristic to predict the final model performance, is there another approach that is more predictive? In this rest of the section, we provide a series of empirical analyses suggesting that the early loss landscape can provide a much stronger indicator of the final loss.

5.3.2 Strong correlation with final performance emerges early

While the initial loss is not very predictive of final performance, we make a surprising discovery that the loss after a few steps of training can be very highly predictive.

Hypothesis: Loss after some steps of training strongly correlates with the final loss.

For each growth operator G, we measure the validation loss at step 505,000 (i.e., after 5,000 steps of training M_2). In Figure 5.2 (right), for each growth operator, we plot the validation loss at step 600,000 vs. the validation loss at step 505,000. Visually, we can see that the loss after 5,000 steps *is highly correlated* with the loss at step 600,000. Numerically, the Pearson correlation between the validation losses at step 600,000 and the validation losses at step 505,000 is 0.982, and the corresponding Spearman correlation is 0.986. In other words, we see that the the final order of the various growing strategies has largely already emerged within the first 5,000 steps. Thus, in this BERT setting, we have strong support for our hypothesis.

The early loss landscape perspective. In general, it can be difficult to predict final performance based on initial *or* early performance. However, we posit that, when *growing* an already-trained model, the early loss landscape has particularly nice properties amenable to early prediction. One possible mental model is as follows. Upon growing, the network enters an unstable state induced by the addition of its new layers. However, since the network is already largely trained, it is able to adapt quickly to use its new layers, resulting in a fast drop in the loss. During this initial fast adaptation phase, the overall ordering of various growing strategies is still unstable. However, once this fast adaptation to the new layers is complete, the loss curves

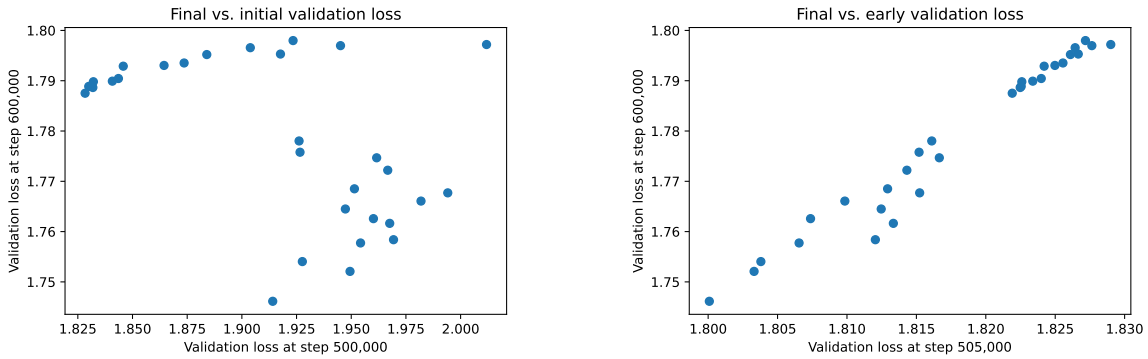


Figure 5.2: Growing a 12-layer BERT model at step 500,000 into a 16-layer BERT model and then training the larger model for 100K steps. Correlation (in validation loss) between (left) the loss at 600K steps and the loss immediately upon growing (i.e., without any training) and (right) the loss at 600K steps and the loss at 505K steps (i.e., after 5,000 steps of training the larger model). Although the correlation at the beginning is low (left plot), the correlation quickly rises after just 5,000 steps of training (right plot).

and, crucially, their relative orderings enter a more stable phase. This can be summarized as:

Phase 1: At initialization, look for a nearby point in the loss landscape that is much better adapted to using the new layers (resulting in a rapid drop in the loss).

Phase 2: Continue training from this adapted initialization (now in a slower, more predictable manner).

This perspective suggests that the initial loss is not the only factor in determining the final performance of a particular growth operator. Rather, how the growth operator influences the network's loss landscape near its new initialization is also crucial in determining its final performance. We call this perspective *landscape-aware growing*.

5.3.3 Prediction within several hundred steps

Based on our insights in Section 5.3.2, we ask: *How early does good prediction become possible?* We zoom in on the first 200 steps of training after growing (i.e., steps 500,000 to 500,200) and observe several interesting properties.

Self-correlation heatmap. For every pair of steps $(i, j) \in \{500000, \ldots, 500200\}^2$, we compute the Spearman correlation between the validation losses at step i and the validation losses at step j. Figure 5.3 (top left) displays this as a correlation heatmap. In this heatmap, we can see a clear phase transition between the earliest few steps of training (whose validation losses do not correlate well with the losses at step 500,200) and the remainder of the first 200 steps, which have strong correlation amongst themselves. This phase transition between the earliest few steps of training and the remainder of the first 200 steps seems to occur roughly around step 500,050.

Correlation with final performance. For every step $i \in \{500000, \ldots, 500200\}$, we compute the Spearman correlation between the validation losses at step i and the validation losses at step 600,000. We observe that the Spearman correlation rises rapidly within the first 200

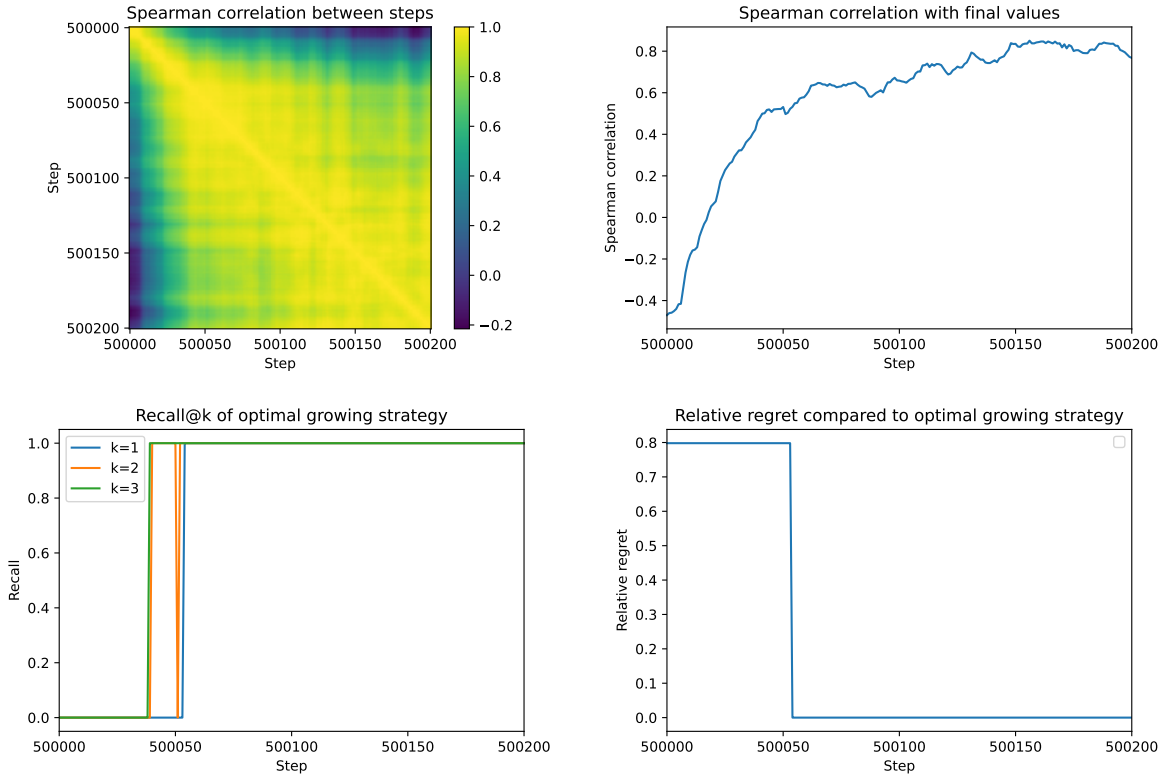


Figure 5.3: Growing BERT from 12 to 16 layers: zooming in on steps 500,000 through 500,200. Spearman correlation heatmap (top left), Spearman correlation with final values (top right), Recall@k (bottom left), Relative regret (bottom right). See Section 5.3.3 for details on how these plots were constructed. For all plots, at each step, the validation loss is first averaged over a window of 11 steps (centered at the step in question) to help smooth out noise.

steps, from less than -0.4 to approximately 0.8. This indicates that strong correlation with final performance emerges very early on in training, providing some support for the conjecture that the phase transition observed above is, in fact, a transition into a more stable phase with high predictive power with respect to final performance.

Measuring Recall@k. Practically speaking, however, we do not need to predict the *full ranking* of strategies. In fact, predicting the exact ordering among the worst-performing strategies has very limited utility. Rather, a practitioner could exploit the above insights by pruning their search space down to just a few top configurations. Then, one practically-relevant question is: Was the best growing strategy within this pruned set? We study this question in the following manner. We first determine which growing strategy has the smallest validation loss at step 600,000, and we let $G^\star$ denote this growth operator. Then, at every step $i \in \{500000, \ldots, 500200\}$, we identify the k growth operators with the smallest validation loss and ask whether $G^\star$ is among these k growth operators; if it is, we plot 1, and if not, we plot 0. Figure 5.3 (bottom left) displays this plot for $k = 1, 2, 3$. As can be seen in the figure, the recall rises to 1 for all values of k around step 500,050, which aligns with the phase transition identified in the correlation heatmap. This provides further support for the conjecture that the phase transition observed above is a transition into a more stable phase with high predictive power with respect to the optimal growing strategies.

Notion of regret. Beyond just identifying or failing to identify the optimal growing strategy at a particular step is the more nuanced question: *how* suboptimal (with respect to validation loss) is it to choose a growing strategy based on its performance at step i (vs. its performance at the end of training). Let G^i denote the growth operator with the smallest validation loss at step i, and let $\ell(G)$ denote the final validation loss at step 600,000 after growing with operator G at step 500,000. Among all growing strategies in the search space, let ℓ^{min} denote the smallest validation loss at step 600,000, and let ℓ^{max} denote the largest validation loss at step 600,000. Then, for every step $i \in \{500000, \ldots, 500200\}$, we can calculate the regret as

$$\ell(G^i) - \ell^{\text{min}} \tag{5.1}$$

and the relative regret as

$$(\ell(G^i) - \ell^{\text{min}})/(\ell^{\text{max}} - \ell^{\text{min}}). \tag{5.2}$$

In other words, the relative regret captures how suboptimal it is to choose a growing strategy based on its validation loss at step i. Figure 5.3 (bottom right) displays the relative regret for $i \in \{500000, \ldots, 500200\}$. We can see that the relative regret starts at approximately 0.8 and drops to 0 around step 500,050, which aligns with the phase transition identified in the correlation heatmap. This provides further support for the conjecture that the phase transition observed above is a transition into a more stable phase with high predictive power with respect to identifying low-loss strategies.

Conclusion. Taken together, these results demonstrate that, soon after growing, (1) an approximate ordering of the growing strategies emerges and (2) it is possible to identify a low-regret strategy.

Table 5.1: Performance of LAG@200 compared to other growing strategies, when growing BERT from 12 layers to 16 layers. See equations 5.1 and 5.2 for the definitions of regret and relative regret. The "Oracle" strategy refers to the best possible strategy within the search space, based on validation loss at step 600,000. LAG@0 roughly follows the "loss preservation" heuristic (i.e., choosing the strategy whose loss is least perturbed by growing). The final two rows most closely resemble gradual stacking (Reddi et al., 2023): (1) stacking the last block on top, and (2) stacking a randomly-initialized block on top. Overall, LAG@200 is able to identify the best-performing strategy, achieving a relative regret of 0. In contrast, the other strategies have a relative regret of at least 0.5 (and even higher).

Strategy	Final Validation Loss	Regret	Relative Regret
Oracle	1.7461	0	0
LAG@200	1.7461	0	0
Best at initialization (LAG@0)	1.7875	0.0414	0.7986
Stack last block on top	1.7747	0.0286	0.5517
Stack random block on top	1.7875	0.0414	0.7986

5.4 Applications

In this section, we extend the insights of Section 5.3 into algorithms for growing and stacking. Our goal is primarily to validate the landscape-aware theory through its algorithmic utility when applied in a very simple manner. We believe that more sophisticated algorithms could be developed to further exploit the landscape-aware theory, with even stronger performance, and that this is merely evidence of a step in the right direction algorithmically.

5.4.1 LAG

We define Landscape-Aware Growing, or LAG@k, as a simple algorithm for growing as follows. Consider a design space $\mathcal{G}$ of growth operators. For each growth operator $G \in \mathcal{G}$, apply G to the pretrained model M_1 to obtain a larger model M_2. Train each such M_2 for k steps, for some small k. Choose the growth operator $\hat{G}$ yielding the lowest validation loss at k steps, and then train $\hat{G}(M_1)$ to completion. Although LAG@k is fairly generic and can be instantiated with various values of k, for the purposes of this evaluation, we look for a phase transition as in Figure 5.3 (top left) and choose k to ensure some margin post-phase-transition.

BERT. We first apply LAG to the BERT-Base setting defined in Section 5.3.1. Given the phase transition around step 500,050, we use LAG@200 (though smaller values of k would have similar behavior here, based on the results in Figure 5.3).

In Table 5.1, we compare the performance of LAG@200 with several other methods. The "Oracle" strategy refers to the best possible strategy within the search space, based on validation loss at step 600,000 (i.e., it is not a practical strategy but rather represents the best one could hope to achieve). We also compare to LAG@0; since the loss of $G(M_1)$ is higher than the loss of M_1 for all growth operators G, choosing the growth operator with the lowest loss after 0 steps of training follows the "loss preservation" heuristic (i.e., choosing the strategy whose loss

Table 5.2: Performance of LAG@2000 compared to other growing strategies, when growing UL2 from 12 layers to 16 layers. See equations 5.1 and 5.2 for the definitions of regret and relative regret. The "Oracle" strategy refers to the best possible strategy within the search space, based on validation loss at step 400,000. LAG@0 roughly follows the "loss preservation" heuristic (i.e., choosing the strategy whose loss is least perturbed by growing). The final two rows most closely resemble gradual stacking (Reddi et al., 2023): (1) stacking the last block on top, and (2) stacking a randomly-initialized block on top. Overall, LAG@2000 achieves a relative regret of 0.0895, which is much smaller than that of the alternative methods.

Strategy	Final Validation Loss	Regret	Relative regret
Oracle	2.1254	0	0
LAG@2000	2.1268	0.0014	0.0895
Best at initialization (LAG@0)	2.1398	0.0144	0.9202
Stack last block on top	2.1357	0.0103	0.6582
Stack random block on top	2.1398	0.0144	0.9202

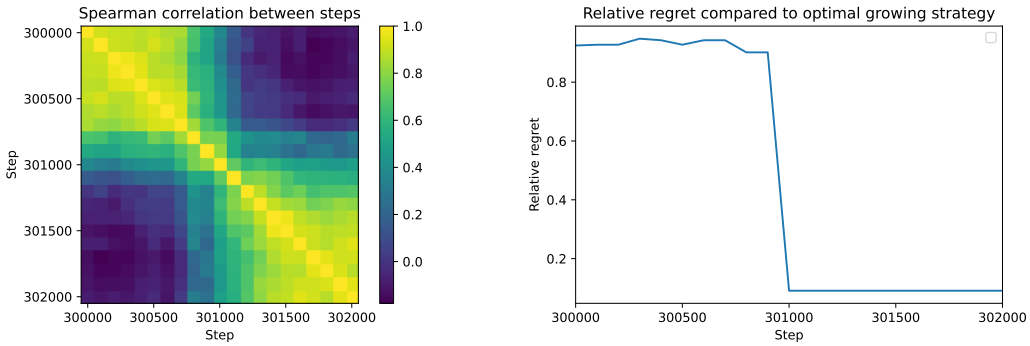


Figure 5.4: Growing UL2 from 12 layers to 16 layers: zooming in on steps 300,000 through 302,000. Spearman correlation heatmap (left) and Spearman correlation with final values (right). See Section 5.3.3 for details on how these plots were constructed. Here, the validation loss is only measured every 100 steps, so these plots do not use smoothing (in contrast with Figure 5.3).

is least perturbed by growing). We also compare to the variants within our search space which most resemble gradual stacking (Reddi et al., 2023): (1) stacking the last block on top, and (2) stacking a randomly-initialized block on top. Due to the limited effect of new final layers with small random initialization, a randomly-initialized block on top can be viewed as another proxy for the "loss preservation" heuristic and does turn out to match LAG@0.

Overall, with LAG@200, we see that we are able to identify the best-performing strategy, achieving a relative regret of 0. In contrast, the other strategies have a relative regret of at least 0.5 (and even higher).

UL2. We also extend LAG to the UL2 setting. Specifically, we begin by pretraining a 12-layer decoder-only model with 1.1B parameters and the UL2 objective (Tay et al., 2022) for 300,000 steps, before growing it to 16 layers and training for an additional 100,000 steps. See further details in Appendix C.1. Since we are growing from 12 to 16 layers as we did in the BERT setting,

we include the same 30 growth operators that we studied for BERT in Section 5.3.1. We compute the correlation heatmap as defined in Section 5.3.3 over the first 2,000 steps immediately after growing (i.e., step 300,000 through step 302,000) and observe a phase transition around step 1,000. To ensure a margin post-phase-transition, we choose $k = 2000$, i.e., LAG@2000. In Table 5.2, we compare the performance of LAG@2000 with the same alternative methods we examined for BERT, above. Overall, we see that LAG@2000 achieves a relative regret of 0.0895, which is fairly small and much smaller than that of the alternative methods. We also include 1-shot evaluations for several key downstream tasks in Appendix C.2, providing evidence that the trends in validation loss hold for downstream metrics as well.

5.4.2 Adaptive stacking

Here, we consider how to apply LAG to gradual stacking. Specifically, LAG motivates an *adaptive* strategy for gradual stacking: at each stage of stacking, multiple growing strategies are spawned in parallel and trained for k steps, for some small k. Then, the strategy with the lowest validation loss is chosen and training is continued for the rest of the stacking stage using just this one strategy. Applied iteratively, with n stages and s strategies explored per stage, this costs $n \times k \times s$ additional steps of training (divided over a range of model sizes). Here, we use adaptive stacking to train a 24-layer BERT-LARGE model in 6 stages, using a roughly uniform stacking schedule (160,000 steps per stage for the first 5 stages, and 200,000 steps in the final stage). We use $k = 200$ steps. In Table C.2 in Appendix C.2, we see that adaptive stacking outperforms last stacking (Reddi et al., 2023), with a final validation loss of 1.5301 for adaptive stacking vs. 1.5432 for last stacking (i.e., post-stacking in Reddi et al., 2023).

We caveat that this is merely intended as a demonstration of how one might naively apply the principle of LAG to stacking. Without further iteration, we can already see gains over fixed stacking, suggesting that this could be a promising direction for future work on improving stacking (or "iterated growing", more generally).

5.5 Related Work

The literature on *growing models* is vast; hence, we only focus on the most relevant works here. Net2Net (Chen et al., 2016) was one of the first works to popularize parameter-reusing growth operators for neural networks, though notably building on much earlier works such as (Fahlman and Lebiere, 1989) and (Gutstein et al., 2008). The primary technical contribution of (Chen et al., 2016) is its *function-preserving* growth operators, which ensure that the new, larger network at initialization represents exactly the same function as the original, smaller network. Specifically, among its main contributions, Chen et al. (2016) highlights how function-preserving growth operators avoid "spending time passing through a period of low performance". Since 2016, various works have built upon this *function-preserving* idea, and it has become a core tenet in the design of neural network growth operators (Wei et al., 2016; Evci et al., 2022).

More recent works in this direction are largely based on Transformers. Chen et al., 2022, Shen et al., 2022, Li et al., 2023a, Yao et al., 2024, and Wang et al., 2024 all explore different variants of

function- or loss-preserving growth operators specifically in the Transformer setting. Despite differences in how they achieve function preservation, they are all united by the perspective - dating back to Net2Net - that "a growth operator that is not loss-preserving wastes time and compute initially until it recovers the same performance of the original model" (Wang et al., 2024). It is this design principle that our work provides compelling evidence against.

Although the primary focus of our work is how to expand a shallow model into a deeper model through one step of growing, we also explore its application to iterated growing. Our work is thus related to the literature on progressive stacking (Gong et al., 2019) and gradual stacking (Reddi et al., 2023), which gradually increase the model depth in stages by reusing layers from the previous stage.

5.6 Conclusion

Overall, we have conducted a fairly extensive empirical analysis of various growth operators and identified that, despite a vast body of prior work on function- and loss-preserving growth operators, initial loss immediately upon growing is not particularly predictive of final performance. Rather, allowing a growth operator to initially disrupt the function (and thus the loss) can actually be desirable if it then leads to a more favorable early loss landscape in which the loss can decrease more rapidly. To that end, we identify that this notion of "early loss landscape" is actually quite early in BERT and is delineated by a measurable phase transition. Based on these insights, we introduce Landscape-Aware Growing (LAG) as a general, design-space-agnostic strategy for growing; with just a little "lag" after initialization, identifying a low-regret growth operator is possible. We validate our approach in UL2 and extend our insights to stacking as well.

Although these results are quite exciting, we find it prudent to point out various limitations beyond the caveats already noted. One limitation is that, due to compute constraints, we could only run one trial per growth operator. We hope that by validating our results in a different, more complex setting (UL2), we mitigate these concerns a bit; however, in an ideal world, we would have multiple trials with different random initializations and different data orders. Another limitation is that we have only explored the BERT and UL2 settings, and our largest model sizes are just over 1B parameters; thus, we do not know how our results generalize to state-of-the-art model sizes. Finally, due to compute constraints, our search space is naturally limited to a subset of all possible growth operators. We have tried to capture a range of both approximately loss-preserving and loss-disrupting growth operators, and we thus believe our search space is quite reasonable with respect to our conceptual insights; however, future work could consider expanding the search space to include even more growth operators from the literature.

Chapter 6

Conclusion

In the preceding pages, we have examined the role of structure in the success of modern deep learning. We began by considering whether classical generalization bounds - specifically algorithmic stability, in our case - could explain deep learning's impressive generalization performance. We saw empirical evidence that the flexibility of neural networks optimized via gradient descent is perhaps beyond the limitations of such a framework. However, when paired with structure, this mystifying flexibility becomes a gift: adaptability.

Therefore, rather than pursuing a general theory of why deep learning works right now, we advocate for the more fruitful path of understanding how gradient-descent-optimized neural networks can adapt to and from structure - structure in data, architecture, and initialization. Through three different examples that together trace the evolution of deep learning over the past six years, we see how

1. convolutional neural networks' filters can adapt to signal amidst significant background noise in images
2. how the Transformer's self-attention mechanism can adapt to domain-agnostic decision-list-like structure in sequence-to-sequence data, and
3. how pretrained language models with existing structure can adapt when new Transformer blocks are inserted to increase their depth.

We believe that analyzing structures like these in detail, as we have done in this thesis, is key to understanding the success of modern deep learning.

Now, in summer 2024 - at the time of concluding this thesis - this message seems as timely as ever. With ever-increasing dataset sizes and model sizes and the overall rise of foundation models, some of the most pressing questions are now less about whether these models will reach low training loss or have good performance on unseen in-distribution data and instead more about what representations/underlying structures are being learned and how this will enable adaptation to new tasks.[1]

In the near-term, some key questions include:

[1] whether zero-shot, with in-context examples, or after some limited fine-tuning

1. What structures are self-attention blocks learning from data?
2. Which aspects of self-attention are important when solving real-world tasks?
3. What is the role of MLPs in Transformers?
4. How can we design better data mixtures and training curricula to help Transformer-based language models learn these structures more efficiently?

In the longer term, as we move toward realizing the dream of more agentic, interactive artificial intelligence, the practical applications of machine learning are becomingly increasing highly multitask. A split has been forming between the pretraining of foundation models on diverse Internet-scale data and their subsequent adaptation to many different downstream tasks of interest. In some cases, these downstream tasks involve no parameter modifications – just the clever tuning of inputs. In other cases, the foundation models' parameters are modified through various fine-tuning techniques to further adjust their performance. In all of these settings, adapting to and from structure remains key. Foundation models are only useful insofar as they have adapted their parameters to represent the underlying structure in their vast pretraining data. And it is these learned structures that enable efficient adaptation to diverse downstream tasks, through various post-training approaches such as in-context learning, supervised fine-tuning, reinforcement learning, etc. Some key questions are therefore: What structures are being learned through the particular combination of large-scale data and architecture? How do these structures enable efficient adaptation to new tasks, and how can we learn even better structures for more efficient adaptation?

In this thesis, we have looked at single tasks (adapting to the structure in single tasks and using this structure to adapt new parameters to the same task). In the coming agentic, multitask settings, we believe that analogous approaches will similarly prove fruitful. As in this thesis, we will likely get further, faster by trying to understand the actual, empirically-supported interplay between structure in data and the structures learned by particular neural network architectures – how neural networks adapt to structure in data, and how they use their learned structures to adapt to new data, in particular, illustrative settings. This is in contrast to a more black-box approach such as that explored in Chapter 2. We are optimistic that the lessons and themes learned from and illustrated in this thesis can carry us forward into the next phase of machine learning and artificial intelligence research.

Appendices

Appendix A

Algorithmic Stability: Additional Details

A.1 Further Methodology Details

Dataset splits. MNIST: From the 60,000 training examples, we randomly sampled subsets as specified in Section 2.2 for training. We used the full 10,000-element test set for evaluation (including computation of uniform stability, as specified in Section 2.2).

CIFAR-10: From the 50,000 training examples, we randomly sampled subsets as specified in Section 2.2 for training. We used the full 10,000-element test set for evaluation (including computation of uniform stability, as specified in Section 2.2).

SVHN: From the 73,257 training examples, we randomly sampled subsets as specified in Section 2.2 for training. To maintain consistency with MNIST and CIFAR-10, we randomly sampled 10,000 elements from the 26,032-element test set for evaluation. All datasets were normalized in the same manner, by dividing each coordinate by 255.

Selection of hyperparameters. The hyperparameters for ResNet-20 on CIFAR-10 are derived from He et al. (2016). The hyperparameters for logistic regression were chosen similarly, intentionally without momentum (since our primary goal was to study SGD) and without a decaying learning rate. The hyperparameters for Configuration 1a were intentionally chosen to vary from Configurations 2a and 2b, in order to create more diversity in our hyperparameter settings; specifically, we deemed it valuable to investigate a smaller batch size (i.e., 32) without momentum, and the corresponding learning rate of 0.01 worked fairly well with this batch size.

Stopping criterion details. We considered three different possible stopping criteria: parameter updates, epoch number, and average training loss. We performed preliminary analyses with all stopping criteria, but after careful consideration, we ultimately chose to focus our analysis on *parameter updates* for the following reasons: (1) parameter updates align with theoretical analyses of uniform stability, such as Hardt et al. (2016) and Feldman and Vondrak (2019), in which the stability parameter is expressed as a function of the number of parameter updates;

and (2) parameter updates appear to give uniform stability the *best* chance at succeeding in explaining generalization, thus making our *negative* results more significant. Specifically, if instead we were to hold the number of epochs fixed across dataset sizes, this means that *larger* datasets would take more steps. This is true for average training loss as a stopping criterion as well, as it typically takes more steps for larger datasets to reach the same average training loss as smaller datasets. Thus, although these stopping criteria are perhaps truer to practice, we believe that they make it even *easier* for uniform stability to fail to explain the strength of generalization.

Dataset size range. We chose to limit our analysis to the ranges specified in Section 2.2 for the following reason. In order to ask the question *Can the strength of decrease with m in our generalization gap be explained by uniform stability?*, we wanted a rate of decrease with m that would be roughly constant in our dataset size range. Figure A.1 shows a plot and curve fit on a normal-scale plot, followed by a log-log plot. Although the curve fit displays some room for improvement in the original plot, the log-log plot reveals different regions of decrease with m. Through this plot and additional such investigations, we noticed that the dataset size range 15,000-50,000 yielded the largest window with a roughly consistent rate of decrease with m. Thus, we chose to focus our analysis on this window in order to draw more meaningful conclusions. As deep learning models are typically trained in large-data regimes, this decision aligns with practical considerations as well.

Curve fitting details. We used `scipy`'s `optimize` package, specifically the `curve_fit` function. We fit parameters a and b in $y = am^b$, where m is the dataset size and y is the metric of interest.

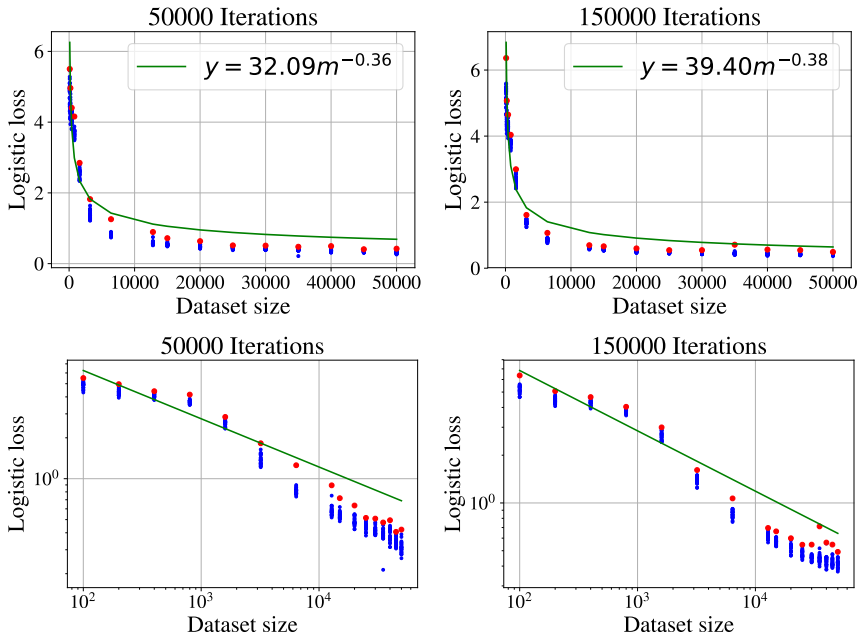


Figure A.1: Configuration 1a, with 20 samples per dataset size m. The left column is generated using 50,000 iterations of training, and the right column is generated using 150,000 iterations of training. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. The top plot is a normal-scale plot, and the bottom plot is the same data plotted as a log-log plot. The curve fit displays some room for improvement in the original plot, and the log-log plot reveals different regions of decrease with m. These plots help us find a dataset size range where the loss has a roughly consistent rate of decrease with m.

A.2 Further Experiments for Section 2.3

In this section, we present stability and generalization results for two additional seeds (Trials 2 and 3) and compare them to the original seed presented in the main paper (Trial 1). Our figures are as follows:

- Figure A.2 has all trials for Configuration 1a (SVHN).
- Figure A.3 has all trials for Configuration 2a (CIFAR-10, no momentum).
- Figure A.4 has all trials for Configuration 2b (CIFAR-10, 0.9 mometum).
- Figure A.5 has generalization (cross-entropy only) and stability results for Iteration 50,000 for Configuration 1a.
- Figure A.6 has generalization (cross-entropy only) and stability results for Iteration 50,000 for Configuration 2a.
- Figure A.7 has generalization (cross-entropy only) and stability results for Iteration 50,000 for Configuration 2b.
- Figure A.8 has generalization (cross-entropy only) and stability results for Iteration 150,000 for Configuration 1a.
- Figure A.9 has generalization (cross-entropy only) and stability results for Iteration 150,000 for Configuration 2a.
- Figure A.10 has generalization (cross-entropy only) and stability results for Iteration 150,000 for Configuration 2b.

The additional trials are roughly consistent with the trial highlighted in the main paper.

A.3 Further Experiments for Section 2.4

In this section, we present further experiments regarding regarding the Euclidean distance between $\mathcal{A}(S)$ and $\mathcal{A}(S')$, parameter norms, and normalized Euclidean distances. Our figures are as follows:

- Figure A.11 presents additional trials for $\|\mathcal{A}(S) - \mathcal{A}(S')\|_2$ at Iteration 100,000.
- Figure A.12 has $\|\mathcal{A}(S)\|_2$ at Iteration 100,000.
- Figure A.13 has normalized Euclidean distances, with further details in the caption.

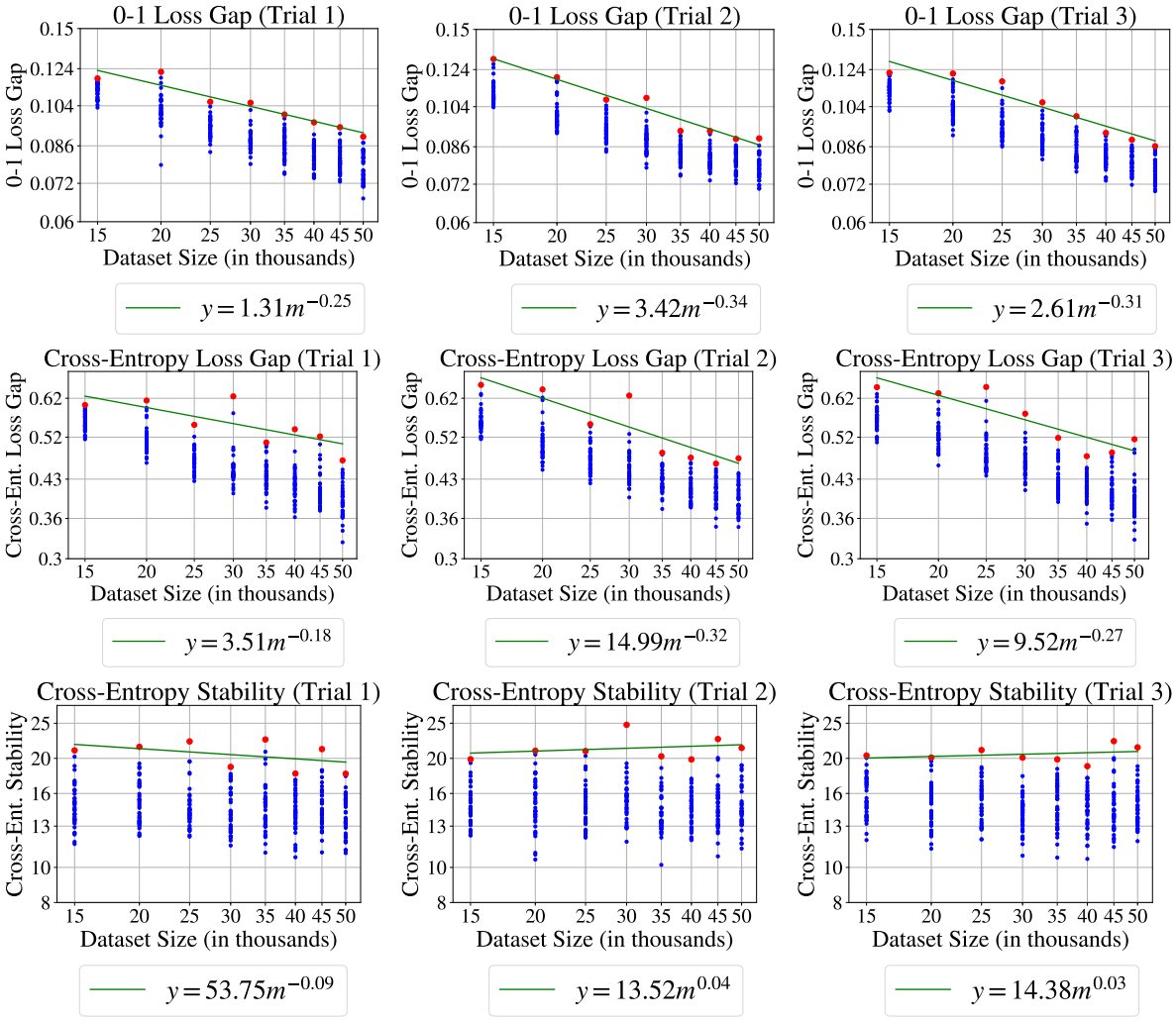


Figure A.2: All trials for Configuration 1a (SVHN). Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the 0-1 loss generalization gap as a function of dataset size, row 2 is the cross-entropy loss generalization gap as a function of dataset size, and row 3 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

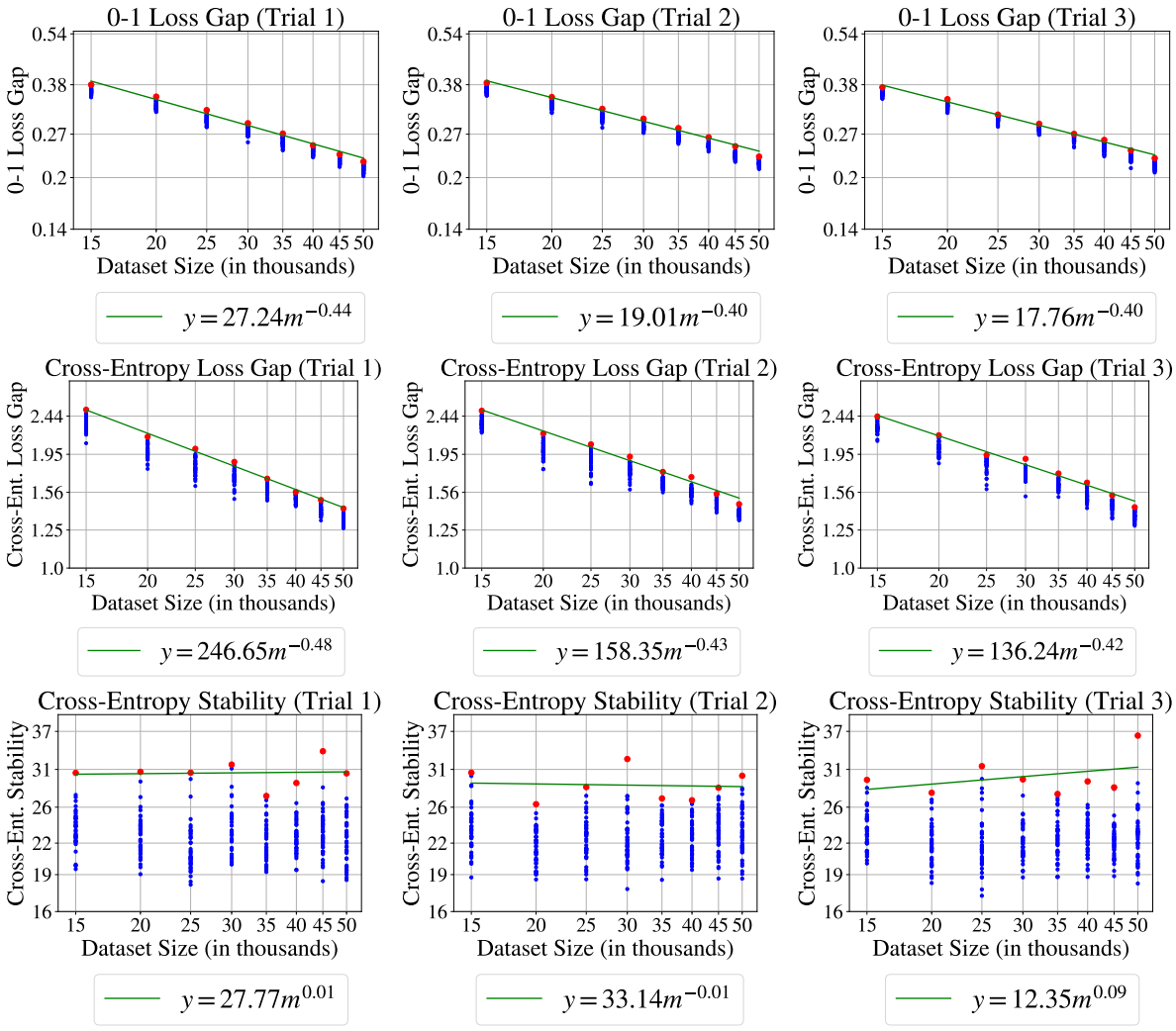


Figure A.3: All trials for Configuration 2a (CIFAR-10, no momentum). Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the 0-1 loss generalization gap as a function of dataset size, row 2 is the cross-entropy loss generalization gap as a function of dataset size, and row 3 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

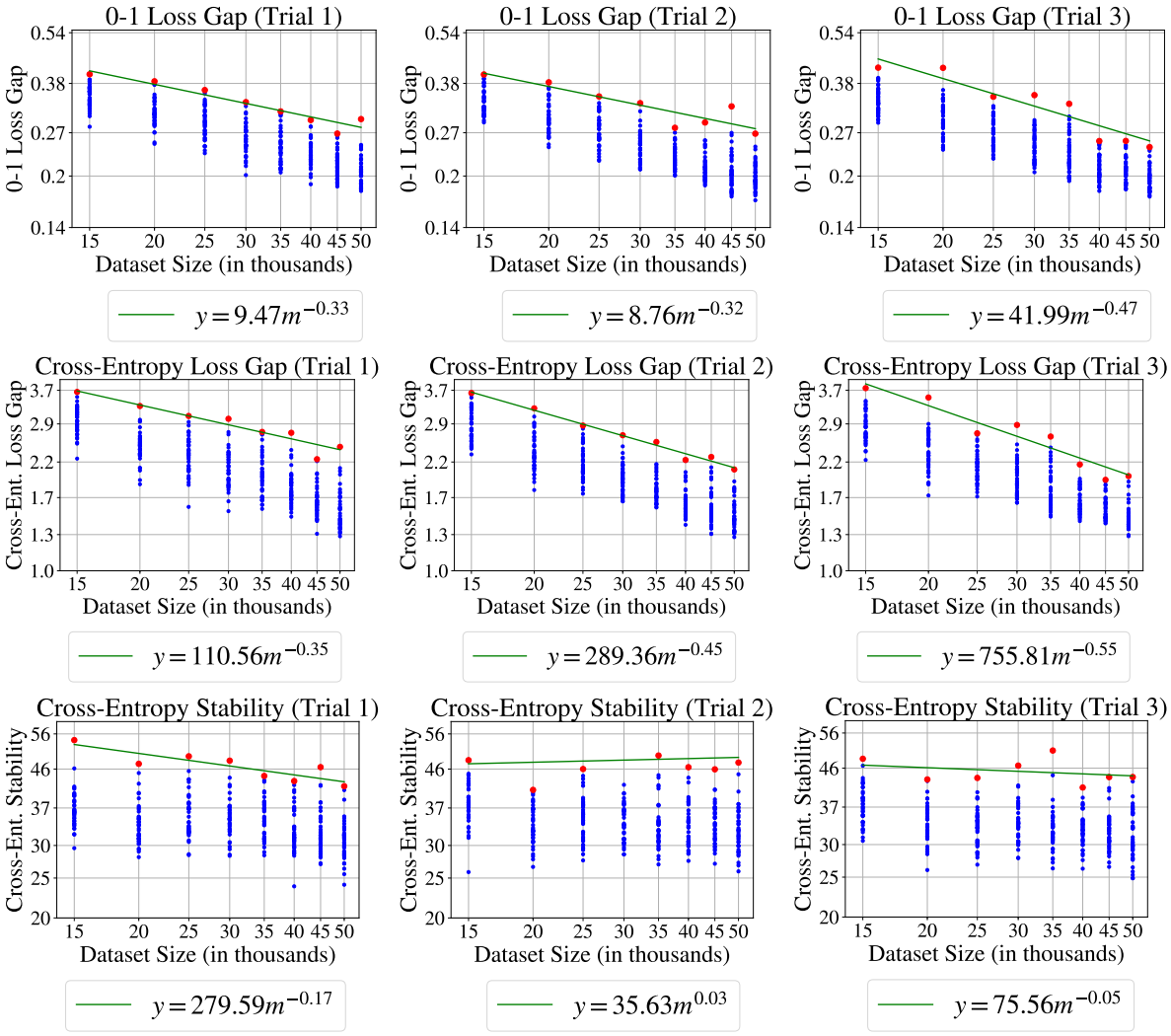


Figure A.4: All trials for Configuration 2b (CIFAR-10, 0.9 momentum). Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the 0-1 loss generalization gap as a function of dataset size, row 2 is the cross-entropy loss generalization gap as a function of dataset size, and row 3 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

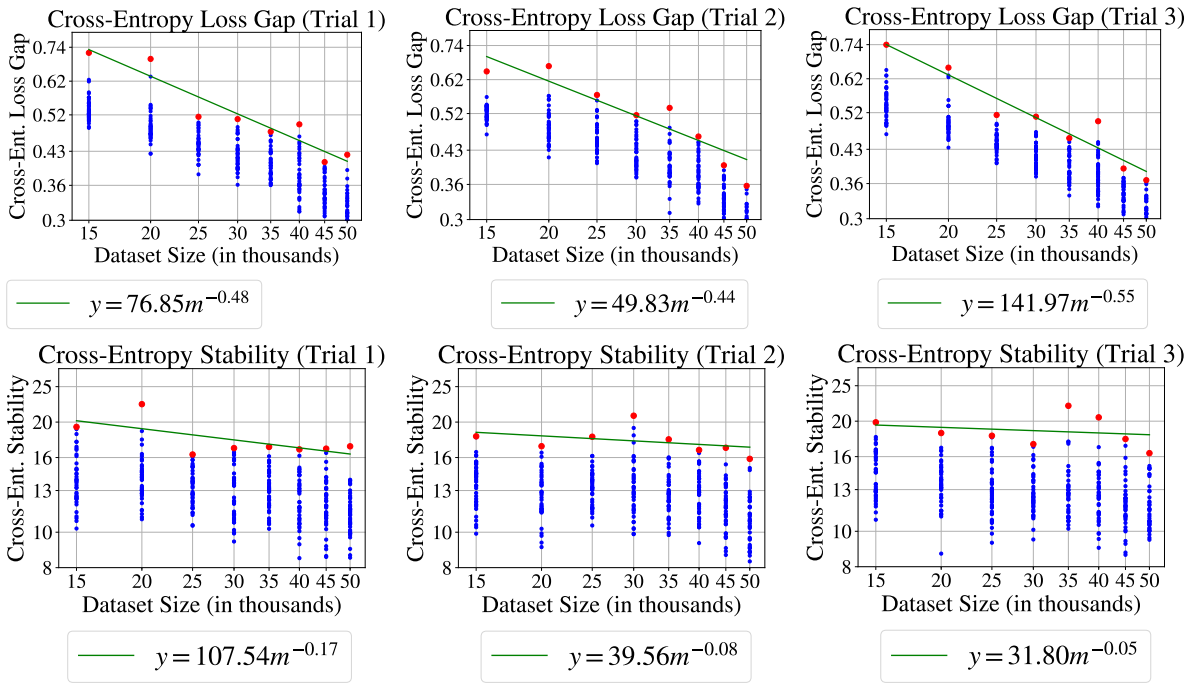


Figure A.5: All trials for Configuration 1a at iteration 50,000. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the cross-entropy loss generalization gap as a function of dataset size, and row 2 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

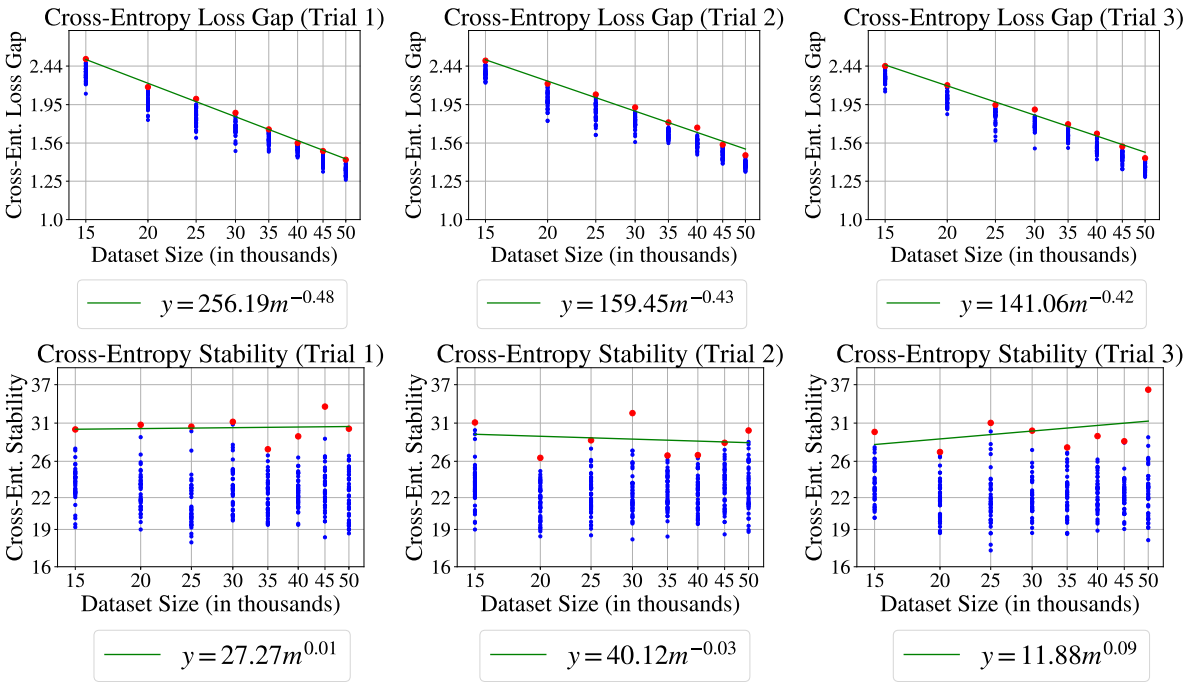


Figure A.6: All trials for Configuration 2a at iteration 50,000. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the cross-entropy loss generalization gap as a function of dataset size, and row 2 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

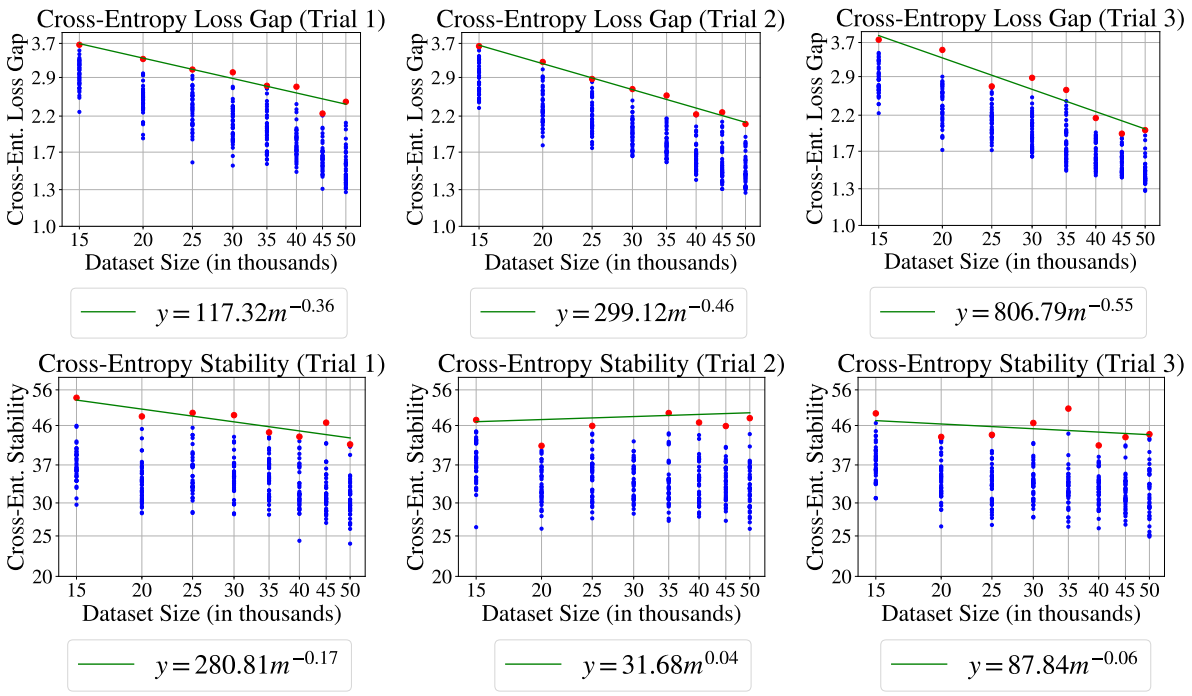


Figure A.7: All trials for Configuration 2b at iteration 50,000. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the cross-entropy loss generalization gap as a function of dataset size, and row 2 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^{m}$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

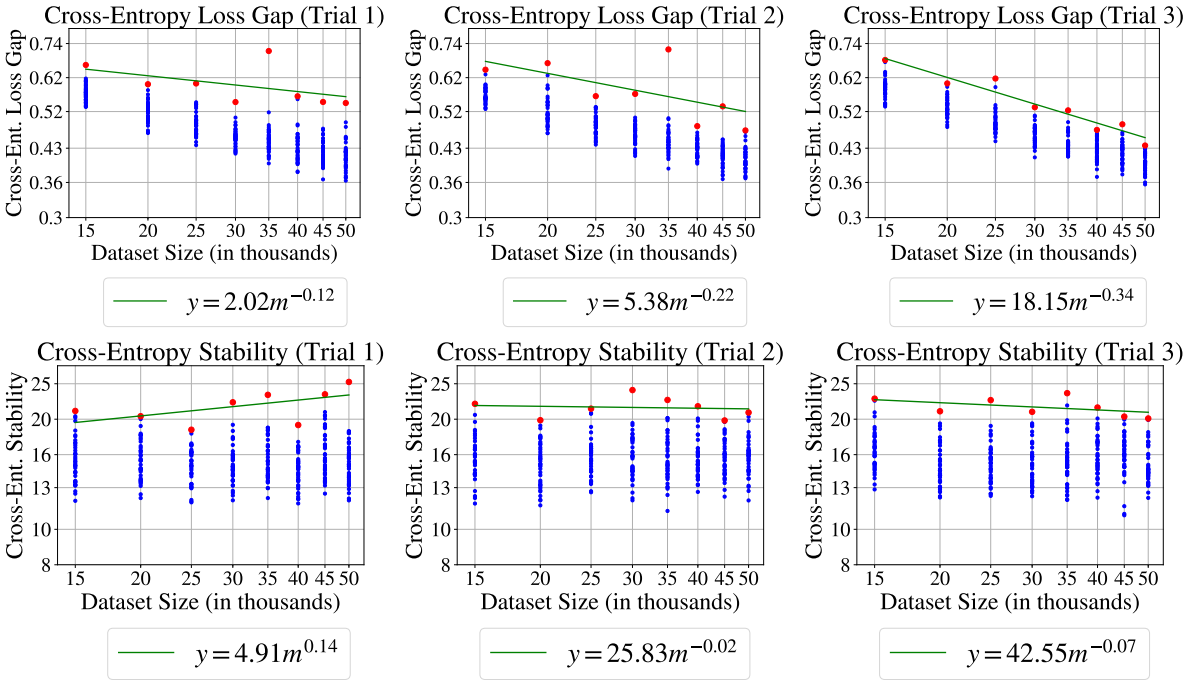


Figure A.8: All trials for Configuration 1a at iteration 150,000. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the cross-entropy loss generalization gap as a function of dataset size, and row 2 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^{m}$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

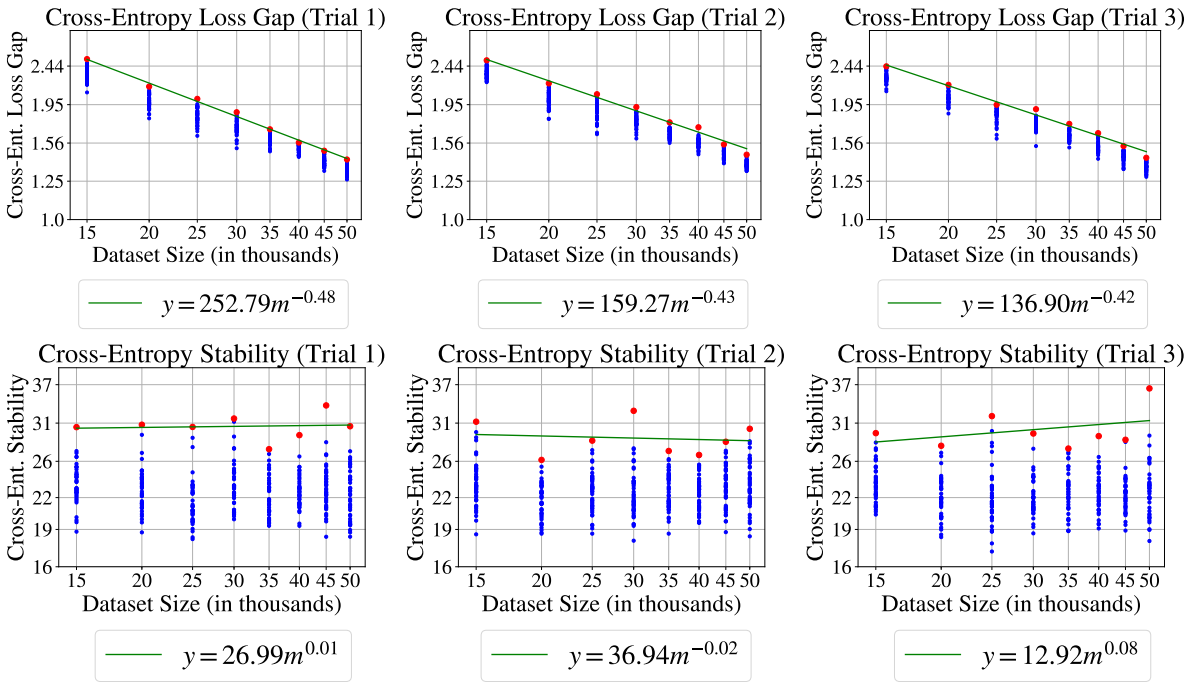


Figure A.9: All trials for Configuration 2a at iteration 150,000. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the cross-entropy loss generalization gap as a function of dataset size, and row 2 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^{m}$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

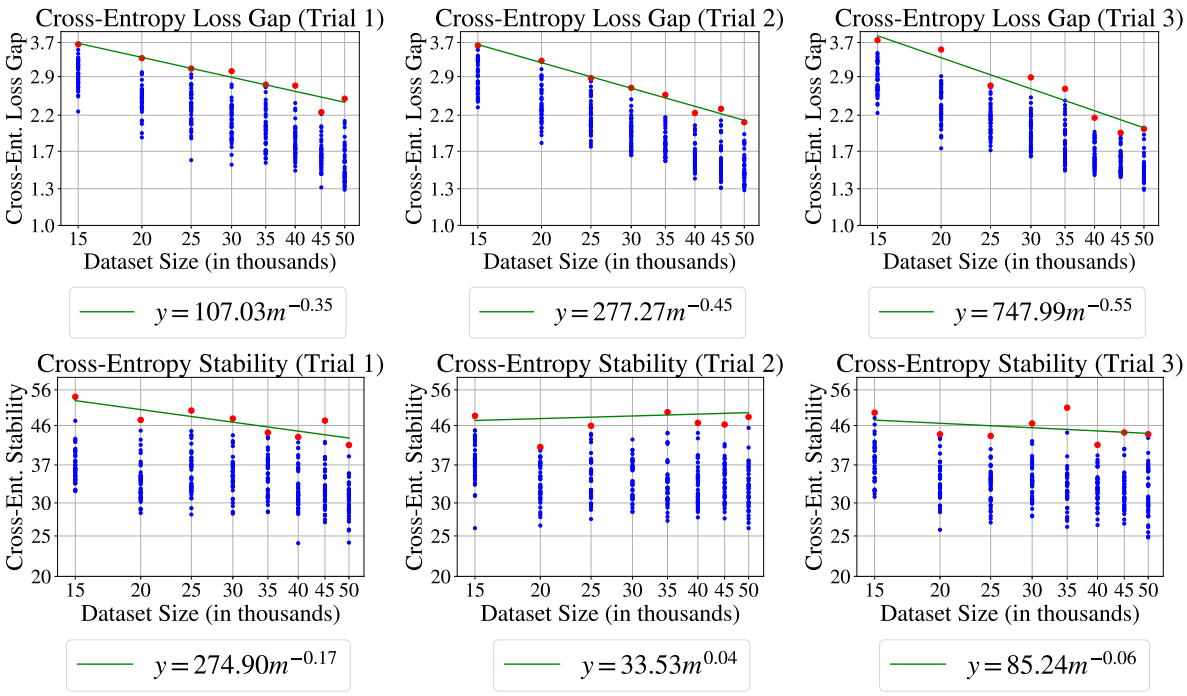


Figure A.10: All trials for Configuration 2b at iteration 150,000. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Row 1 is the cross-entropy loss generalization gap as a function of dataset size, and row 2 is the cross-entropy loss stability as a function of dataset size. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^{m}$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, the maximum value per dataset size m is plotted as a red dot, and a curve of the form $y = am^b$ is fit to the red dots and plotted in green. Overall, across all trials, the cross-entropy loss stability has a dependence on m that is *not* comparable to the generalization gap's decay with m.

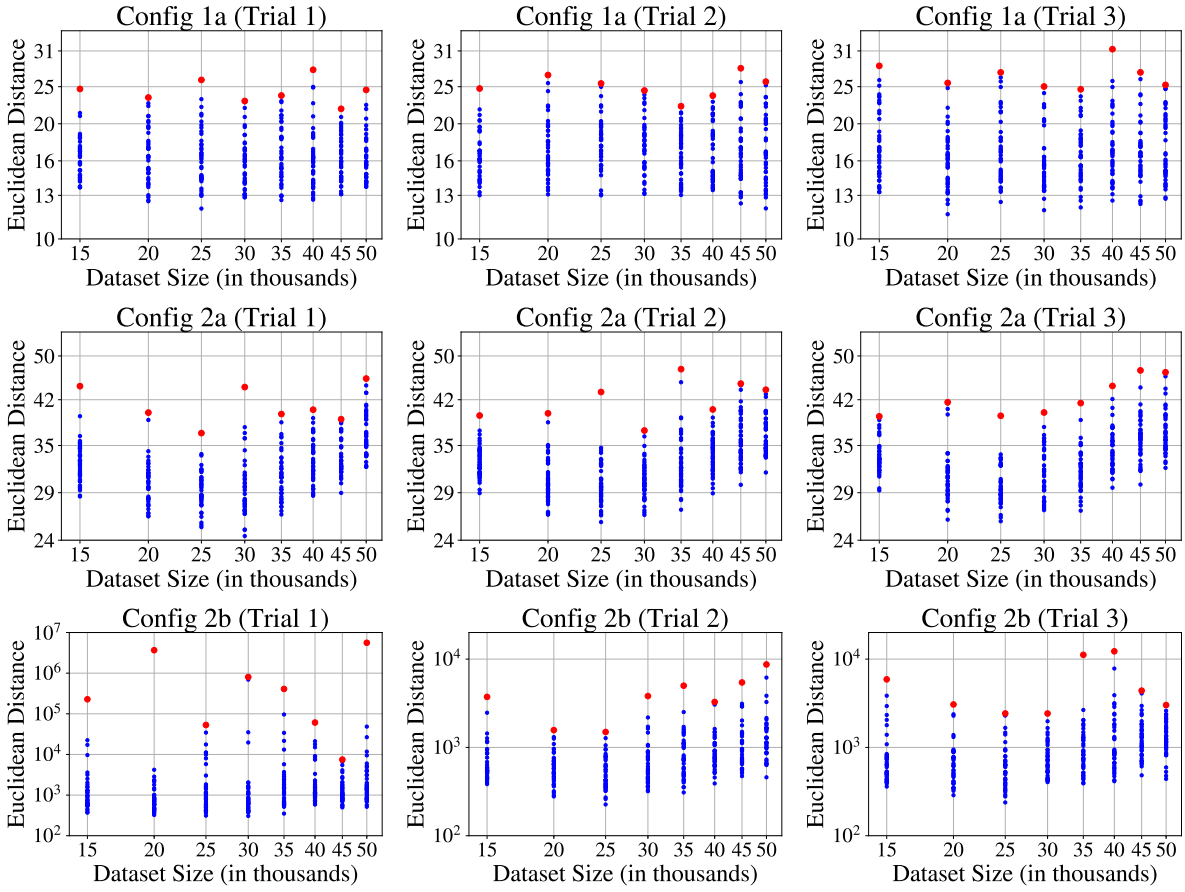


Figure A.11: All trials for $\|\mathcal{A}(S) - \mathcal{A}(S')\|_2$ at $t = 100,000$. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Each row is a different neural network configuration. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, and the maximum value per dataset size m is plotted as a red dot.

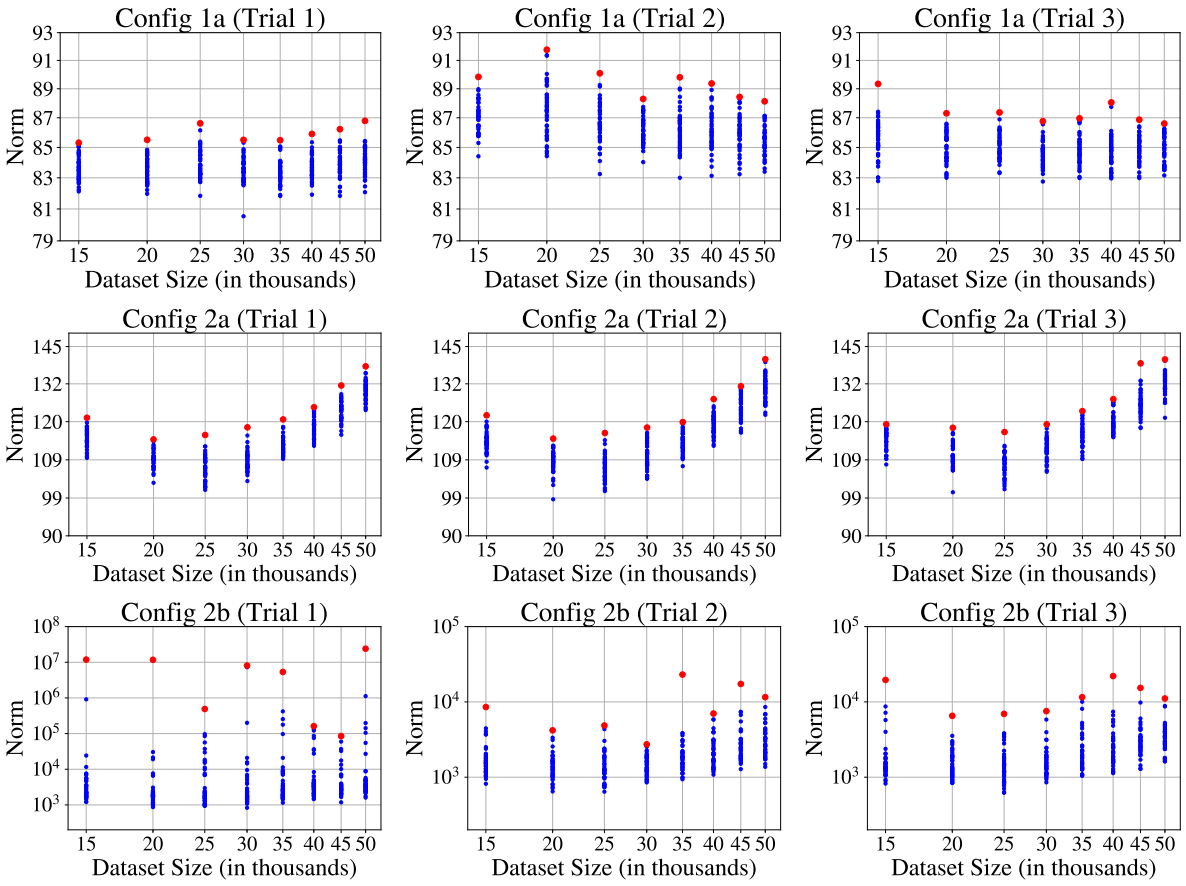


Figure A.12: All trials for $\|\mathcal{A}(S)\|_2$ at $t = 100,000$. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Each row is a different neural network configuration. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, and the maximum value per dataset size m is plotted as a red dot.

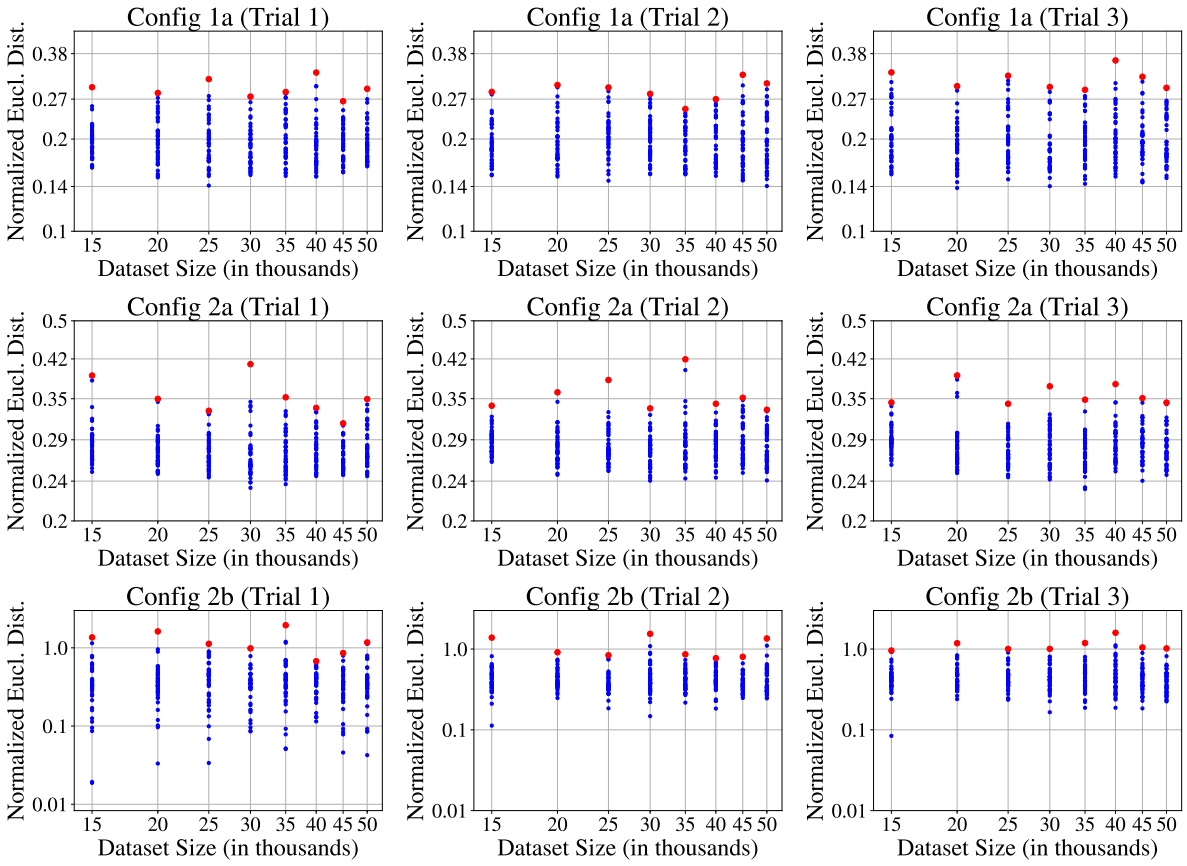


Figure A.13: Normalized Euclidean distance for all trials at $t = 100,000$. For each Euclidean distance $\|\mathcal{A}(S) - \mathcal{A}(S')\|_2$, we divide by $(\|\mathcal{A}(S)\|_2 + \|\mathcal{A}(S')\|_2)/2$. Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). Each row is a different neural network configuration. There are 40 samples per dataset size m. Each sample involves independently drawing $S \sim \mathcal{D}_{\text{train}}^m$, $z \sim \mathcal{D}_{\text{test}}$, $i \sim U([m])$, and $S' := S^{i \leftarrow z}$. Each sample is plotted as a blue dot, and the maximum value per dataset size m is plotted as a red dot. The results suggest that normalizing largely mitigates the growth in Euclidean distance with dataset size; however, this does not appear to yield a significant *decrease* in Euclidean distance with dataset size.

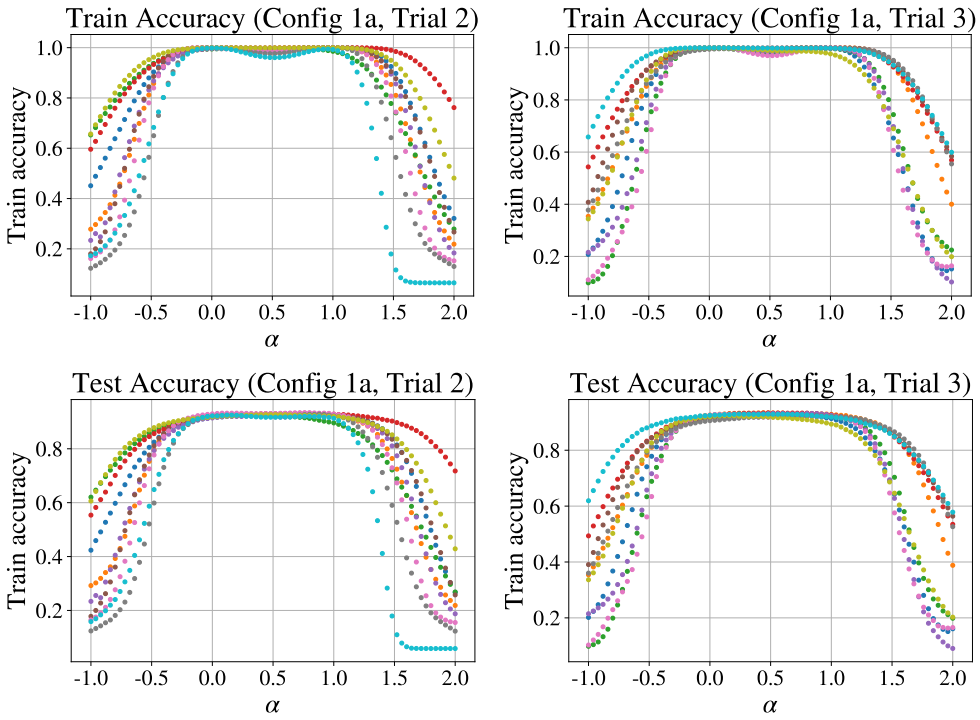


Figure A.14: Additional linear mode connectivity interpolation trials for Configuration 1a (SVHN). Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). In each plot, the x-axis is α, ranging from -1.0 to 2.0, and the y-axis is the train or test accuracy evaluated at the parameters $\alpha W_S + (1-\alpha)W_{S'}$ for some (S, S') pair. Specifically, each color represents a different (S, S') pair from among the 40 samples described in Section 2.2; each plot includes 10 such pairs randomly selected from among the 40. Overall, we see roughly nondecreasing accuracy when linearly interpolating between W_S and $W_{S'}$.

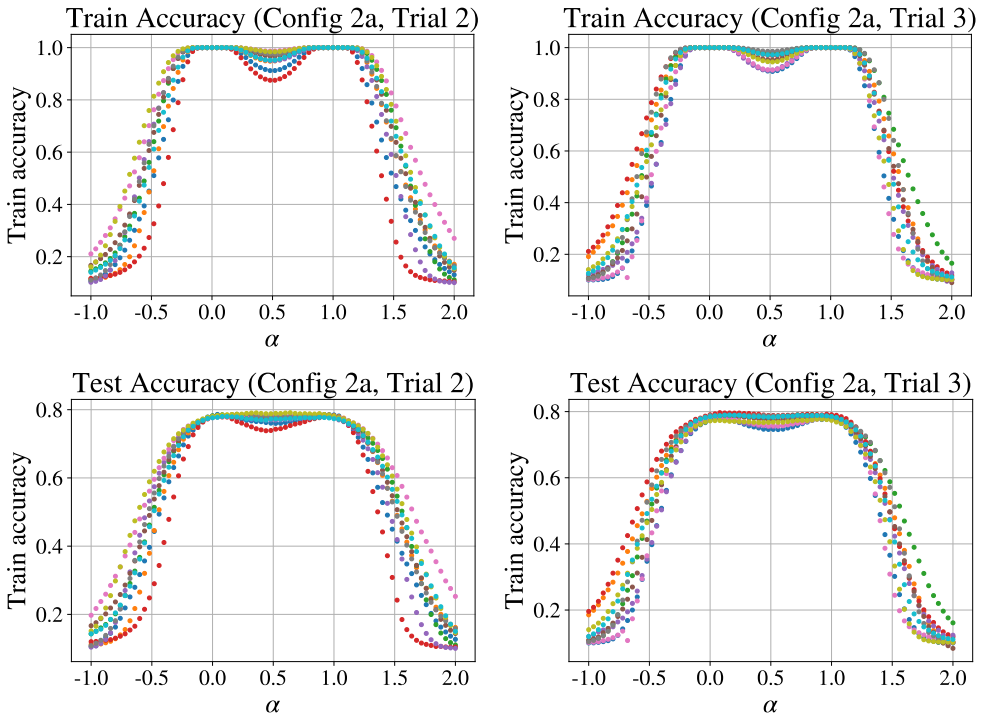


Figure A.15: Additional linear mode connectivity interpolation trials for Configuration 2a (CIFAR-10, no momentum). (Note: We omit Configuration 2b from additional trials because the lack of connectivity seen in the body of the paper is not our focus in these additional trials; rather, we are simply interested in confirming cases of linear mode connectivity.) Each column is a different trial (i.e., different random seed controlling initialization and SGD data order). In each plot, the x-axis is α, ranging from -1.0 to 2.0, and the y-axis is the train or test accuracy evaluated at the parameters $\alpha W_S + (1-\alpha)W_{S'}$ for some (S, S') pair. Specifically, each color represents a different (S, S') pair from among the 40 samples described in Section 2.2; each plot includes 10 such pairs randomly selected from among the 40. Overall, we see slightly decreasing accuracy when linearly interpolating between W_S and $W_{S'}$.

Appendix B

Local Signal Adaptivity: Additional Details

B.1 Full Proofs for Section 3.4

B.1.1 Simplifying calculations and notation

We introduce the following calculations and notation to simplify the subsequent proofs.

Let $I = [k+1] \setminus \{i^\star\}$.

For any given t, we decompose $\mathbf{w}^{(t)} = (\mathbf{w}^{(t)} \cdot \mathbf{w}_\star)\mathbf{w}_\star + \mathbf{w}_\perp^{(t)}$.

We let $c_t := \mathbf{w}^{(t)} \cdot \mathbf{w}_\star$ and $v_t := \|\mathbf{w}_\perp^{(t)}\|_2$. For $i \in I$, we define $A_i := X_i \cdot \mathbf{w}_\star$ and $B_i := X_i \cdot \mathbf{w}_\perp^{(t)}/\|\mathbf{w}_\perp^{(t)}\|_2$. Since X_i is a spherically symmetric Gaussian vector and $\mathbf{w}_\star$ and $\mathbf{w}_\perp^{(t)}/\|\mathbf{w}_\perp^{(t)}\|_2$ define two fixed (per t) orthogonal directions, we have

$$A_i, B_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2).$$

Finally, we define $Z_i := c_t A_i + v_t B_i \sim \mathcal{N}(0, (c_t^2 + v_t^2)\sigma^2)$ and $s_t^2 := (c_t^2 + v_t^2)\sigma^2$. With this notation, we obtain the following simplification of $f_t(X)$, used extensively throughout the proofs:

$$\begin{aligned} f_t(X) &= \sum_{i=1}^{k+1} \Big[\mathrm{ReLU}(\langle \mathbf{w}_t, X_i\rangle + b_t) - \mathrm{ReLU}(-\langle \mathbf{w}_t, X_i\rangle + b_t)\Big] \\ &= \sum_{i\in I} \Big[\mathrm{ReLU}(Z_i + b_t) - \mathrm{ReLU}(-Z_i + b_t)\Big] + \mathrm{ReLU}(c_t Y + b_t) - \mathrm{ReLU}(-c_t Y + b_t). \end{aligned}$$

The population gradients then simplify as follows:

$$\nabla_{\mathbf{w}} \mathbb{E}[\ell(f_t(X), y)] = \mathbb{E}\left[-Y\sigma(-Y f_t(X)) \sum_{i\in I} A_i \Big[\mathbb{I}\{Z_i \geq -b_t\} + \mathbb{I}\{Z_i \leq b_t\}\Big]\right] \mathbf{w}_\star$$

$$+\mathbb{E}\left[-\sigma(-Yf_t(X))\Big[\mathbb{I}\{c_tY \geq -b_t\} + \mathbb{I}\{c_tY \leq b_t\}\Big]\right]\mathbf{w}_\star$$
$$+\mathbb{E}\left[-Y\sigma(-Yf_t(X))\sum_{i\in I} B_i\Big[\mathbb{I}\{Z_i \geq -b_t\} + \mathbb{I}\{Z_i \leq b_t\}\Big]\right]\mathbf{w}_\perp/\|\mathbf{w}_\perp\|_2.$$

$$\nabla_b\mathbb{E}[\ell(f_t(X),y)] = \mathbb{E}\left[-Y\sigma(-Yf_t(X))\sum_{i\in I}\big[\mathbb{I}\{Z_i \geq -b_t\} - \mathbb{I}\{Z_i \leq b_t\}\big]\right]$$
$$+\mathbb{E}\left[-Y\sigma(-Yf_t(X))\left[\mathbb{I}\{c_tY \geq -b_t\} - \mathbb{I}\{c_tY \leq b_t\}\right]\right].$$

We define the following variants of the sum over I:

$$S := \sum_{i\in I}\Big[\mathrm{ReLU}(Z_i + b_t) - \mathrm{ReLU}(-Z_i + b_t)\Big]$$
$$S_i := \mathrm{ReLU}(Z_i + b_t) - \mathrm{ReLU}(-Z_i + b_t)$$
$$S_{-i} := \sum_{j\in I\setminus\{i\}}\Big[\mathrm{ReLU}(Z_j + b_t) - \mathrm{ReLU}(-Z_j + b_t)\Big]$$

Thus, for any $i \in I$, $S = S_{-i} + S_i$.

We further define:

$$f_{-i}(X) := f_t(X) - S_i = S_{-i} + \mathrm{ReLU}(c_tY + b_t) - \mathrm{ReLU}(-c_tY + b_t).$$

We define $\alpha_t := |b_t|/s_t$.

B.1.2 Frequently-used Gaussian facts

Following common notation, we let $\phi(x)$ denote the standard normal PDF, $\Phi(x)$ denote the standard normal CDF, and $\Phi^c(x)$ denote the standard normal complementary CDF.

Using the notation defined in Section B.1.1, we have:

$$\mathbb{E}[Z_i \mid Z_i \geq -b_t] = s_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)} = \sqrt{c_t^2 + v_t^2}\,\sigma\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}. \tag{B.1}$$

$$\mathrm{Var}[Z_i \mid Z_i \geq -b_t] = s_t^2\left[1 + \alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)} - \left(\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)^2\right]. \tag{B.2}$$

$$\mathbb{E}[Z_i^2 \mid Z_i \geq -b_t] = s_t^2\left[1 + \alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right] = (c_t^2 + v_t^2)\sigma^2\left[1 + \alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right]. \tag{B.3}$$

$$\mathbb{E}[c_tA_i \mid Z_i] = Z_i\frac{c_t^2\sigma^2}{c_t^2\sigma^2 + v_t^2\sigma^2} = Z_i\frac{c_t^2}{c_t^2 + v_t^2}. \tag{B.4}$$

$$\mathbb{E}[v_t B_i \mid Z_i] = Z_i \frac{v_t^2 \sigma^2}{v_t^2 \sigma^2 + c_t^2 \sigma^2} = Z_i \frac{v_t^2}{c_t^2 + v_t^2}. \tag{B.5}$$

We use the following (relatively tight) bounds on Mills ratio $\frac{\Phi^c(x)}{\phi(x)}$, for $x \geq 0$:

$$\frac{2}{\sqrt{x^2+4}+x} \leq \frac{\Phi^c(x)}{\phi(x)} \leq \frac{2}{\sqrt{x^2+2}+x}. \tag{B.6}$$

Lemma B.1

Let $X_1, \ldots, X_n$ be i.i.d. $\mathcal{N}(0, \sigma^2)$ RVs. Then $\mathbb{P}\left[\max_{j \leq n} X_j \geq 2\sigma\sqrt{\log n}\right] \leq 1/n$.

Proof. For all $t > 0$, $\mathbb{P}[\max_j X_j \geq t] \leq \sum_j \mathbb{P}[X_j \geq t] \leq n \cdot \exp\left(-\frac{t^2}{2\sigma^2}\right)$. Now let $t = 2\sigma\sqrt{\log n}$.

Then $\mathbb{P}[\max_j X_j \geq t] \leq n \exp\left(-\frac{4\sigma^2 \log n}{2\sigma^2}\right) = n \exp(-2 \log n) = 1/n$. □

Lemma B.2

Let $X_1, \ldots, X_n$ be i.i.d. $\mathcal{N}(0, \sigma^2)$ RVs. Then $\mathbb{P}\left[\max_{j \leq n} |X_j| \geq 2\sigma\sqrt{\log n}\right] \leq 2/n$.

Proof. For all $t > 0$, $\mathbb{P}[\max_j |X_j| \geq t] \leq \sum_j \mathbb{P}[|X_j| \geq t] \leq n \cdot 2\exp\left(-\frac{t^2}{2\sigma^2}\right)$. Now let $t = 2\sigma\sqrt{\log n}$.

Then $\mathbb{P}[\max_j |X_j| \geq t] \leq 2n \exp\left(-\frac{4\sigma^2 \log n}{2\sigma^2}\right) = 2n \exp(-2 \log n) = 2/n$. □

B.1.3 Neural network upper bound proofs

Training invariants

Lemma B.3: Initialization

Initialize $\mathbf{w}^{(0)} \sim \mathcal{N}\left(0, \frac{\delta^2}{kd} I_{d\times d}\right)$. Then with probability at least $1 - e^{-\Omega(k)}$, we have $|c^{(0)}|, v^{(0)} \leq \delta$.

Proof. Let E denote the event that, $\forall i \in [d]$, $w_i^{(0)} \leq \frac{\delta}{\sqrt{d}}$. Then via a Gaussian tail bound and union bound, we have

$$\mathbb{P}[\neg E] \leq \sum_{i\in[d]} \mathbb{P}\left[|w_i^{(0)}| \geq \frac{\delta}{\sqrt{d}}\right] \leq d \cdot 2e^{-k/2}.$$

E implies $\|w^{(0)}\|_2 \leq \delta$, so $|c^{(0)}|, v^{(0)} \leq \delta$. □

Remark B.1. *We choose $\delta = 1/k^{100}$. Thus, $\mathbf{w}^{(0)} \sim \mathcal{N}(0, \sigma_0^2 I_{d\times d})$, where σ_0 is $1/\text{poly}(k)$. We choose η so that $\eta/\delta = \Omega(k)$.*

Lemma B.4: Stochastic gradients

Using $n = \text{poly}(k)$ samples per mini-batch, at any given time step $t \leq T$, with probability at least $1 - e^{-\widetilde{\Omega}(k)}$, we have:

$$\left\| \frac{1}{n} \sum_{(X,y)\in\mathcal{Z}} \nabla_{\mathbf{w}} \ell(f_t(X), y) - \mathbb{E}\left[\nabla_{\mathbf{w}} \ell(f_t(X), y)\right] \right\|_2 \leq \frac{\delta}{\eta \text{poly}(k)}.$$

$$\left| \frac{1}{n} \sum_{(X,y)\in\mathcal{Z}} \nabla_b \ell(f_t(X), y) - \mathbb{E}\left[\nabla_b \ell(f_t(X), y)\right] \right| \leq \frac{1}{k}.$$

Proof. Consider a mini-batch of n examples $\mathcal{Z} \sim \mathcal{D}^n$, where $n = \text{poly}(k)$.

For some (X, y), $j \in [d]$, let E_j be the event that $|X_{i,j}| \leq 1$ for all $i \in I$ and $g_j := (\nabla_{\mathbf{w}} \ell(f(X), y))_j$.

$$\begin{aligned}
\mathbb{E}[g_j] &= \mathbb{E}[g_j \mid E_j]\mathbb{P}[E_j] + \mathbb{E}[g_j \mid \neg E_j]\mathbb{P}[\neg E_j] \\
&= \mathbb{E}[g_j \mid E_j]\left(1 - e^{-\widetilde{\Omega}(k)}\right) + \mathbb{E}[g_j \mid \neg E_j] e^{-\widetilde{\Omega}(k)} \\
\mathbb{E}[g_j] - \mathbb{E}[g_j \mid E_j] &= e^{-\widetilde{\Omega}(k)} \left(\mathbb{E}[g_j \mid \neg E_j] - \mathbb{E}[g_j \mid E_j]\right) \\
|\mathbb{E}[g_j] - \mathbb{E}[g_j \mid E_j]| &\leq e^{-\widetilde{\Omega}(k)} \mathcal{O}(k) \\
|\mathbb{E}[g_j] - \mathbb{E}[g_j \mid E_j]| &\leq e^{-\widetilde{\Omega}(k)}.
\end{aligned}$$

Let $\widehat{E}_j$ be the event that $|X_{i,j}| \leq 1$ for all $i \in I$, for all $(X, y) \in \mathcal{Z}$. For any $j \in [d], t > 0$, we have:

$$\begin{aligned}
&\mathbb{P}\left[\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}(\nabla_{\mathbf{w}}\ell(f(X),y))_j - \mathbb{E}[(\nabla_{\mathbf{w}}\ell(f(X),y))_j]\right| \geq t\right] \\
&= \mathbb{P}\left[\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}(\nabla_{\mathbf{w}}\ell(f(X),y))_j - \mathbb{E}[(\nabla_{\mathbf{w}}\ell(f(X),y))_j]\right| \geq t \;\middle|\; \widehat{E}_j\right]\mathbb{P}[\widehat{E}_j] \\
&+ \mathbb{P}\left[\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}(\nabla_{\mathbf{w}}\ell(f(X),y))_j - \mathbb{E}[(\nabla_{\mathbf{w}}\ell(f(X),y))_j]\right| \geq t \;\middle|\; \neg\widehat{E}_j\right]\mathbb{P}[\neg\widehat{E}_j] \\
&\leq \mathbb{P}\left[\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}(\nabla_{\mathbf{w}}\ell(f(X),y))_j - \mathbb{E}[(\nabla_{\mathbf{w}}\ell(f(X),y))_j]\right| \geq t \;\middle|\; \widehat{E}_j\right] + e^{-\widetilde{\Omega}(k)} \\
&\leq \mathbb{P}\left[\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}(\nabla_{\mathbf{w}}\ell(f(X),y))_j - \mathbb{E}[g_j \mid E_j]\right| + \left|\mathbb{E}[g_j \mid E_j] - \mathbb{E}[g_j]\right| \geq t \;\middle|\; \widehat{E}_j\right] + e^{-\widetilde{\Omega}(k)} \\
&\leq \mathbb{P}\left[\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}(\nabla_{\mathbf{w}}\ell(f(X),y))_j - \mathbb{E}[g_j \mid E_j]\right| \geq t - \left|\mathbb{E}[g_j \mid E_j] - \mathbb{E}[g_j]\right| \;\middle|\; \widehat{E}_j\right] + e^{-\widetilde{\Omega}(k)}
\end{aligned}$$

We can now apply a Hoeffding bound with $t = \frac{\delta}{\eta\sqrt{d}\text{poly}(k)}$. For sufficiently large $n = \text{poly}(k)$, we obtain:

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}(\nabla_{\mathbf{w}}\ell(f(X),y))_j - \mathbb{E}[(\nabla_{\mathbf{w}}\ell(f(X),y))_j]\right| \geq t\right] \leq e^{-\widetilde{\Omega}(k)}.$$

Finally, with probability at least $1 - e^{-\widetilde{\Omega}(k)}$, we have:

$$\left\|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}\nabla_{\mathbf{w}}\ell(f(X),y) - \mathbb{E}\left[\nabla_{\mathbf{w}}\ell(f(X),y)\right]\right\|_2 \leq \frac{\delta}{\eta\text{poly}(k)}.$$

For all $(X, y) \in \mathbb{R}^{(k+1)\times d} \times \{-1, 1\}$, we have $|\nabla_b\ell(f(X), y)| \leq \text{poly}(k)$ (deterministically).

Then, by a Hoeffding bound for sufficiently large $n = \text{poly}(k)$, with probability at least $1 - e^{-\widetilde{\Omega}(k)}$, we have:

$$\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}\nabla_b\ell(f(X),y) - \mathbb{E}\left[\nabla_b\ell(f(X),y)\right]\right| \leq \frac{1}{k}.$$

□

Lemma B.5

There exists an absolute constant k_0 such that, for every $k \geq k_0$, we have $\alpha_t \leq 2\sqrt{\log k}$ for all t such that $\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(f_t(X)) \neq y] \geq 0.01$.

Proof. Recall that we have defined $\alpha_t := |b_t|/s_t$ in Section B.1.1.

At the start of training, when the derivatives of b and c are each coming from the signal patch and thus both b and c are increasing, the ratio b/c is determined by $\eta_b/\eta_{\mathbf{w}} = 1/k$.

Once b starts to decrease, we consider α_t for $b_t < 0$ and $c_t > 0$. We prove this case by contradiction.

Suppose $\alpha_t > 2\sqrt{\log k}$. Then we have:

$$\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(f_t(X)) \neq y] \leq \mathbb{P}[\max_i |Z_i| \geq |b_t|] \leq \sum_{i\in I} \mathbb{P}[|Z_i| \geq |b_t|] \leq k \cdot 2e^{-\alpha_t^2/2} < 2/k.$$

For sufficiently large k, we obtain $\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(f_t(X)) \neq y] < 0.01$, a contradiction. Thus, we must have $\alpha_t \leq 2\sqrt{\log k}$. □

Lemma B.6

There exists a constant $\beta > 0$ and a constant k_0 such that, for every $k \geq k_0$, the following holds for all $t \leq T$ such that $\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(f_t(X)) \neq y] \geq 0.01$:

$$\text{Var}[S_i \mid |S_i| \leq s_t \log^2 k] \geq \beta(c_t + b_t)^2/k \;\; \forall i \in I.$$

Proof. For any $M > 0$, let E_M be the event that $|S_i| \leq M \;\; \forall i \in I$. For any $t > 0$, we then have:

$$\begin{aligned}
\mathbb{P}\left[\sum_{i\in I} S_i \geq t\right] &= \mathbb{P}\left[\sum_{i\in I} S_i \geq t \;\middle|\; E_M\right]\mathbb{P}[E_M] + \mathbb{P}\left[\sum_{i\in I} S_i \geq t \;\middle|\; \neg E_M\right]\mathbb{P}[\neg E_M] \\
&\leq \mathbb{P}\left[\sum_{i\in I} S_i \geq t \;\middle|\; E_M\right] + \mathbb{P}[\neg E_M] \\
&\leq \mathbb{P}\left[\sum_{i\in I} S_i \geq t \;\middle|\; E_M\right] + k\mathbb{P}[|S_1| > M] \\
&\leq \exp\left(-\frac{t^2}{2k\,\text{Var}[S_1 \mid |S_1| \leq M] + 2Mt/3}\right) + k\mathbb{P}[|Z_1| > M - b] \\
&\leq \exp\left(-\frac{t^2}{2k\,\text{Var}[S_1 \mid |S_1| \leq M] + 2Mt/3}\right) + 2k\exp\left(-\frac{(M-b)^2}{2s_t^2}\right),
\end{aligned}$$

by applying a Bernstein inequality.

Choosing $M = s_t \log^2 k$ and $t = c_t + b_t$ and defining $u_t^2 := \mathrm{Var}[S_1 \mid |S_1| \le M]$, we obtain:

$$\mathbb{P}\left[\sum_{i\in I} S_i \ge c_t + b_t\right] \le \exp\left(-\frac{(c_t+b_t)^2}{2ku_t^2 + 2s_t\log^2 k(c_t+b_t)/3}\right) + 2k\exp\left(-\frac{(s_t\log^2 k - b_t)^2}{2s_t^2}\right).$$

$2k\exp\left(-\frac{(s_t\log^2 k - b_t)^2}{2s_t^2}\right) = e^{-\Omega(\log^4 k)}$, and $2s_t\log^2 k(c_t+b_t)/3 = \mathcal{O}(c_t^2\log^3 k/\sqrt{k})$. Therefore, unless $u_t^2 \ge \beta(c_t+b_t)^2/k$ for some constant $\beta > 0$, we will have $\mathbb{P}\left[\sum_{i\in I} S_i \ge c_t + b_t\right] < 0.01$ for sufficiently large k.

□

Corollary B.1. *There exists a constant $\beta > 0$ and a constant k_0 such that, for every $k \ge k_0$, the following holds for all $t \le T$ such that $\mathbb{P}_{(X,y)\sim\mathcal{D}}[\mathrm{sign}(f_t(X)) \ne y] \ge 0.01$ and all $i \in I$:*

$$\mathrm{Var}[S_i] \ge \beta\frac{(c_t+b_t)^2}{k}.$$

Function value stays $\mathcal{O}(1)$ with $\Omega(1)$ probability

Lemma B.7

There exists a constant $C > 0$ and constant k_0 such that, for all $k \geq k_0$, the following holds for all $t \leq T$:

$$c_t \leq C.$$

Proof. Since we are doing stochastic gradient descent with stochastic gradients very close to the population gradients, we can easily conclude that the loss cannot grow throughout training. $\mathbb{E}[\ell(f_0(X), y)] \leq 1$ at the start of training, so we must have $\mathbb{E}[\ell(f_t(X), y)] \leq 1$ for all $t \leq T$.

Let E denote the event that $|S_i| \leq s_t \log^2 k \ \ \forall i \in I$.

$$\begin{aligned}
1 &\geq \mathbb{E}[\ell(f_t(X), y)] \\
&= \mathbb{E}[\log(1 + \exp(-Y f_t(X)))] \\
&= \frac{1}{2}\mathbb{E}[\log(1 + \exp(-f_t(X))) \mid Y = 1] + \frac{1}{2}\mathbb{E}[\log(1 + \exp(f_t(X))) \mid Y = -1] \\
&\geq \frac{1}{2}\mathbb{E}[\text{ReLU}(-f_t(X)) \mid Y = 1] + \frac{1}{2}\mathbb{E}[\text{ReLU}(f_t(X)) \mid Y = -1] \\
&= \mathbb{E}[\text{ReLU}(-f_t(X)) \mid Y = 1] \\
&= \mathbb{E}\left[\text{ReLU}\left(-(c_t + b_t) - \sum_{i \in I} S_i\right)\right] \\
&\geq \mathbb{E}\left[\text{ReLU}\left(-(c_t + b_t) - \sum_{i \in I} S_i\right) \,\middle|\, \sum_{i \in I} S_i \leq -2(c_t + b_t)\right] \mathbb{P}\left[\sum_{i \in I} S_i \leq -2(c_t + b_t)\right] \\
&\geq (c_t + b_t)\mathbb{P}\left[\sum_{i \in I} S_i \leq -2(c_t + b_t)\right] \\
&= (c_t + b_t)\left(\mathbb{P}\left[\sum_{i \in I} S_i \leq -2(c_t + b_t) \mid E\right]\mathbb{P}[E] + \mathbb{P}\left[\sum_{i \in I} S_i \leq -2(c_t + b_t) \mid \neg E\right]\mathbb{P}[\neg E]\right) \\
&\geq (c_t + b_t)\mathbb{P}\left[\sum_{i \in I} S_i \leq -2(c_t + b_t) \mid E\right]\mathbb{P}[E].
\end{aligned}$$

Following the notation in the proof of Lemma B.6, we let $u_t^2 := \text{Var}[S_i \mid |S_i| \leq s_t \log^2 k]$, for any $i \in I$. Since $\mathbb{E}[S_i \mid |S_i| \leq s_t \log^2 k] = 0$, we also have $\mathbb{E}[S_i^2 \mid |S_i| \leq s_t \log^2 k] = u_t^2$.

We then use Lemma B.6 and Berry-Esseen to lower bound $\mathbb{P}\left[\sum_{i \in I} S_i \leq -2(c_t + b_t) \mid E\right]$ as follows, where C_1 is some positive constant.

$$\begin{aligned}
\sup_{x \in \mathbb{R}} \left|\mathbb{P}\left(\frac{\sum_{i \in I} S_i}{u_t\sqrt{k}} \leq x \,\middle|\, E\right) - \Phi(x)\right| &\leq \frac{C_1}{\sqrt{k}} \cdot \frac{\mathbb{E}[|S_i|^3 \mid |S_i| \leq s_t \log^2 k]}{u^3} \\
&\leq \frac{C_1}{\sqrt{k}} \cdot \frac{s_t \log^2 k \cdot \mathbb{E}[|S_i|^2 \mid |S_i| \leq s_t \log^2 k]}{u_t^3}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{C_1}{\sqrt{k}} \cdot \frac{s_t \log^2 k}{u_t} \\
&\leq \frac{C_1 s_t \log^2 k}{\sqrt{\beta}(c_t + b_t)} \\
&\leq \frac{C_1}{2\sqrt{\beta}} \frac{\log^3 k}{\sqrt{k}},
\end{aligned}
$$

by Lemma B.6, which says that $u_t \geq \sqrt{\beta}\frac{c_t+b_t}{\sqrt{k}}$ for some constant $\beta > 0$. Based on Lemma B.6, we choose x such that $x < 0$ and $|x| \leq 2/\sqrt{\beta}$. We obtain

$$
\mathbb{P}\left[\sum_{i \in I} S_i \leq -2(c_t + b_t) \mid E\right] \geq \Phi(x) - \frac{C_1}{2\sqrt{\beta}} \frac{\log^3 k}{\sqrt{k}} \geq \frac{1}{2}\Phi(x)
$$

for sufficiently large k.

By Lemma B.6's proof, we know that $\mathbb{P}[E] = 1 - e^{-\Omega(\log^4 k)}$.

Putting these pieces together, we obtain $1 \geq \mathbb{E}[\ell(f_t(X), y)] = \frac{\Phi(x)}{4}(c_t + b_t)$, and by Lemma B.5, we conclude that there must exist a constant $C > 0$ such that $c_t \leq C$.

□

Lemma B.8

For all $t \leq T$, $\mathbb{P}[|S| < c_t + b_t + 10] \geq 4/5$.

Proof. We closely follow the proof of Lemma B.7. Since we are doing stochastic gradient descent with stochastic gradients very close to the population gradients, we can easily conclude that the loss cannot grow throughout training. $\mathbb{E}[\ell(f_0(X), y)] \leq 1$ at the start of training, so we must have $\mathbb{E}[\ell(f_t(X), y)] \leq 1$ for all $t \leq T$.

$$
\begin{aligned}
1 &\geq \mathbb{E}[\ell(f_t(X), y)] \\
&= \mathbb{E}[\log(1 + \exp(-Y f_t(X)))] \\
&= \frac{1}{2}\mathbb{E}[\log(1 + \exp(-f_t(X))) \mid Y = 1] + \frac{1}{2}\mathbb{E}[\log(1 + \exp(f_t(X))) \mid Y = -1] \\
&\geq \frac{1}{2}\mathbb{E}[\mathrm{ReLU}(-f_t(X)) \mid Y = 1] + \frac{1}{2}\mathbb{E}[\mathrm{ReLU}(f_t(X)) \mid Y = -1] \\
&= \mathbb{E}[\mathrm{ReLU}(-f_t(X)) \mid Y = 1] \\
&= \mathbb{E}\left[\mathrm{ReLU}\left(-(c_t + b_t) - \sum_{i \in I} S_i\right)\right] \\
&\geq \mathbb{E}\left[\mathrm{ReLU}\left(-(c_t + b_t) - \sum_{i \in I} S_i\right) \,\middle|\, \sum_{i \in I} S_i \leq -(c_t + b_t) - 10\right] \mathbb{P}\left[\sum_{i \in I} S_i \leq -(c_t + b_t) - 10\right] \\
&\geq 10 \cdot \mathbb{P}\left[\sum_{i \in I} S_i \leq -(c_t + b_t) - 10\right]
\end{aligned}
$$

$$= 5 \cdot \mathbb{P}\left[\left|\sum_{i \in I} S_i\right| \geq (c_t + b_t) + 10\right]$$
$$= 5 \cdot \left(1 - \mathbb{P}\left[\left|\sum_{i \in I} S_i\right| < (c_t + b_t) + 10\right]\right),$$

finally yielding

$$\mathbb{P}\left[\left|\sum_{i \in I} S_i\right| < (c_t + b_t) + 10\right] \geq 4/5.$$

□

Initial growth of c

Lemma: Lemma 3.1, Full

For any constant $H > 0$, there exist constants $A > 0$ and k_0 such that, for all $k \geq k_0$, for any $t \leq T$ such that $|c_t| \leq H/\log^4 k$, we have the following with probability at least $1 - e^{-\widetilde{\Omega}(k)}$ over a mini-batch $\mathcal{Z} \sim \mathcal{D}^n$ of $n = \text{poly}(k)$ samples:

$$\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}} \nabla_{\mathbf{w}}\ell(f_t(X), y)\cdot \mathbf{w}_\star \leq -A.$$

Proof. We recall the following:

$$\nabla_{\mathbf{w}}\mathbb{E}[\ell(f_t(X),y)]\cdot\mathbf{w}^\star = \underbrace{\mathbb{E}\left[-Y\sigma(-Yf_t(X))\sum_{i\in I}A_i\Big[\mathbb{I}\{Z_i+b_t\geq 0\}+\mathbb{I}\{-Z_i+b_t\geq 0\}\Big]\right]}_{(1)} + \underbrace{\mathbb{E}\left[-\sigma(-Yf(X))\Big[\mathbb{I}\{c_tY+b_t\geq 0\}+\mathbb{I}\{-c_tY+b_t\geq 0\}\Big]\right]}_{(2)}.$$

Let $g(Y) := \text{ReLU}\Big(c_tY+b\Big) - \text{ReLU}\Big(-c_tY+b\Big)$. Then $g(1) = -g(-1)$ and $f(X) = S + g(Y)$.

We first simplify (1) as follows:

$$\begin{aligned}
(1) &= \mathbb{E}\left[-Y\sigma(-Yf_t(X))\sum_{i\in I}A_i\Big[\mathbb{I}\{Z_i\geq -b_t\}+\mathbb{I}\{Z_i\leq b_t\}\Big]\right]\\
&= \mathbb{E}\left[-\sigma\Big(-S-g(1)\Big)\sum_{i\in I}A_i\Big[\mathbb{I}\{Z_i\geq -b_t\}+\mathbb{I}\{Z_i\leq b_t\}\Big]\right]\mathbb{P}[Y=+1]\\
&+ \mathbb{E}\left[\sigma\Big(S+g(-1)\Big)\sum_{i\in I}A_i\Big[\mathbb{I}\{Z_i\geq -b_t\}+\mathbb{I}\{Z_i\leq b_t\}\Big]\right]\mathbb{P}[Y=-1]\\
&= \frac{1}{2}\sum_{i\in I}\mathbb{E}\Big[-\sigma\Big(-g(1)-(S_{-i}+S_i)\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\Big]\\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\Big[-\sigma\Big(-g(1)-(S_{-i}+S_i)\Big)A_i\mathbb{I}\{Z_i\leq b_t\}\Big]\\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\Big[\sigma\Big(-g(1)+(S_{-i}+S_i)\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\Big]\\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\Big[\sigma\Big(-g(1)+(S_{-i}+S_i)\Big)A_i\mathbb{I}\{Z_i\leq b_t\}\Big] \qquad (g(-1)=-g(1))
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[-\sigma\Big(-g(1)+S_{-i}-S_i\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\right] \\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[-\sigma\Big(-g(1)+S_{-i}-S_i\Big)A_i\mathbb{I}\{Z_i\leq b_t\}\right] \\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\right] \\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)A_i\mathbb{I}\{Z_i\leq b_t\}\right] && (S_{-i}\stackrel{d}{=}-S_{-i}) \\
&= \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[-\sigma\Big(-g(1)+S_{-i}-S_i\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\right] \\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\right] \\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\right] \\
&+ \frac{1}{2}\sum_{i\in I}\mathbb{E}\left[-\sigma\Big(-g(1)+S_{-i}-S_i\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\right] && (Z_i\stackrel{d}{=}-Z_i).
\end{aligned}
$$

We then collapse the four summands above. Letting $a_2 := -g(1)+S_{-i}+S_i$, $a_1 := -g(1)+S_{-i}-S_i$, and $\Delta := a_2-a_1 = 2S_i$, we have:

$$
\begin{aligned}
(1) &= \sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)A_i\mathbb{I}\{Z_i\geq -b_t\}\right] \\
&= \sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)A_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t] \\
&= \sum_{i\in I}\mathbb{E}\left[\mathbb{E}\Big[\Big(\sigma(a_2)-\sigma(a_1)\Big)A_i \;\Big|\; Z_j\;\forall j\in I, Z_i\geq -b_t\Big] \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t] \\
&= \sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)\frac{1}{c_t}Z_i\frac{c_t^2}{c_t^2+v_t^2} \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t] \\
&= \frac{c_t}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)Z_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t].
\end{aligned}
$$

We first note that $(\sigma(a_2)-\sigma(a_1))Z_i \geq 0$ for all instantiations of the random variables $Z_i, i\in I$. This is because σ is monotonically increasing and $\operatorname{sign}(S_i)=\operatorname{sign}(Z_i)$.

Then, using the fact that σ is $1/4$-Lipschitz and $|S_i|\leq \max\{2|Z_i|, |Z_i|+|b_t|\}\leq 2|Z_i|+|b_t|$, we upper bound $|(1)|$ as follows:

$$
\begin{aligned}
|(1)| &= \frac{|c_t|}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)Z_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t] \\
&\leq \frac{|c_t|}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\frac{1}{2}S_iZ_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]
\end{aligned}
$$

$$
\begin{aligned}
&\leq \frac{|c_t|}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\frac{1}{2}(2Z_i^2+|b_t||Z_i|)\;\middle|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]\\
&= \frac{|c_t|}{c_t^2+v_t^2}\sum_{i\in I}\left(\mathbb{E}[Z_i^2\mid Z_i\geq -b_t]+\frac{1}{2}|b_t|\mathbb{E}[|Z_i|\mid Z_i\geq -b_t]\right)\mathbb{P}[Z_i\geq -b_t]\\
&\leq \frac{|c_t|}{c_t^2+v_t^2}\sum_{i\in I}\left(\mathbb{E}[Z_i^2\mid Z_i\geq |b_t|]+\frac{1}{2}|b_t|\mathbb{E}[|Z_i|\mid Z_i\geq |b_t|]\right)\mathbb{P}[Z_i\geq -b_t]\\
&= \frac{|c_t|}{c_t^2+v_t^2}\sum_{i\in I}\left((c_t^2+v_t^2)\sigma^2\left[1+\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right]+\frac{1}{2}|b_t|\sqrt{c_t^2+v_t^2}\sigma\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i\geq -b_t]\\
&= |c_t|\sum_{i\in I}\left(\sigma^2\left[1+\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right]+\frac{1}{2}|b_t|\frac{1}{\sqrt{c_t^2+v_t^2}}\frac{\sigma^2}{\sigma}\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i\geq -b_t]\\
&= |c_t|\sum_{i\in I}\left(\sigma^2\left[1+\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right]+\frac{1}{2}\sigma^2\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i\geq -b_t]\\
&= |c_t|\sum_{i\in I}\left(\sigma^2+\frac{3}{2}\sigma^2\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i\geq -b_t].
\end{aligned}
$$

Since $\alpha_t \leq 2\sqrt{\log k}$ by Lemma B.5, we have $|(1)| \leq A_1|c_t|\log^3 k$ for some constant $A_1 > 0$. By Lemmas B.7 and B.8, there exists a constant $A_2 > 0$ such that $(2) \leq -A_2$.

Thus, if $|c_t| \leq H\frac{1}{\log^4 k}$ for any constant $H > 0$, then there exists a constant $A_3 > 0$ such that $(1)+(2) \leq -A_3$ for any sufficiently large k.

Finally, by Lemma B.4, we have with probability at least $1 - e^{-\widetilde{\Omega}(k)}$,

$$
\left\|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}\nabla_{\mathbf{w}}\ell(f_t(X),y) - \mathbb{E}\left[\nabla_{\mathbf{w}}\ell(f_t(X),y)\right]\right\|_2 \leq \frac{\delta}{\eta\text{poly}(k)},
$$

and therefore with probability at least $1 - e^{-\widetilde{\Omega}(k)}$,

$$
\left|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}\nabla_{\mathbf{w}}\ell(f_t(X),y)\cdot\mathbf{w}_\star - \mathbb{E}\left[\nabla_{\mathbf{w}}\ell(f_t(X),y)\right]\cdot\mathbf{w}_\star\right| \leq \frac{\delta}{\eta\text{poly}(k)}.
$$

We conclude that, with probability at least $1 - e^{-\widetilde{\Omega}(k)}$, we have

$$
\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}\nabla_{\mathbf{w}}\ell(f_t(X),y)\cdot\mathbf{w}_\star \leq -A_3 + \frac{\delta}{\eta\text{poly}(k)} \leq -A
$$

for some constant $A > 0$, for any sufficiently large k.

□

Bounded growth of $\|\mathbf{w}_\perp^{(t)}\|_2$

Lemma: Lemma 3.2, Full

For any $t \le T$, we have the following with probability at least $1 - e^{-\widetilde{\Omega}(k)}$ over a mini-batch $\mathcal{Z} \sim \mathcal{D}^n$ of $n = \mathrm{poly}(k)$ samples:

$$\|\mathbf{w}_\perp^{(t)}\|_2 - \|\mathbf{w}_\perp^{(t-1)}\|_2 \le \frac{\delta}{\mathrm{poly}(k)}.$$

Proof.

$$\nabla_{\mathbf{w}} \mathbb{E}[\ell(f_t(X), y)] \cdot \mathbf{w}_\perp^{(t)} / \|\mathbf{w}_\perp^{(t)}\|_2 = \underbrace{\mathbb{E}\left[-Y\sigma(-Y f_t(X)) \sum_{i \in I} B_i \Big[\mathbb{I}\{Z_i \ge -b_t\} + \mathbb{I}\{Z_i \le b_t\}\Big]\right]}_{(1)}.$$

Let $g(Y) := \mathrm{ReLU}\Big(c_t Y + b_t\Big) - \mathrm{ReLU}\Big(-c_t Y + b_t\Big)$. Then $g(1) = -g(-1)$ and $f_t(X) = S + g(Y)$.

We simplify (1) as follows:

$$\begin{aligned}
(1) &= \mathbb{E}\left[-Y\sigma(-Y f_t(X)) \sum_{i \in I} B_i \Big[\mathbb{I}\{Z_i \ge -b_t\} + \mathbb{I}\{Z_i \le b_t\}\Big]\right] \\
&= \mathbb{E}\left[-\sigma\Big(-S - g(1)\Big) \sum_{i \in I} B_i \Big[\mathbb{I}\{Z_i \ge -b_t\} + \mathbb{I}\{Z_i \le b_t\}\Big]\right] \mathbb{P}[Y = +1] \\
&+ \mathbb{E}\left[\sigma\Big(S + g(-1)\Big) \sum_{i \in I} B_i \Big[\mathbb{I}\{Z_i \ge -b_t\} + \mathbb{I}\{Z_i \le b_t\}\Big]\right] \mathbb{P}[Y = -1] \\
&= \frac{1}{2} \sum_{i \in I} \mathbb{E}\Big[-\sigma\Big(-g(1) - (S_{-i} + S_i)\Big) B_i \mathbb{I}\{Z_i \ge -b_t\}\Big] \\
&+ \frac{1}{2} \sum_{i \in I} \mathbb{E}\Big[-\sigma\Big(-g(1) - (S_{-i} + S_i)\Big) B_i \mathbb{I}\{Z_i \le b_t\}\Big] \\
&+ \frac{1}{2} \sum_{i \in I} \mathbb{E}\Big[\sigma\Big(-g(1) + (S_{-i} + S_i)\Big) B_i \mathbb{I}\{Z_i \ge -b_t\}\Big] \\
&+ \frac{1}{2} \sum_{i \in I} \mathbb{E}\Big[\sigma\Big(-g(1) + (S_{-i} + S_i)\Big) B_i \mathbb{I}\{Z_i \le b_t\}\Big] && (g(-1) = -g(1)) \\
&= \frac{1}{2} \sum_{i \in I} \mathbb{E}\Big[-\sigma\Big(-g(1) + S_{-i} - S_i\Big) B_i \mathbb{I}\{Z_i \ge -b_t\}\Big] \\
&+ \frac{1}{2} \sum_{i \in I} \mathbb{E}\Big[-\sigma\Big(-g(1) + S_{-i} - S_i\Big) B_i \mathbb{I}\{Z_i \le b_t\}\Big]
\end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)B_i\mathbb{I}\{Z_i\geq -b_t\}\right]\\
&+\frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)B_i\mathbb{I}\{Z_i\leq b_t\}\right] && (S_{-i}\stackrel{d}{=}-S_{-i})\\
&=\frac{1}{2}\sum_{i\in I}\mathbb{E}\left[-\sigma\Big(-g(1)+S_{-i}-S_i\Big)B_i\mathbb{I}\{Z_i\geq -b_t\}\right]\\
&+\frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)B_i\mathbb{I}\{Z_i\geq -b_t\}\right]\\
&+\frac{1}{2}\sum_{i\in I}\mathbb{E}\left[\sigma\Big(-g(1)+S_{-i}+S_i\Big)B_i\mathbb{I}\{Z_i\geq -b_t\}\right]\\
&+\frac{1}{2}\sum_{i\in I}\mathbb{E}\left[-\sigma\Big(-g(1)+S_{-i}-S_i\Big)B_i\mathbb{I}\{Z_i\geq -b_t\}\right] && (Z_i\stackrel{d}{=}-Z_i).
\end{aligned}
$$

We then collapse the four summands above. Letting $a_2 := -g(1)+S_{-i}+S_i$, $a_1 := -g(1)+S_{-i}-S_i$, and $\Delta := a_2-a_1 = 2S_i$, we have:

$$
\begin{aligned}
(1) &= \sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)B_i\mathbb{I}\{Z_i\geq -b_t\}\right]\\
&= \sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)B_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]\\
&= \sum_{i\in I}\mathbb{E}\left[\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)B_i \;\Big|\; Z_j\;\forall j\in I, Z_i\geq -b_t\right]\;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]\\
&= \sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)\frac{1}{v_t}Z_i\frac{v_t^2}{c_t^2+v_t^2} \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]\\
&= \frac{v_t}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)Z_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t].
\end{aligned}
$$

We first note that $(\sigma(a_2)-\sigma(a_1))Z_i \geq 0$ for all instantiations of the random variables $Z_i, i\in I$. This is because σ is monotonically increasing and $\operatorname{sign}(S_i)=\operatorname{sign}(Z_i)$. Then, using the fact that σ is $1/4$-Lipschitz and $|S_i|\leq \max\{2|Z_i|, |Z_i|+|b_t|\}\leq 2|Z_i|+|b_t|$, we upper bound (1) as follows:

$$
\begin{aligned}
(1) &= \frac{v_t}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\Big(\sigma(a_2)-\sigma(a_1)\Big)Z_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]\\
&\leq \frac{v_t}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\frac{1}{2}S_iZ_i \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]\\
&\leq \frac{v_t}{c_t^2+v_t^2}\sum_{i\in I}\mathbb{E}\left[\frac{1}{2}(2Z_i^2+|b_t||Z_i|) \;\Big|\; Z_i\geq -b_t\right]\mathbb{P}[Z_i\geq -b_t]\\
&= \frac{v_t}{c_t^2+v_t^2}\sum_{i\in I}\left(\mathbb{E}[Z_i^2 \mid Z_i\geq -b_t]+\frac{1}{2}|b_t|\mathbb{E}[|Z_i| \mid Z_i\geq -b_t]\right)\mathbb{P}[Z_i\geq -b_t]
\end{aligned}
$$

$$
\begin{aligned}
&\leq \frac{v_t}{c_t^2+v_t^2}\sum_{i\in I}\left(\mathbb{E}[Z_i^2 \mid Z_i \geq |b_t|] + \frac{1}{2}|b_t|\mathbb{E}[|Z_i| \mid Z_i \geq |b_t|]\right)\mathbb{P}[Z_i \geq -b_t] \\
&= \frac{v_t}{c_t^2+v_t^2}\sum_{i\in I}\left((c_t^2+v_t^2)\sigma^2\left[1+\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right] + \frac{1}{2}|b_t|\sqrt{c_t^2+v_t^2}\sigma\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i \geq -b_t] \\
&= v_t\sum_{i\in I}\left(\sigma^2\left[1+\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right] + \frac{1}{2}|b_t|\frac{1}{\sqrt{c_t^2+v_t^2}}\frac{\sigma^2}{\sigma}\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i \geq -b_t] \\
&= v_t\sum_{i\in I}\left(\sigma^2\left[1+\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right] + \frac{1}{2}\sigma^2\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i \geq -b_t] \\
&= v_t\sum_{i\in I}\left(\sigma^2 + \frac{3}{2}\sigma^2\alpha_t\frac{\phi(\alpha_t)}{\Phi^c(\alpha_t)}\right)\mathbb{P}[Z_i \geq -b_t].
\end{aligned}
$$

Since $\alpha_t \leq 2\sqrt{\log k}$ by Lemma B.5, we have $(1) \leq Av_t \log^3 k$ for some constant $A > 0$. On its own, this would cause v_t to decrease toward 0. Finally, by Lemma B.4, with probability at least $1 - e^{-\tilde{\Omega}(k)}$:

$$
\left\|\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}}\nabla_{\mathbf{w}}\ell(f(X),y) - \mathbb{E}\left[\nabla_{\mathbf{w}}\ell(f(X),y)\right]\right\|_2 \leq \frac{\delta}{\eta\text{poly}(k)}.
$$

Thus, for any t, $\|\mathbf{w}_\perp^{(t+1)}\|_2 - \|\mathbf{w}_\perp^{(t)}\|_2 \leq \frac{\delta}{\text{poly}(k)}$.

□

b decreasing

Lemma B.9

There exists a constant $C_1 > 0$ and a constant k_0 such that, for all $k \geq k_0$, we have the following for all $t \leq T$ for which $b_t < 0$ and for all $i \in I$:

$$\mathbb{E}[|S_i|] \geq C_1 \sqrt{\frac{\mathrm{Var}[S_i]}{k^{3/4}}}.$$

Proof. Consider an arbitrary $t \leq T$.

The probability of misclassification is $\mathbb{P}[S > c_t + b_t]$.

Let B denote the event that $\forall i \in I$, $Z_i \leq 2s_t\sqrt{\log k}$. By Lemma B.1, $\mathbb{P}[\neg B] \leq 1/k$.

Let $M = \sum_{i \in I} \mathbb{I}\{Z_i \geq -b_t\}$.

We first note that, for sufficiently large k, we have $\mathbb{P}[S > c_t + b_t \mid B, M < k^{1/4}] = 0$. This is because, conditioned on both B and $M < k^{1/4}$, we have:

$$\begin{aligned}
S &< k^{1/4} \cdot 2s_t\sqrt{\log k} \\
&= \frac{2k^{1/4}\sqrt{c_t^2 + v_t^2}\log^{3/2} k}{\sqrt{k}} \\
&= \frac{2\sqrt{c_t^2 + v_t^2}\log^{3/2} k}{k^{1/4}} \\
&< \frac{1}{2}c_t \\
&< c_t + b_t,
\end{aligned}$$

by Lemmas 3.2 and B.5, for sufficiently large k.

Let $P = \mathbb{P}[Z_i \geq -b_t]$ and let $P' = \mathbb{P}[Z_i \geq -b_t \mid B]$, for any $i \in I$. We note that $\frac{1}{2}\mathbb{P}[S_i \neq 0] = P \geq P'$, and therefore $\mathbb{P}[S_i \neq 0] \geq 2P'$.

We then upper bound $\mathbb{P}[S > c_t + b_t]$ as follows:

$$\begin{aligned}
0.01 \leq \mathbb{P}[S > c_t + b_t] &= \mathbb{P}[S > c_t + b_t \mid B]\mathbb{P}[B] + \mathbb{P}[S > c_t + b_t \mid \neg B]\mathbb{P}[\neg B] \\
&\leq \mathbb{P}[S > c_t + b_t \mid B] + \mathbb{P}[\neg B] \\
&= \mathbb{P}[S > c_t + b_t \mid B, M < k^{1/4}]\mathbb{P}[M < k^{1/4} \mid B] \\
&+ \mathbb{P}[S > c_t + b_t \mid B, M \geq k^{1/4}]\mathbb{P}[M \geq k^{1/4} \mid B] + \mathbb{P}[\neg B] \\
&= \mathbb{P}[S > c_t + b_t \mid B, M \geq k^{1/4}]\mathbb{P}[M \geq k^{1/4} \mid B] + \mathbb{P}[\neg B] \\
&\leq \mathbb{P}[M \geq k^{1/4} \mid B] + \mathbb{P}[\neg B] \\
&\leq \mathbb{P}[M \geq k^{1/4} \mid B] + 1/k.
\end{aligned}$$

We therefore have $\mathbb{P}[M \geq k^{1/4} \mid B] \geq 0.01 - 1/k \geq 0.005$ for sufficiently large k. So, via Chernoff, we conclude that there exists a constant $C_6 > 0$ such that $P' \geq C_6 \frac{1}{k^{3/4}}$ and thus there exists a constant $C_5 > 0$ such that $\mathbb{P}[S_i \neq 0] \geq C_5 \frac{1}{k^{3/4}}$.

Via Equations B.1 and B.2, we have

$$\frac{\mathbb{E}[|S_i|]}{\sqrt{\mathrm{Var}[S_i]}} \geq C_4' \sqrt{\Phi^c(\alpha_t)} = C_4 \sqrt{\mathbb{P}[S_i \neq 0]} \geq C_4 C_5 \sqrt{\frac{1}{k^{3/4}}},$$

completing the proof.

□

Lemma: Lemma 3.3, Full

There exist constants $C, C_2 > 0$ and a constant k_0 such that, for all $k \geq k_0$, for any $t \leq T$ such that $c_t \geq C_2/\log^4 k$, we have the following with probability at least $1 - e^{-\widetilde{\Omega}(k)}$ over a mini-batch $\mathcal{Z} \sim \mathcal{D}^n$ of $n = \mathrm{poly}(k)$ samples:

$$\frac{1}{n} \sum_{(X,y) \in \mathcal{Z}} \nabla_b \ell(f_t(X), y) \geq C.$$

Proof. Consider an arbitrary $t \leq T$ such that $c_t \geq C_2/\log^4 k$.

$$\nabla_b \mathbb{E}[\ell(f_t(X), y)] = \sum_{i \in I} \underbrace{\mathbb{E}\left[\sigma'(\overline{-Y f_{-i}(X)}) \left(\mathrm{ReLU}(Z_i + b_t) + \mathrm{ReLU}(-Z_i + b_t)\right)\right]}_{(1)}$$
$$- \underbrace{\mathbb{E}\left[Y \sigma(-Y f_t(X)) \left(\mathbb{I}\{c_t Y \geq -b_t\} - \mathbb{I}\{c_t Y \leq b_t\}\right)\right]}_{(2)},$$

where (1) comes from a first-order Taylor expansion of the sigmoid function σ around $-Y f_{-i}(X)$, σ' is the first derivative of σ, and $\overline{-Y f_{-i}(X)}$ is some value between $-Y f_{-i}(X)$ and $-Y f_t(X)$ so that $\sigma(-Y f_t(X)) = \sigma(-Y f_{-i}(X)) + \sigma'(\overline{-Y f_{-i}(X)})(-Y f_t(X) + Y f_{-i}(X))$ holds with equality.

By Corollary B.1, $\mathrm{Var}[S_i] \geq \beta(c_t + b_t)^2/k$.

By Lemma B.9, when $b_t < 0$, we have

$$\mathbb{E}[|S_i|] \geq C_1 \sqrt{\frac{\mathrm{Var}[S_i]}{k^{3/4}}} \geq C_1 \sqrt{\beta} \frac{c_t + b_t}{k^{7/8}}.$$

When $b_t \geq 0$ instead of < 0, this only increases $\mathbb{E}[|S_i|]$. Combined with Lemma B.5, we conclude that, for any b_t, we have:

$$\sum_{i \in I} \mathbb{E}[|S_i|] \geq C_1' c_t k^{1/8},$$

for some constant $C_1' > 0$.

Let E_1 be the event that $f_{-i}(X) \leq C + 10$. Let E_2 be the event that $f_i(X) \leq 1$. Let $E = E_1 \cap E_2$. Under E and Lemma B.7, $\sigma'(\overline{-Yf_{-i}(X)})$ is at least some constant $D > 0$.

$$\begin{aligned}(1) &= \mathbb{E}\left[\sigma'(\overline{-Yf_{-i}(X)})\left(\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b)\right) \mid E_1, E_2\right]\mathbb{P}[E_1]\mathbb{P}[E_2] \\ &+ \mathbb{E}\left[\sigma'(\overline{-Yf_{-i}(X)})\left(\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b)\right) \mid \neg E\right]\mathbb{P}[\neg E] \\ &\geq D' \cdot \mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b) \mid E_2],\end{aligned}$$

for some constant $D' = D \cdot \mathbb{P}[E]$.

Finally, we related the conditioned expectation to the unconditioned expectation as follows:

$$\begin{aligned}&\mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b)] \\ &= \mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b) \mid E_2]\mathbb{P}[E_2] \\ &+ \mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b) \mid \neg E_2]\mathbb{P}[\neg E_2] \\ &= \mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b) \mid E_2](1 - e^{-\widetilde{\Omega}(k)}) + e^{-\widetilde{\Omega}(k)},\end{aligned}$$

so we have

$$\mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b) \mid E_2] = \frac{\mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b)] - e^{-\widetilde{\Omega}(k)}}{1 - e^{-\widetilde{\Omega}(k)}}.$$

Thus, $(1) \geq A \cdot \mathbb{E}[\text{ReLU}(Z_i + b) + \text{ReLU}(-Z_i + b)]$, for some constant $A > 0$.

We have a trivial upper bound on (2) of 1 (because σ and the indicators are bounded).

Therefore, for $c_t \geq C_2/\log^4 k$, as specified, we have $\nabla_b \mathbb{E}[\ell(f_t(X), y)] = k \cdot (1) - (2) \geq 2C$ for sufficiently large k.

Finally, by Lemma B.4, we have the following with probability at least $1 - e^{-\widetilde{\Omega}(k)}$, over a mini-batch $\mathcal{Z} \sim \mathcal{D}^n$ of $n = \text{poly}(k)$ samples:

$$\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}} \nabla_b \ell(f_t(X), y) \geq C.$$

□

Conclusion: efficiency in time and sample complexity

Theorem: Theorem 3.1

There exists an absolute constant k_0 such that, for every $k \geq k_0$, using $\text{poly}(k)$ samples from $\mathcal{D}$, learning rate $\eta = 1/\text{poly}(k)$, and $T = \text{poly}(k)$ iterations, w.h.p. over the randomness of the initialization and the samples, we have $\Pr_{(X,y)\sim\mathcal{D}}[\text{sign}(f_T(X)) \neq y] \leq 0.01$, for the final network f_T returned by Algorithm 3.1.

Proof. Throughout, consider the largest k_0 such that all intermediate lemmas hold for all $k \geq k_0$. Then, for all $k \geq k_0$, we have the following.

By Lemma 3.3, there exist constants $C, C_2 > 0$ such that, for all $k \geq k_0$, for any $t \leq T$ such that $c_t \geq C_2/\log^4 k$, the following holds with probability at least $1 - e^{-\widetilde{\Omega}(k)}$ over a mini-batch $\mathcal{Z} \sim \mathcal{D}^n$ of $n = \text{poly}(k)$ samples:

$$\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}} \nabla_b \ell(f_t(X), y) \geq C.$$

By Lemma 3.1, plugging in $H = 2C_2$, there exists a constant $A > 0$ such that, for all $k \geq k_0$, for any $t \leq T$ such that $|c_t| \leq 2C_2/\log^4 k$, we have the following with probability at least $1 - e^{-\widetilde{\Omega}(k)}$ over a mini-batch $\mathcal{Z} \sim \mathcal{D}^n$ of $n = \text{poly}(k)$ samples:

$$\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}} \nabla_{\mathbf{w}} \ell(f_t(X), y) \cdot \mathbf{w}_\star \leq -A.$$

Therefore, within $T_1 = (\frac{2C_2}{\log^4 k} + \delta)\frac{1}{\eta A} = \text{poly}(k)$ iterations, c_t will rise to $2C_2/\log^4 k$. Since we only have $\text{poly}(k)$ iterations, the overall failure probability for this first phase is still $e^{-\widetilde{\Omega}(k)}$. Although c_t is not guaranteed to continue *increasing*, Lemma 3.1 guarantees that c_t will not drop below $C_2/\log^4 k$ after it reaches $2C_2/\log^4 k$.

Once c_t has reached $C_2/\log^4 k$, Lemma 3.3 says that, with probability at least $1 - e^{-\widetilde{\Omega}(k)}$ per iteration, $\frac{1}{n}\sum_{(X,y)\in\mathcal{Z}} \nabla_b \ell(f_t(X), y) \geq C$. Thus, b_t will decrease at a rate of $\eta_b C$ per iteration.

By Lemma 3.2, while $t \leq \text{poly}(k)$, we have $v_t \leq 2\delta$. Since $c_t \geq C_2/\log^4 k$ and $v_t \leq 2\delta$, we have (loosely) $s_t \leq 2c_t\sigma$, and so $\frac{|b_t|}{s_t} \geq \frac{|b_t|}{2c_t\sigma}$.

By Lemma B.5, once we have $\alpha_t = |b_t|/s_t > 2\sqrt{\log k}$, we will have $\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(f_t(X)) \neq y] < 0.01$. By Lemma B.7, there exists a constant $C_3 > 0$ such that $c_t \leq C_3$. Thus, once $\frac{|b_t|}{2C_3\sigma} > 2\sqrt{\log k}$, or equivalently $|b_t| > 4C_3\frac{\log^{3/2} k}{\sqrt{k}}$, we will have $\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(f_t(X)) \neq y] < 0.01$.

Since b is decreasing at a rate of at least $\eta_b C$ per iteration, this occurs within $T_2 = (4C_3\frac{\log^{3/2} k}{\sqrt{k}} + \frac{C_2}{\log^4 k}\frac{\eta_b}{\eta})\frac{1}{\eta_b C}$ iterations, where $\frac{C_2}{\log^4 k}\frac{\eta_b}{\eta}$ accounts for b's initial growth at the beginning of training.

Since $T_1 + T_2 = \text{poly}(k)$, we reach final classification error ≤ 0.01 within $\text{poly}(k)$ iterations (assuming the $\text{poly}(k)$ in Lemma 3.2 is at least $T_1 + T_2$).

Since a mini-batch of size $n = \text{poly}(k)$ is used per iteration, the final sample complexity over $\text{poly}(k)$ total iterations is also $\text{poly}(k)$.

Since the failure probability per iteration is $e^{-\widetilde{\Omega}(k)}$, the total failure probability over initialization and $\text{poly}(k)$ iterations is also $e^{-\widetilde{\Omega}(k)}$. This completes the proof.

□

B.2 Full Proofs for Section 3.5

Lemma B.10: Small ball probability

Consider n independent Rademacher random variables ξ_i for $i \in [n]$ and constants a_i for $i \in [n]$ such that $|a_i| \geq 1$. Then, for any length-2Δ interval B, for $\Delta > 0$, we have $\mathbb{P}\left[\sum_{i\in[n]} \xi_i a_i \in B\right] \leq \frac{s\sqrt{\frac{2}{\pi}}+o(1)}{\sqrt{n}}$ whenever $s \leq n$ and $s - 1 \leq \Delta < s$ for some natural number s.

B.2.1 Warm up: one filter

We first present a proof when there is only one filter, to help elucidate the proof skeleton. Then, in Section B.2.2, we provide the full proof of Theorem 3.2. Readers can ignore this subsection and skip directly to Section B.2.2 if they like; this subsection is merely provided to help make some of the themes in Section B.2.2 a bit clearer.

Theorem B.1: $m = 1$

There exists $k_1 \in \mathbb{N}$ such that, for all $k \geq k_1$, with probability at least 0.999 over the random initialization $\mathbf{w}^{(0)} \sim \mathcal{N}(0, \sigma_0^2 I_d)$, the following holds for all $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$:

$$\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y] \geq 0.1.$$

Proof. We recall that the finite-width CNTK is defined as:

$$k_{\mathbf{w}}(X) = \sum_{i\in[k+1]} \langle \mathbf{w}, X_i\rangle 1_{|\langle \mathbf{w}^{(0)}, X_i\rangle|+b\geq 0}.$$

This comes from the gradient of the CNN function $f_{\mathbf{w},b}(X)$ with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} f_{\mathbf{w},b}(X) = \sum_{i=1}^{k+1} X_i \left[\mathbb{I}\{\langle \mathbf{w}, X_i\rangle + b > 0\} + \mathbb{I}\{-\langle \mathbf{w}, X_i\rangle + b > 0\}\right].$$

For convenience, for a single row of the input image X, denoted $x \in \mathbb{R}^d$, we define

$$g(x) := \langle \mathbf{w}, x\rangle 1_{|\langle \mathbf{w}^{(0)}, x\rangle|+b\geq 0},$$

which represents the total contribution to $k_{\mathbf{w}}(X)$ coming from x. We thus have

$$k_{\mathbf{w}}(X) = \sum_{i\in[k+1]} g(X_i) = g(y\mathbf{w}_\star) + \sum_{i\in I} g(\epsilon_i).$$

If $y(g(y\mathbf{w}_\star)) \leq 0$, then by symmetry of $g(\epsilon_i)$, we have $\mathbb{P}_{X,y\sim\mathcal{D}}\left[y\sum_{i\in I} g(\epsilon_i) \leq 0\right] = 0.5$, so

$$\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y] \geq 0.5 > 0.1.$$

It thus remains to consider the more complex case $y(g(y\mathbf{w}_\star)) > 0$.

There exists some constant $D_{0.999} > 0$ such that

$$\mathbb{P}_{\mathbf{w}^{(0)} \sim \mathcal{N}(0, \sigma_0^2 I)} \left[\underbrace{\left| \left\langle \frac{\mathbf{w}^{(0)}}{\|\mathbf{w}^{(0)}\|}, \mathbf{w}_\star \right\rangle \right| \leq \frac{D_{0.999}}{\sqrt{d}}}_{\text{Init Event}} \right] \geq 0.999.$$

We assume $\mathbf{w}^{(0)}$ satisfies the Init Event. Then, by Lemma B.11, for any $\mathbf{w}, b$, there exist constants $C > 0, p_C \in (0, 1]$ such that

$$p := \mathbb{P}_{\epsilon_i \sim \mathcal{N}(0, \sigma^2 I_d)} \left[\underbrace{|g(\epsilon_i)| \geq C\sigma |g(\mathbf{w}_\star)|}_{\text{Event } A} \right] \geq p_C.$$

Since each noise row has the same distribution, the same C, p, p_C hold for all $i \in I$.

Let $I^+ := \{i \in I : |g(\epsilon_i)| \geq C\sigma |g(\mathbf{w}_\star)|\}$ and let $I^- := I \setminus I^+$.

By Chernoff, $\mathbb{P}[|I^+| \leq 0.5kp] \leq e^{-0.5^2 kp/2}$.

Applying Lemma B.10, we obtain:

$$\begin{aligned}
\mathbb{P}\left[\sum_{i \in I^+} g(\epsilon_i) \in \left[-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)| \right] \right] &= \mathbb{P}\left[\sum_{i \in I^+} \frac{g(\epsilon_i)}{C\sigma |g(\mathbf{w}_\star)|} \in \left[-\frac{|g(\mathbf{w}_\star)|}{C\sigma |g(\mathbf{w}_\star)|}, \frac{|g(\mathbf{w}_\star)|}{C\sigma |g(\mathbf{w}_\star)|} \right] \right] \\
&= \mathbb{P}\left[\sum_{i \in I^+} \frac{g(\epsilon_i)}{C\sigma |g(\mathbf{w}_\star)|} \in \left[-\frac{1}{C\sigma}, \frac{1}{C\sigma} \right] \right] \\
&= \mathcal{O}\left(\frac{1}{\sigma \sqrt{|I^+|}} \right).
\end{aligned}$$

Thus, the **probability of misclassification** $\mathbb{P}_{(X,y) \sim \mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y]$ is at least:
(We have $\text{sign}(0) := 0$, so $k_{\mathbf{w}}(X) = 0$ is necessarily a misclassification event.)

$$\begin{aligned}
\mathbb{P}_{(X,y) \sim \mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y] &= \mathbb{P}_{(X,y) \sim \mathcal{D}}[y k_{\mathbf{w}}(X) \leq 0] \\
&= \mathbb{P}_{(X,y) \sim \mathcal{D}}\left[y \sum_{i \in I} g(\epsilon_i) \leq -y g(\mathbf{w}_\star) \right] \\
&\geq \mathbb{P}_{(X,y) \sim \mathcal{D}}\left[\sum_{i \in I} g(\epsilon_i) \geq |g(\mathbf{w}_\star)| \right] \\
&\geq \mathbb{P}_{(X,y) \sim \mathcal{D}}\left[\sum_{i \in I^+} g(\epsilon_i) \geq |g(\mathbf{w}_\star)|, \sum_{i \in I^-} g(\epsilon_i) \geq 0 \right] \\
&= \mathbb{P}_{(X,y) \sim \mathcal{D}}\left[\sum_{i \in I^+} g(\epsilon_i) \geq |g(\mathbf{w}_\star)| \right] \cdot \mathbb{P}_{(X,y) \sim \mathcal{D}}\left[\sum_{i \in I^-} g(\epsilon_i) \geq 0 \right]
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \geq |g(\mathbf{w}_\star)|\right] \\
&= \frac{1}{4}\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \notin \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right)\right] \\
&= \frac{1}{4}\left(1 - \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right)\right]\right)
\end{aligned}
$$

$$
\begin{aligned}
&\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right)\right] \\
&= \underbrace{\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right) \mid |I^+| \leq 0.5kp\right]}_{\leq 1} \mathbb{P}[|I^+| \leq 0.5kp] \\
&+ \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right) \mid |I^+| > 0.5kp\right] \underbrace{\mathbb{P}[|I^+| > 0.5kp]}_{\leq 1} \\
&\leq e^{-0.5^2 kp/2} + \mathcal{O}\left(\frac{1}{\sigma\sqrt{kp}}\right) \\
&\leq \mathcal{O}\left(\frac{1}{(\log k)\sqrt{p_C}}\right) \\
&= \mathcal{O}\left(\frac{1}{\log k}\right).
\end{aligned}
$$

Putting this together, we get

$$
\mathbb{P}_{(X,y)\sim\mathcal{D}}[\operatorname{sign}(k_{\mathbf{w}}(X)) \neq y] \geq \frac{1}{4}\left(1 - \mathcal{O}(1/\log k)\right).
$$

Thus, there exists $k_1 \in \mathbb{N}$ such that, for all $k \geq k_1$, with probability at least 0.999 over the random initialization $\mathbf{w}^{(0)} \sim \mathcal{N}(0, \sigma_0^2 I_d)$, the following holds for all $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$:

$$
\mathbb{P}_{(X,y)\sim\mathcal{D}}[\operatorname{sign}(k_{\mathbf{w}}(X)) \neq y] \geq 0.1.
$$

□

Lemma B.11: $m = 1$

Assume $\mathbf{w}^{(0)}$ satisfies the Init Event. Then, for any $\mathbf{w}, b$, there exist constants $C > 0, p_C \in$

$(0, 1]$ such that

$$\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}\Big[\underbrace{|g(\epsilon)| \geq C\sigma|g(\mathbf{w}_\star)|}_{\text{Event } A}\Big] \geq p_C.$$

Proof. As a reminder, in this lemma, we are looking exclusively at the $m = 1$ case. When $m = 1$, we have:

$$g(\mathbf{w}_\star) = \langle \mathbf{w}, \mathbf{w}_\star\rangle 1_{|\langle \mathbf{w}^{(0)}, \mathbf{w}_\star\rangle|+b\geq 0}.$$

We now consider different possible settings of $\mathbf{w}^{(0)}, \mathbf{w}, b$, via a case analysis.

Case 1: Suppose $|\langle \mathbf{w}^{(0)}, \mathbf{w}_\star\rangle| + b < 0$. Then $g(\mathbf{w}_\star) = 0$. $|g(\epsilon)| \geq 0$, so for all $C > 0$, $p_C = 1$ satisfies the lemma statement.

Case 2: Suppose $|\langle \mathbf{w}^{(0)}, \mathbf{w}_\star\rangle| + b \geq 0$. Then $g(\mathbf{w}_\star) = \langle \mathbf{w}, \mathbf{w}_\star\rangle$, so Event A becomes:

$$|\langle \mathbf{w}, \epsilon\rangle 1_{|\langle \mathbf{w}^{(0)}, \epsilon\rangle|+b\geq 0}| \geq C\sigma|\langle \mathbf{w}, \mathbf{w}_\star\rangle|.$$

Let E be the event that $|\langle \mathbf{w}^{(0)}, \epsilon\rangle| \geq |\langle \mathbf{w}^{(0)}, \mathbf{w}_\star\rangle|$. We are introducing E so that we can condition A on E and thus simplify our analysis of $\mathbb{P}[A]$. We can simplify $\mathbb{P}[E]$ as follows, for use later:

$$\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[E] = \mathbb{P}_{x\sim\mathcal{N}(0,\sigma^2)}\left[|x| \geq \left|\left\langle \frac{\mathbf{w}^{(0)}}{\|\mathbf{w}^{(0)}\|}, \mathbf{w}_\star\right\rangle\right|\right].$$

We therefore analyze $\mathbb{P}[A]$ as follows:

$$\begin{aligned}
\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[A] &= \mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[A \mid E]\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[E] + \mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[A \mid \neg E]\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[\neg E] \\
&\geq \mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[A \mid E]\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[E] \\
&= \mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[|\langle \mathbf{w}, \epsilon\rangle| \geq C\sigma|\langle \mathbf{w}, \mathbf{w}_\star\rangle| \mid E]\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[E] \\
&\geq \mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[|\langle \mathbf{w}, \epsilon\rangle| \geq C\sigma|\langle \mathbf{w}, \mathbf{w}_\star\rangle|]\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[E] \\
&= \mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}\left[\frac{1}{\sigma}\left|\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \epsilon\right\rangle\right| \geq C\left|\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{w}_\star\right\rangle\right|\right]\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[E] \\
&= \mathbb{P}_{x\sim\mathcal{N}(0,1)}\left[|x| \geq C\left|\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{w}_\star\right\rangle\right|\right]\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}[E] \\
&= \mathbb{P}_{x\sim\mathcal{N}(0,1)}\left[|x| \geq C\left|\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{w}_\star\right\rangle\right|\right]\mathbb{P}_{x\sim\mathcal{N}(0,\sigma^2)}\left[|x| \geq \left|\left\langle \frac{\mathbf{w}^{(0)}}{\|\mathbf{w}^{(0)}\|}, \mathbf{w}_\star\right\rangle\right|\right]
\end{aligned}$$

The first probability depends on C and the angle between $\mathbf{w}$ and $\mathbf{w}_\star$.
The second probability depends on σ and the angle between $\mathbf{w}^{(0)}$ and $\mathbf{w}_\star$.

□

B.2.2 Multiple filters

Throughout, we use the shorthand $\mathbf{w}^{(0)}$ to denote $\{\mathbf{w}_j^{(0)}\}_{j\in[m]}$, $\mathbf{w}$ to denote $\{\mathbf{w}_j\}_{j\in[m]}$, and b to denote $\{b_j\}_{j\in[m]}$.

Theorem: Theorem 3.2

For any $C > 0$, if there exists k_0 such that $m \leq C$ for all $k \geq k_0$, then there exists $k_1 \geq k_0$ such that, for all $k \geq k_1$, with probability at least 0.999 over the random initialization $\{\mathbf{w}_j^{(0)}\}_{j\in[m]}$ where each $\mathbf{w}_j^{(0)}$ i.i.d. $\sim \mathcal{N}(0, \sigma_0^2 I)$, the following holds for *every* set of weights $\mathbf{w} := \{\mathbf{w}_j\}_{j\in[m]}$ and *every* set of biases $b := \{b_j\}_{j\in[m]}$,

$$\Pr_{X,y\sim\mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y] \geq 0.1.$$

Proof. We recall that the finite-width CNTK is defined as:

$$k_{\mathbf{w}}(X) = \sum_{i\in[k+1]} \sum_{j\in[m]} \langle \mathbf{w}_j, X_i\rangle 1_{|\langle \mathbf{w}_j^{(0)}, X_i\rangle| + b_j \geq 0}.$$

This comes from the gradient of the CNN function $f_{\mathbf{w},b}(X)$ with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} f_{\mathbf{w},b}(X) = \sum_{i=1}^{k+1} X_i \left[\mathbb{I}\{\langle \mathbf{w}, X_i\rangle + b > 0\} + \mathbb{I}\{-\langle \mathbf{w}, X_i\rangle + b > 0\}\right].$$

There exist some positive constants $D_{0.999}^{(l)}, D_{0.999}^{(u)}, D_{0.999}^{(p)} > 0$ such that

$$\mathbb{P}_{\mathbf{w}^{(0)}}\Bigg[\underbrace{\forall j \in [m], \frac{D_{0.999}^{(l)}}{\sqrt{d}} \leq \left|\left\langle \frac{\mathbf{w}_j^{(0)}}{\|\mathbf{w}_j^{(0)}\|}, \mathbf{w}_\star \right\rangle\right| \leq \frac{D_{0.999}^{(u)}}{\sqrt{d}}}_{\text{Init Event}} \bigcap$$
$$\forall (j,j') \in \mathcal{J}, \left|\left\langle \frac{\mathbf{w}_j^{(0)}}{\|\mathbf{w}_j^{(0)}\|_2}, \frac{\mathbf{w}_{j'}^{(0)}}{\|\mathbf{w}_{j'}^{(0)}\|_2} \right\rangle\right| \leq \frac{D_{0.999}^{(p)}}{\sqrt{d}}\Bigg] \geq 0.999.$$

For convenience, for a single row of the input image X, denoted $x \in \mathbb{R}^d$, we define

$$g(x) := \sum_{j\in[m]} \langle \mathbf{w}_j, x\rangle 1_{|\langle \mathbf{w}_j^{(0)}, x\rangle| + b_j \geq 0},$$

which represents the total contribution to $k_{\mathbf{w}}(X)$ coming from x. We thus have

$$k_{\mathbf{w}}(X) = \sum_{i\in[k+1]} g(X_i) = g(y\mathbf{w}_\star) + \sum_{i\in I} g(\epsilon_i).$$

Throughout the remainder of the proof, we consider arbitrary initialization $\mathbf{w}^{(0)}$ satisfying the above criteria and arbitrary $\mathbf{w}, b$.

If $y(g(y\mathbf{w}_\star)) \leq 0$, then by symmetry of $g(\epsilon_i)$, we have $\mathbb{P}_{X,y\sim\mathcal{D}}\left[y\sum_{i\in I} g(\epsilon_i) \leq 0\right] = 0.5$, so

$$\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y] \geq 0.5 > 0.1.$$

It thus remains to consider the more complex case $y(g(y\mathbf{w}_\star)) > 0$.

By Lemma B.12, for any $\mathbf{w}, b$, there exist constants $C > 0, p_C \in (0, 1]$ such that

$$p := \mathbb{P}_{\epsilon_i \sim \mathcal{N}(0,\sigma^2 I_d)}\Big[\underbrace{|g(\epsilon_i)| \geq C\sigma|g(\mathbf{w}_\star)|}_{\text{Event } A}\Big] \geq p_C.$$

Since each noise row has the same distribution, the same C, p, p_C hold for all $i \in I$.

Let $I^+ := \{i \in I : |g(\epsilon_i)| \geq C\sigma|g(\mathbf{w}_\star)|\}$ and let $I^- := I \setminus I^+$.

By Chernoff, $\mathbb{P}[|I^+| \leq 0.5kp] \leq e^{-0.5^2 kp/2}$.

Applying Lemma B.10, we obtain:

$$\begin{aligned}
\mathbb{P}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left[-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right]\right] &= \mathbb{P}\left[\sum_{i\in I^+} \frac{g(\epsilon_i)}{C\sigma|g(\mathbf{w}_\star)|} \in \left[-\frac{|g(\mathbf{w}_\star)|}{C\sigma|g(\mathbf{w}_\star)|}, \frac{|g(\mathbf{w}_\star)|}{C\sigma|g(\mathbf{w}_\star)|}\right]\right] \\
&= \mathbb{P}\left[\sum_{i\in I^+} \frac{g(\epsilon_i)}{C\sigma|g(\mathbf{w}_\star)|} \in \left[-\frac{1}{C\sigma}, \frac{1}{C\sigma}\right]\right] \\
&= \mathcal{O}\left(\frac{1}{\sigma\sqrt{|I^+|}}\right).
\end{aligned}$$

Thus, the **probability of misclassification** $\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y]$ is at least: (We have $\text{sign}(0) := 0$, so $k_{\mathbf{w}}(X) = 0$ is necessarily a misclassification event.)

$$\begin{aligned}
\mathbb{P}_{(X,y)\sim\mathcal{D}}[\text{sign}(k_{\mathbf{w}}(X)) \neq y] &= \mathbb{P}_{(X,y)\sim\mathcal{D}}[yk_{\mathbf{w}}(X) \leq 0] \\
&= \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[y\sum_{i\in I} g(\epsilon_i) \leq -yg(\mathbf{w}_\star)\right] \\
&\geq \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I} g(\epsilon_i) \geq |g(\mathbf{w}_\star)|\right] \\
&\geq \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \geq |g(\mathbf{w}_\star)|, \sum_{i\in I^-} g(\epsilon_i) \geq 0\right] \\
&= \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \geq |g(\mathbf{w}_\star)|\right] \cdot \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^-} g(\epsilon_i) \geq 0\right] \\
&= \frac{1}{2}\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \geq |g(\mathbf{w}_\star)|\right] \\
&= \frac{1}{4}\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \notin \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right)\right] \\
&= \frac{1}{4}\left(1 - \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right)\right]\right)
\end{aligned}$$

$$
\begin{aligned}
&\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right)\right] \\
&= \underbrace{\mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right) \mid |I^+| \le 0.5kp\right]}_{\le 1} \mathbb{P}[|I^+| \le 0.5kp] \\
&+ \mathbb{P}_{(X,y)\sim\mathcal{D}}\left[\sum_{i\in I^+} g(\epsilon_i) \in \left(-|g(\mathbf{w}_\star)|, |g(\mathbf{w}_\star)|\right) \mid |I^+| > 0.5kp\right] \underbrace{\mathbb{P}[|I^+| > 0.5kp]}_{\le 1} \\
&\le e^{-0.5^2 kp/2} + \mathcal{O}\left(\frac{1}{\sigma\sqrt{kp}}\right) \\
&\le \mathcal{O}\left(\frac{1}{(\log k)\sqrt{p_C}}\right) \\
&= \mathcal{O}\left(\frac{1}{\log k}\right).
\end{aligned}
$$

Putting this together, we get

$$
\mathbb{P}_{(X,y)\sim\mathcal{D}}[\operatorname{sign}(k_{\mathbf{w}}(X)) \neq y] \ge \frac{1}{4}\left(1 - \mathcal{O}(1/\log k)\right).
$$

Thus, there exists $k_1 \in \mathbb{N}$ such that, for all $k \ge k_1$, with probability at least 0.999 over the random initialization $\mathbf{w}^{(0)} \sim \mathcal{N}(0, \sigma_0^2 I_d)$, the following holds for all $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$:

$$
\mathbb{P}_{(X,y)\sim\mathcal{D}}[\operatorname{sign}(k_{\mathbf{w}}(X)) \neq y] \ge 0.1.
$$

□

Lemma B.12: general case: $1 \le m \le C$

Assume $\mathbf{w}^{(0)}$ satisfies the Init Event. Then, for any $\mathbf{w}, b$, there exist constants $C > 0, p_C \in (0, 1]$ such that

$$
\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}\Big[\underbrace{|g(\epsilon)| \ge C\sigma|g(\mathbf{w}_\star)|}_{\text{Event } A}\Big] \ge p_C.
$$

Proof. We first consider the trivial case where $g(\mathbf{w}_\star) = 0$. We necessarily have $|g(\epsilon)| \ge 0$, so for all $C > 0$, $p_C = 1$ satisfies the lemma statement. So throughout the rest of this proof, we focus on the case where $g(\mathbf{w}_\star) \neq 0$.

For any $\mathbf{w} \in \mathbb{R}^d$, define

$$\mathcal{M}(\mathbf{w}) := \{j \in [m] : |\langle \mathbf{w}_j^{(0)}, \mathbf{w}\rangle| + b_j \geq 0\}.$$

Note that, for any $\mathbf{w}$, $\mathcal{M}(\mathbf{w})$ depends on the random initialization $\mathbf{w}^{(0)}$ and the chosen bias b.

We use $\mathcal{M}$ to write $g(\mathbf{w}_\star), g(\epsilon)$ as follows:

$$g(\mathbf{w}_\star) = \sum_{j\in[m]} \langle \mathbf{w}_j, \mathbf{w}_\star\rangle 1_{|\langle \mathbf{w}_j^{(0)}, \mathbf{w}_\star\rangle|+b_j\geq 0} = \sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \langle \mathbf{w}_j, \mathbf{w}_\star\rangle = \left\langle \sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j, \mathbf{w}_\star \right\rangle$$
$$g(\epsilon) = \sum_{j\in[m]} \langle \mathbf{w}_j, \epsilon\rangle 1_{|\langle \mathbf{w}_j^{(0)}, \epsilon\rangle|+b_j\geq 0} = \sum_{j\in\mathcal{M}(\epsilon)} \langle \mathbf{w}_j, \epsilon\rangle = \left\langle \sum_{j\in\mathcal{M}(\epsilon)} \mathbf{w}_j, \epsilon \right\rangle.$$

We first show that $\mathbb{P}[\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star)] \geq q$, for some constant $q > 0$, the proof of which continues until (B.9).

For $j \in [m]$, let $\hat{v}_j := \frac{\mathbf{w}_j^{(0)}}{\|\mathbf{w}_j^{(0)}\|}$, and let $\{u_j\}_{j\in[m]}$ be the unnormalized output of Gram-Schmidt applied to $\{\hat{v}_j\}_{j\in[m]}$. Formally, Gram-Schmidt will yield:

$$u_1 = \hat{v}_1$$
$$u_j = \hat{v}_j - \sum_{i=1}^{j-1} \frac{\langle \hat{v}_j, u_i\rangle}{\langle u_i, u_i\rangle} u_i.$$

We will show that the following invariant holds for all $j \in [m]$:

$$u_j = \hat{v}_j \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_1 \pm \cdots \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_{j-1}. \tag{B.7}$$

By the triangle inequality, (B.7) yields $\|u_j\| \leq 1 + \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)$ and $\|u_j\| \geq 1 - \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)$, which implies

$$\|u_j\| = \Theta(1). \tag{B.8}$$

For $j = 1$, (B.7) holds because $u_j = \hat{v}_1$ by definition.

For any $j > 1$, assuming (B.7) holds for all $i < j$ (and thus its corollary (B.8)), we have

$$u_j = \hat{v}_j - \sum_{i=1}^{j-1} \frac{\left\langle \hat{v}_j, \hat{v}_i \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_1 \pm \cdots \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_{i-1}\right\rangle}{\|u_i\|^2} \left[\hat{v}_i \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_1 \pm \cdots \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_{i-1}\right]$$
$$= \hat{v}_j \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_1 \pm \cdots \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_{j-1}.$$

With invariants (B.7) and (B.8) holding for all $j \in [m]$, we can now lower bound $\mathbb{P}[\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star)]$.

We first define E_1 as the event that, for all $j \in \mathcal{M}(\mathbf{w}_\star)$, $|\langle \hat{v}_j, \epsilon \rangle| \geq |\langle \hat{v}_j, \mathbf{w}_\star \rangle|$.

We define E_2 as the event that, for all $j \in [m] \setminus \mathcal{M}(\mathbf{w}_\star)$, $|\langle \hat{v}_j, \epsilon \rangle| \leq |\langle \hat{v}_j, \mathbf{w}_\star \rangle|$.

We then have

$$\mathbb{P}[\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star)] \geq \mathbb{P}\left[E_1 \cap E_2\right].$$

We will then lower bound $\mathbb{P}\left[E_1 \cap E_2\right]$ by considering the probability that each $|\langle \hat{v}_j, \epsilon \rangle|$ belongs to a particular interval.

We define the two intervals

$$I^{(l)} := \left[\frac{D^{(l)}_{0.999}}{2\sqrt{d}}, \frac{D^{(l)}_{0.999}}{\sqrt{d}}\right], \;\; I^{(u)} := \left[\frac{D^{(u)}_{0.999}}{\sqrt{d}}, \frac{2D^{(u)}_{0.999}}{\sqrt{d}}\right].$$

Since the *Init Event* holds, we can see that $|\langle \hat{v}_j, \epsilon \rangle| \in I^{(l)} \implies |\langle \hat{v}_j, \epsilon \rangle| \leq |\langle \hat{v}_j, \mathbf{w}_\star \rangle|$.

Analogously, $|\langle \hat{v}_j, \epsilon \rangle| \in I^{(u)} \implies |\langle \hat{v}_j, \epsilon \rangle| \geq |\langle \hat{v}_j, \mathbf{w}_\star \rangle|$.

Thus, for all $j \in \mathcal{M}(\mathbf{w}_\star)$, let I_j represent $I^{(u)}$, and for all $j \in [m] \setminus \mathcal{M}(\mathbf{w}_\star)$, let I_j represent $I^{(l)}$.

We then have

$$\begin{aligned}
\mathbb{P}\left[E_1 \cap E_2\right] &\geq \mathbb{P}[(\forall j \in [m])\, |\langle \hat{v}_j, \epsilon \rangle| \in I_j] \\
&= \prod_{j=1}^{m} \mathbb{P}\Big[|\langle \hat{v}_j, \epsilon \rangle| \in I_j \;\Big|\; |\langle \hat{v}_1, \epsilon \rangle| \in I_1, \ldots, |\langle \hat{v}_{j-1}, \epsilon \rangle| \in I_{j-1}\Big].
\end{aligned}$$

We consider $\mathbb{P}\Big[|\langle \hat{v}_j, \epsilon \rangle| \in I_j \;\Big|\; |\langle \hat{v}_1, \epsilon \rangle| \in I_1, \ldots, |\langle \hat{v}_{j-1}, \epsilon \rangle| \in I_{j-1}\Big]$ for an arbitrary $j \in [m]$.

$$\begin{aligned}
&\mathbb{P}\Big[|\langle \hat{v}_j, \epsilon \rangle| \in I_j \;\Big|\; |\langle \hat{v}_1, \epsilon \rangle| \in I_1, \ldots, |\langle \hat{v}_{j-1}, \epsilon \rangle| \in I_{j-1}\Big] \\
&= \mathbb{P}\left[\left|\left\langle u_j \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_1 \pm \cdots \pm \mathcal{O}\left(\frac{1}{\sqrt{d}}\right)\hat{v}_{j-1}, \epsilon \right\rangle\right| \in I_j \;\middle|\; |\langle \hat{v}_1, \epsilon \rangle| \in I_1, \ldots, |\langle \hat{v}_{j-1}, \epsilon \rangle| \in I_{j-1}\right] \\
&= \mathbb{P}\left[\left|\langle u_j, \epsilon \rangle \pm \mathcal{O}\left(\frac{1}{d}\right)\right| \in I_j \;\middle|\; |\langle \hat{v}_1, \epsilon \rangle| \in I_1, \ldots, |\langle \hat{v}_{j-1}, \epsilon \rangle| \in I_{j-1}\right] \\
&= \mathbb{P}\left[\left|\langle u_j, \epsilon \rangle \pm \mathcal{O}\left(\frac{1}{d}\right)\right| \in I_j\right] \\
&= \mathbb{P}_{x \sim \mathcal{N}(0, \|u_j\|^2\sigma^2)}\left[\left|x \pm \mathcal{O}\left(\frac{1}{d}\right)\right| \in I_j\right] \\
&= \begin{cases}
\mathbb{P}_{x \sim \mathcal{N}(0, \|u_j\|^2\sigma^2)}\left[|x| \in \left[\frac{D^{(l)}_{0.999}}{2\sqrt{d}} + \mathcal{O}\big(\frac{1}{d}\big), \frac{D^{(l)}_{0.999}}{\sqrt{d}} - \mathcal{O}\big(\frac{1}{d}\big)\right]\right] & \text{if } I_j = I^{(l)} \\
\mathbb{P}_{x \sim \mathcal{N}(0, \|u_j\|^2\sigma^2)}\left[|x| \in \left[\frac{D^{(u)}_{0.999}}{\sqrt{d}} + \mathcal{O}\big(\frac{1}{d}\big), \frac{2D^{(u)}_{0.999}}{\sqrt{d}} - \mathcal{O}\big(\frac{1}{d}\big)\right]\right] & \text{if } I_j = I^{(u)}
\end{cases}.
\end{aligned}$$

Since $\|u_j\|\sigma = \Theta\left(\frac{1}{\sqrt{d}}\right)$, in either case the probability is at least a constant.

Thus, for some constant $q_j > 0$, we have

$$\mathbb{P}\left[|\langle \hat{v}_j, \epsilon\rangle| \in I_j \;\middle|\; |\langle \hat{v}_1, \epsilon\rangle| \in I_1, \ldots, |\langle \hat{v}_{j-1}, \epsilon\rangle| \in I_{j-1}\right] = q_j.$$

So we obtain

$$\mathbb{P}[\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star)] \geq \mathbb{P}\left[E_1 \cap E_2\right] \geq \mathbb{P}[(\forall j \in [m])\ |\langle \hat{v}_j, \epsilon\rangle| \in I_j] = \prod_{j=1}^{m} q_j =: q, \tag{B.9}$$

for some constant $q > 0$.

Let

$$\hat{v} := \frac{\sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j}{\|\sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j\|_2}.$$

Unless otherwise specified, $\mathbb{P}$ means $\mathbb{P}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I_d)}$:

$$\begin{aligned}
\mathbb{P}\Big[\underbrace{|g(\epsilon)| \geq C\sigma|g(\mathbf{w}_\star)|}_{\text{Event } A}\Big] &\geq \mathbb{P}\left[\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star), \left|\left\langle \sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j, \epsilon\right\rangle\right| \geq C\sigma|g(\mathbf{w}_\star)|\right] \\
1 - \mathbb{P}\Big[\underbrace{|g(\epsilon)| \geq C\sigma|g(\mathbf{w}_\star)|}_{\text{Event } A}\Big] &\leq 1 - \mathbb{P}\left[\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star), \left|\left\langle \sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j, \epsilon\right\rangle\right| \geq C\sigma|g(\mathbf{w}_\star)|\right] \\
&= \mathbb{P}\left[\neg\left(\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star), \left|\left\langle \sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j, \epsilon\right\rangle\right| \geq C\sigma|g(\mathbf{w}_\star)|\right)\right] \\
&= \mathbb{P}\left[\neg\Big(\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star)\Big) \cup \neg\left(\left|\left\langle \sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j, \epsilon\right\rangle\right| \geq C\sigma|g(\mathbf{w}_\star)|\right)\right] \\
&\leq 1 - \mathbb{P}[\mathcal{M}(\epsilon) = \mathcal{M}(\mathbf{w}_\star)] + \mathbb{P}\left[\left|\left\langle \sum_{j\in\mathcal{M}(\mathbf{w}_\star)} \mathbf{w}_j, \epsilon\right\rangle\right| < C\sigma|g(\mathbf{w}_\star)|\right] \\
&\leq 1 - q + \mathbb{P}\left[\frac{1}{\sigma}\left|\langle \hat{v}, \epsilon\rangle\right| < C\left|\langle \hat{v}, \mathbf{w}_\star\rangle\right|\right] \\
&= 1 - q + \mathbb{P}_{x\sim\mathcal{N}(0,1)}\left[|x| < C\left|\langle \hat{v}, \mathbf{w}_\star\rangle\right|\right].
\end{aligned}$$

There exists a $C > 0$ such that $\mathbb{P}_{x\sim\mathcal{N}(0,1)}\left[|x| < C\left|\langle \hat{v}, \mathbf{w}_\star\rangle\right|\right] = q/2$.

We thus have $1 - \mathbb{P}\Big[\underbrace{|g(\epsilon)| \geq C\sigma|g(\mathbf{w}_\star)|}_{\text{Event } A}\Big] \leq 1 - q/2$, so $\mathbb{P}\Big[\underbrace{|g(\epsilon)| \geq C\sigma|g(\mathbf{w}_\star)|}_{\text{Event } A}\Big] \geq q/2$.

□

B.3 Experiment Details

Details of all experiments are reported here. Code to reproduce the experiments is available at https://github.com/skarp/local-signal-adaptivity. All experiments were run on a single Nvidia GeForce RTX 2080 Ti GPU on an internal cluster and took between 2 and 10 hours per run. Models were implemented using the `jax` python library (Bradbury et al., 2018), with the `neural-tangents` library (Novak et al., 2020) used for the NTK implementation and some code borrowed from the `autol2` library (Lewkowycz and Gur-Ari, 2020). All three libraries are available under an Apache License, Version 2.0.

Synthetic data. The experiments with our model on synthetic data (Figure 3.3) were run on data with the following parameters: $k = 1000$, $d = 10$, and $\sigma = 1$. The model was trained for 10,000 steps of SGD with learning rates $\eta_w = 0.1$ and $\eta_b = 0.1/1000$ and minibatches of size 500, with new i.i.d. samples generated for each batch. Batch size and learning rate were chosen as large (resp. small) as possible given computational constraints, to best simulate the population-gradient setting.

CIFAR variants. All experiments on CIFAR variants, including the sparsity experiments in Figure 3.2, all experiments in Section 3.6, and the experiments in Appendix B.4 below, use the following:

- **Dataset details**. The ImageNet Plants synset was constructed using all WordNet IDs which are subclasses of the Plants ID n00017222. The CIFAR-Vehicles task is to classify between CIFAR-10 classes airplane, automobile, ship and truck, with animal classes as noise (bird, cat, deer, dog, frog, and horse). ImageNet and CIFAR-10 are both datasets of public-domain images. Our modifications do not add any personally-identifiable information or offensive content. Example images are included in the code repository.
- **Model architectures**. We use the same architectures as those in Allen-Zhu and Li, 2020a. **WRN**: we use a WideResNet WRN10-10, meaning depth=10 and widening factor k=10. This corresponds to a total of 3 residual blocks and 7 total convolutional layers. **NTK**: We use the corresponding finite-width (linearized) NTK of the WRN.
- **Training details**. We again largely follow the protocol of Allen-Zhu and Li, 2020a. However, we do not use data augmentation or Cutout since it is unclear how these may interact with the constructed noise images. **WRN**: Momentum optimizer with mass = 0.9, initial lr = 0.1, batch size = 128, weight decay = 0.0005, 200 epoch, and decay lr by 0.2 at epochs 80, 100, and 120 epochs. For the WRN experiments on larger images reported below, we reduced batch size to 50 due to computational constraints. In order to maintain stable training with this smaller batch size, we also had to reduce the initial learning rate to 0.05. **NTK**: Adam optimizer, initial lr = 0.001, batch size = 50, no weight decay, 200 epoch, and decay lr by 0.2 at epochs 140 and 170 epochs.

B.4 Additional Experiments

In Section 3.6 we saw that as the background noise level increases, NTK performance degrades significantly more quickly than WRN performance. Those experiments are done with 16x16 images on a 32x32 noise background, and all but the CIFAR-Vehicles experiment place the image at a random location on the background. Here we conduct additional experiments to show that the performance gap is not due to varying image location or small image size. Specifically, we replicate the experiment on CIFAR-2 with Gaussian noise, but with two additional image-placement methods and at a larger size of 32x32 images on a 64x64 background. The image is either placed at a random location on the background, in a random corner, or in fixed corner.

Table B.1: Test accuracy of WRN and NTK on CIFAR-2 with various levels of Gaussian noise. The image is placed on the noise background in a random location, random corner, or fixed corner. The image is sized 16x16 on a 32x32 background (first two rows) or 32x32 on a 64x64 background (last two rows).

		random location				**random corner**				**fixed corner**			
		noise σ				**noise σ**				**noise σ**			
size		**0.0**	**0.5**	**1.0**	**2.0**	**0.0**	**0.5**	**1.0**	**2.0**	**0.0**	**0.5**	**1.0**	**2.0**
16x16	**WRN**	94.7	94.0	92.8	91.7	94.3	94.2	92.7	92.6	94.4	94.3	94.0	92.9
on 32x32	**NTK**	89.9	87.3	83.7	76.9	91.6	88.3	84.5	78.3	92.0	88.8	85.2	80.0
32x32	**WRN**	96.1	96.2	95.8	94.9	96.5	96.5	96.5	95.9	96.9	96.6	96.0	95.6
on 64x64	**NTK**	91.2	89.4	84.6	82.2	91.8	89.1	85.3	79.1	93.1	91.0	88.3	82.9

Overall, we find that the gap in WRN vs NTK performance degradation persists in all cases. With respect to image size, we note that all models perform better on the larger images. However, the WRN appears slightly less impacted by noise on the larger images, while degradation affects the NTK roughly equally on large and small images. With respect to image placement, virtually all models perform best with fixed-corner placement, followed by random-corner, followed by random-location. It is not clear that either the WRN or NTK is more affected by placement than the other.

Appendix C

Landscape-Aware Growing: Additional Details

C.1 Training Details

C.1.1 BERT growing.

We train BERT-BASE on Books and Wikipedia following Devlin et al., 2018. We use batch size 256 and sequence length 512, AdamW as the optimizer (Loshchilov and Hutter, 2019), 10,000 steps of linear learning rate warmup, and a constant learning rate of 0.0001 following warmup. We train BERT-BASE for a total of 500,000 steps and then grow it from 12 layers to 16 layers. We ensure that all models see the data in the same order, to ensure that this does not impact relative performance.

C.1.2 UL2 growing.

Our pretrained UL2 model is a 12-layer decoder-only model with model dimension 2048, hidden dimension 5120, and 32 attention heads. We train this model using the UL2 objective (Tay et al., 2022) with 60% causal LM, 20% prefix LM, and 20% span corruption. We train on a mixture of C4 (57%) (Raffel et al., 2020), Wikipedia (17%) (Foundation, n.d.), GitHub (17%), and Arxiv (9%). We use AdaFactor (Shazeer and Stern, 2018) as the optimizer, with batch size 256, sequence length 1280, and 10,000 steps of linear warmup to a peak learning rate of 0.01, followed by a cosine decay schedule so that step 400,000 would end with learning rate 0.001. However, we pause training at 300,000 steps and then grow the 12-layer model to 16 layers, after which we train for another 100,000 steps (totaling 400,000 steps). We ensure that all models see the data in the same order, to ensure that this does not impact relative performance.

C.1.3 BERT stacking.

Here, we present the training details for last stacking and adaptive stacking. Broadly, we follow the stagewise paradigm of Reddi et al., 2023. We train BERT-LARGE on Books and Wikipedia following Devlin et al., 2018. We use batch size 256 and sequence length 512, AdamW as the optimizer (Loshchilov and Hutter, 2019), 10,000 steps of linear learning rate warmup, and a constant learning rate of 0.0001 following warmup. We train for a total of 1,000,000 steps divided among 6 stages. The first 5 stages have 160,000 steps each, and the final stage has 200,000 steps. In each stage, the total number of layers is increased by 4.

Last stacking. In each stage, we duplicate the *final* 4 layers and stack them at the end of the network (i.e., "post-stacking" from Reddi et al., 2023).

Adaptive stacking. In each stage, we consider a set of growth operators as in Section 5.2. To keep the design space more manageable, we restrict the design space to parameter duplication (i.e., no random initialization, beyond the very first stage). We train with each growth operator for 200 steps (i.e., LAG@200) before choosing the best-performing growth operator according to its validation loss and then continuing training with just this operator for the rest of the stage. The set of growth operators considered when initializing each stage is as follows:

- Stage 1 (4 layers): random initialization (no existing layers to use)
- Stage 2 (8 layers): indices $\{0\}$, block sizes $\{1, 2, 4\}$
- Stage 3 (12 layers): indices $\{0, 1, 2, 3, 4\}$, block sizes $\{1, 2, 4\}$
- Stage 4 (16 layers): indices $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$, block sizes $\{1, 2, 4\}$
- Stage 5 (20 layers): indices $\{0, 2, 4, 6, 8, 10, 12\}$, block sizes $\{1, 2, 4\}$
- Stage 6 (24 layers): indices $\{0, 2, 4, 6, 8, 10, 12, 14, 16\}$, block sizes $\{1, 2, 4\}$

Note that we switch from every index to only even indices at Stage 5, in order to keep the search space more manageable.

C.2 Further Empirical Results

Here, we present downstream metrics for UL2 and the adaptive stacking vs. last stacking results in table form.

Table C.1: Performance of LAG@2000 compared to other growing strategies, when growing UL2 from 12 layers to 16 layers, on 3 different downstream evaluations. The "Oracle" strategy refers to the best possible strategy within the search space. LAG@0 roughly follows the "loss preservation" heuristic (i.e., choosing the strategy whose loss is least perturbed by growing). The final two rows most closely resemble gradual stacking (Reddi et al., 2023): (1) stacking the last block on top, and (2) stacking a randomly-initialized block on top. Overall, across the 3 downstream metrics, LAG@2000 outperforms all strategies other than the "Oracle" strategy.

Strategy	**TyDi QA (en)** exact match - 1 shot	**Trivia QA** exact match - 1 shot	**LAMBADA** accuracy - 1 shot
Oracle	28.18	13.84	8.32
LAG@2000	25.23	13.44	7.60
Best at initialization (LAG@0)	21.82	12.19	6.89
Stack last block on top	21.59	12.39	6.90
Stack random block on top	21.82	12.19	6.89

Table C.2: Final validation loss of adaptive stacking vs. last stacking. In both adaptive stacking and last stacking, a 24-layer BERT-LARGE model is trained in 6 stages, using a roughly uniform stacking schedule (160,000 steps per stage for the first 5 stages, and 200,000 steps in the final stage). In adaptive stacking, we use $k = 200$ steps: at each stage of stacking, multiple growing strategies are spawned in parallel and trained for 200 steps, and then the strategy with the lowest validation loss is chosen to be trained for the rest of this stacking stage. Overall, we see that adaptive stacking outperforms last stacking (i.e., lower final validation loss).

Strategy	**Final validation loss**
Adaptive stacking	1.5301
Last stacking	1.5432

Bibliography

[1] Vladimir Vapnik and Alexey Chervonenkis. “On the uniform convergence of relative frequencies of events to their probabilities”. In: *Theory of Probability and its Applications*. 1971 (cit. on p. 1).

[2] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning internal representations by error-propagation”. In: (1986) (cit. on p. 1).

[3] Scott Fahlman and Christian Lebiere. “The cascade-correlation learning architecture”. In: *Advances in Neural Information Processing Systems 2* (1989) (cit. on p. 57).

[4] Avrim Blum and Ronald L. Rivest. “Training a 3-node neural network is NP-complete”. In: *Neural Networks*. 1992 (cit. on p. 1).

[5] Yann LeCun and Yoshua Bengio. “Convolutional networks for images, speech, and time series”. In: (1995) (cit. on p. 1).

[6] Peter L. Bartlett and Shai Ben-David. “Hardness results for neural network approximation problems”. In: *Theoretical Computer Science*. 2002 (cit. on p. 1).

[7] Peter L. Bartlett and Shahar Mendelson. “Rademacher and Gaussian Complexities: Risk Bounds and Structural Results”. In: *Journal of Machine Learning Research*. 2002 (cit. on p. 1).

[8] Olivier Bousquet and André Elisseeff. “Stability and generalization”. In: *Journal of Machine Learning* 2 (2002), pp. 499–526 (cit. on pp. 3, 4, 7).

[9] Samuel Kutin and Partha Niyogi. “Almost-everywhere Algorithmic Stability and Generalization Error”. In: *Proceedings of UAI*. 2002, pp. 275–282 (cit. on p. 8).

[10] Ryan O’Donnell and Rocco A. Servedio. “New Degree Bounds for Polynomial Threshold Functions”. In: *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*. STOC ’03. San Diego, CA, USA: Association for Computing Machinery, 2003, pp. 325–334. ISBN: 1581136749. DOI: 10.1145/780542.780592. URL: https://doi.org/10.1145/780542.780592 (cit. on p. 21).

[11] Steven Gutstein, Olac Fuentes, and Eric Freudenthal. “Knowledge Transfer in Deep Convolutional Neural Nets”. In: *International Journal on Artificial Intelligence Tools* (2008) (cit. on p. 57).

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 27).

[13] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: (2009) (cit. on p. 9).

[14] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: (2009) (cit. on p. 27).

[15] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 2011 (cit. on p. 9).

[16] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. "In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning". In: *arXiv preprint arXiv:1412.6614* (2014) (cit. on p. 2).

[17] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. "Net2net: Accelerating learning via knowledge transfer". In: *arXiv preprint arXiv:1511.05641* (2015) (cit. on p. 47).

[18] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. "Norm-based capacity control in neural networks". In: *Conference on Learning Theory*. 2015, pp. 1376–1401 (cit. on p. 7).

[19] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. "Net2net: Accelerating learning via knowledge transfer". In: *International Conference on Learning Representations* (2016) (cit. on p. 57).

[20] Moritz Hardt, Benjamin Recht, and Yoram Singer. "Train faster, generalize better: Stability of stochastic gradient descent". In: *Proceedings of the 33rd International Conference on Machine Learning*. 2016 (cit. on pp. 8, 14, 15, 62).

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016 (cit. on pp. 10, 62).

[22] Kenji Kawaguchi. "Deep learning without poor local minima". In: *Advances in Neural Information Processing Systems*. 2016, pp. 586–594 (cit. on p. 19).

[23] Yuanzhi Li, Yingyu Liang, and Andrej Risteski. "Recovery guarantee of non-negative matrix factorization via alternating updates". In: *Advances in neural information processing systems*. 2016, pp. 4987–4995 (cit. on p. 20).

[24] Daniel Soudry and Yair Carmon. "No bad local minima: Data independent training error guarantees for multilayer neural networks". In: *arXiv preprint arXiv:1605.08361* (2016) (cit. on p. 19).

[25] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. "Network Morphism". In: *Proceedings of the 33rd International Conference on Machine Learning*. 2016 (cit. on p. 57).

[26] Bo Xie, Yingyu Liang, and Le Song. "Diversity leads to generalization in neural networks". In: *arXiv preprint Arxiv:1611.03131* (2016) (cit. on p. 19).

[27] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *CoRR* abs/1605.07146 (2016). URL: http://arxiv.org/abs/1605.07146 (cit. on p. 27).

[28] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. “Spectrally-normalized margin bounds for neural networks”. In: *Advances in Neural Information Processing Systems*. 2017 (cit. on p. 2).

[29] Peter Bartlett, Dylan J Foster, and Matus Telgarsky. “Spectrally-normalized margin bounds for neural networks”. In: *arXiv preprint arXiv:1706.08498* (2017) (cit. on p. 7).

[30] Digvijay Boob and Guanghui Lan. “Theoretical properties of the global optimizer of two layer neural network”. In: *arXiv preprint arXiv:1710.11241* (2017) (cit. on p. 19).

[31] Alon Brutzkus and Amir Globerson. “Globally optimal gradient descent for a convnet with gaussian inputs”. In: *arXiv preprint arXiv:1702.07966* (2017) (cit. on p. 19).

[32] Gintare Karolina Dziugaite and Daniel M Roy. “Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data”. In: *arXiv preprint arXiv:1703.11008* (2017) (cit. on p. 7).

[33] Rong Ge, Jason D Lee, and Tengyu Ma. “Learning One-hidden-layer Neural Networks with Landscape Design”. In: *arXiv preprint arXiv:1711.00501* (2017) (cit. on p. 19).

[34] Yuanzhi Li and Yingyu Liang. “Provable alternating gradient descent for non-negative matrix factorization with strong correlations”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2062–2070 (cit. on p. 20).

[35] Yuanzhi Li and Yang Yuan. “Convergence analysis of two-layer neural networks with relu activation”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 597–607 (cit. on pp. 19, 20).

[36] Tongliang Liu, Gábor Lugosi, Gergely Neu, and Dacheng Tao. “Algorithmic Stability and Hypothesis Complexity”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017 (cit. on p. 8).

[37] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. “Exploring generalization in deep learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5947–5956 (cit. on p. 7).

[38] Behnam Neyshabur, Srinadh Bhojanapalli, David Mcallester, and Nati Srebro. “Exploring Generalization in Deep Learning”. In: *Advances in Neural Information Processing Systems*. 2017 (cit. on p. 2).

[39] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. “Theoretical insights into the optimization landscape of over-parameterized shallow neural networks”. In: *arXiv preprint arXiv:1707.04926* (2017) (cit. on p. 19).

[40] Yuandong Tian. “An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis”. In: *arXiv preprint arXiv:1703.00560* (2017) (cit. on p. 19).

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. 2017 (cit. on p. 31).

[42] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. “Understanding deep learning requires rethinking generalization”. In: *Proceedings of ICLR*. 2017 (cit. on p. 7).

[43] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. "Recovery guarantees for one-hidden-layer neural networks". In: *arXiv preprint arXiv:1706.03175* (2017) (cit. on p. 19).

[44] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. "Stronger Generalization Bounds for Deep Nets via a Compression Approach". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018 (cit. on p. 2).

[45] Ainesh Bakshi, Rajesh Jayaram, and David P Woodruff. "Learning Two Layer Rectified Neural Networks in Polynomial Time". In: *arXiv preprint arXiv:1811.01885* (2018) (cit. on p. 20).

[46] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018. URL: http://github.com/google/jax (cit. on p. 111).

[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on pp. 113, 114).

[48] Gamaleldin F. Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. "Large Margin Deep Networks for Classification". In: *Proceedings of NeurIPS*. 2018 (cit. on p. 7).

[49] Rong Ge, Rohith Kuditipudi, Zhize Li, and Xiang Wang. "Learning Two-layer Neural Networks with Symmetric Inputs". In: *arXiv preprint arXiv:1810.06793* (2018) (cit. on p. 19).

[50] Arthur Jacot, Franck Gabriel, and Clement Hongler. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018 (cit. on p. 18).

[51] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*. 2018, pp. 8571–8580 (cit. on p. 4).

[52] Ilja Kuzborskij and Christoph H. Lampert. "Data-Dependent Stability of Stochastic Gradient Descent". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018 (cit. on p. 8).

[53] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. "Algorithmic Regularization in Over-parameterized Matrix Sensing and Neural Networks with Quadratic Activations". In: *COLT*. 2018 (cit. on p. 19).

[54] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. "A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks". In: *Proceedings of ICLR*. 2018 (cit. on p. 7).

[55] Noam Shazeer and Mitchell Stern. "Adafactor: Adaptive learning rates with sublinear memory cost". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4596–4604 (cit. on p. 113).

[56] Santosh Vempala and John Wilmes. "Polynomial Convergence of Gradient Descent for Training One-Hidden-Layer Neural Networks". In: *arXiv preprint arXiv:1805.02677* (2018) (cit. on p. 19).

[57] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. "On the margin theory of feedforward neural networks". In: *arXiv preprint arXiv:1810.05369* (2018) (cit. on pp. 20, 21).

[58] Karim Abou-Moustafa and Csaba Szepesvári. "An Exponential Efron-Stein Inequality for L_q Stable Learning Rules". In: *Proceedings of the 30th International Conference on Algorithmic Learning Theory*. 2019 (cit. on p. 7).

[59] Zeyuan Allen-Zhu and Yuanzhi Li. "What Can ResNet Learn Efficiently, Going Beyond Kernels?" In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019 (cit. on pp. 18, 20).

[60] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. "Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers". In: *NeurIPS*. Full version available at http://arxiv.org/abs/1811.04918. 2019 (cit. on p. 27).

[61] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. "A Convergence Theory for Deep Learning via Over-Parameterization". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019 (cit. on p. 2).

[62] Peter Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. "Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks". In: *Journal of Machine Learning Research* 20 (2019), pp. 1–17 (cit. on p. 7).

[63] Olivier Bousquet, Yegor Klochkov, and Nikita Zhivotovskiy. "Sharper bounds for uniformly stable algorithms". In: *arXiv preprint arXiv:1910.07833* (2019) (cit. on p. 11).

[64] Lenaic Chizat, Edouard Oyallon, and Francis Bach. "On Lazy Training in Differentiable Programming". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019 (cit. on p. 18).

[65] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. "Gradient Descent Provably Optimizes Over-parameterized Neural Networks". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=S1eK3i09YQ (cit. on p. 18).

[66] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. "Gradient Descent Finds Global Minima of Deep Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019 (cit. on p. 2).

[67] Vitaly Feldman and Jan Vondrak. "High probability generalization bounds for uniformly stable algorithms with nearly optimal rate". In: *Proceedings of COLT*. 2019 (cit. on pp. 8, 11, 62).

[68] Dylan J. Foster, Spencer Greenberg, Satyen Kale, Haipeng Luo, Mehryar Mohri, and Karthik Sridharan. "Hypothesis Set Stability and Generalization". In: *Proceedings of NeurIPS 2019*. 2019, pp. 6726–6736 (cit. on p. 8).

[69] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. "Limitations of Lazy Training of Two-layers Neural Network". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019 (cit. on p. 18).

[70] Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. "Efficient training of BERT by progressively stacking". In: *International conference on machine learning*. PMLR. 2019, pp. 2337–2346 (cit. on pp. 47, 49, 50, 58).

[71] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. "Fisher-Rao Metric, Geometry, and Complexity of Neural Networks". In: *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*. 2019 (cit. on p. 7).

[72] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7 (cit. on pp. 113, 114).

[73] Vaishnavh Nagarajan and J. Zico Kolter. "Uniform convergence may be unable to explain generalization in deep learning". In: *Advances in Neural Information Processing Systems*. 2019 (cit. on p. 2).

[74] Vaishnavh Nagarajan and J. Zico Kolter. "Uniform convergence may be unable to explain generalization in deep learning". In: *Proceedings of NeurIPS*. 2019 (cit. on p. 7).

[75] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. "The role of over-parametrization in generalization of neural networks". In: *International Conference on Learning Representations*. 2019 (cit. on p. 2).

[76] Samet Oymak and Mahdi Soltanolkotabi. "Towards moderate overparameterization: global convergence guarantees for training shallow neural networks". In: *arXiv preprint arXiv:1902.04674* (2019) (cit. on p. 20).

[77] Alexander A. Sherstov and Pei Wu. "Near-Optimal Lower Bounds on the Threshold Degree and Sign-Rank of AC0". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 401–412. ISBN: 9781450367059. DOI: 10.1145/3313276.3316408. URL: https://doi.org/10.1145/3313276.3316408 (cit. on p. 21).

[78] Taiji Suzuki. "Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=H1ebTsActm (cit. on p. 20).

[79] Taiji Suzuki and Atsushi Nitanda. "Deep learning is adaptive to intrinsic dimensionality of model smoothness in anisotropic Besov space". In: (2019). URL: https://arxiv.org/abs/1910.12799 (cit. on p. 20).

[80] Gilad Yehudai and Ohad Shamir. "On the power and limitations of random features for understanding neural networks". In: *arXiv preprint arXiv:1904.00687* (2019) (cit. on pp. 20, 21).

[81] Bing Yu, Junzhao Zhang, and Zhanxing Zhu. "On the Learning Dynamics of Two-layer Nonlinear Convolutional Neural Networks". In: *CoRR* abs/1905.10157 (2019). URL: http://arxiv.org/abs/1905.10157 (cit. on p. 22).

[82] Difan Zou and Quanquan Gu. "An Improved Analysis of Training Over-parameterized Deep Neural Networks". In: *Advances in Neural Information Processing Systems*. 2019 (cit. on p. 2).

[83] Zeyuan Allen-Zhu and Yuanzhi Li. "Backward Feature Correction: How Deep Learning Performs Deep Learning". In: *arXiv preprint arXiv:2001.04413* (2020) (cit. on pp. 18, 20, 21, 111).

[84] Zeyuan Allen-Zhu and Yuanzhi Li. "Feature Purification: How Adversarial Training Performs Robust Deep Learning". In: *arXiv preprint arXiv:2005.10190* (2020) (cit. on p. 22).

[85] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. 2020 (cit. on p. 31).

[86] Minshuo Chen, Yu Bai, Jason D Lee, Tuo Zhao, Huan Wang, Caiming Xiong, and Richard Socher. "Towards Understanding Hierarchical Learning: Benefits of Neural Representations". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 22134–22145 (cit. on p. 18).

[87] Amit Daniely and Eran Malach. "Learning Parities with Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 20356–20365 (cit. on pp. 18, 20, 21).

[88] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. "Linear Mode Connectivity and The Lottery Ticket Hypothesis". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020 (cit. on p. 15).

[89] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. "Disentangling feature and lazy training in deep neural networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2020.11 (2020) (cit. on p. 18).

[90] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. "When Do Neural Networks Outperform Kernel Methods?" In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 14820–14830 (cit. on pp. 18, 20, 21).

[91] Rémi Gribonval, Gitta Kutyniok, Morten Nielsen, and Felix Voigtlaender. "Approximation spaces of deep neural networks". In: (2020). URL: https://arxiv.org/abs/1905.01208 (cit. on p. 20).

[92] Ziwei Ji and Matus Telgarsky. "Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow ReLU networks". In: *International Conference on Learning Representations*. 2020 (cit. on p. 2).

[93] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. "Fantastic Generalization Measures and Where to Find Them". In: *Proceedings of ICLR*. 2020 (cit. on p. 7).

[94] Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. "Fantastic Generalization Measures and Where to Find Them". In: *International Conference on Learning Representations*. 2020 (cit. on p. 2).

[95] Pritish Kamath, Omar Montasser, and Nathan Srebro. "Approximate is Good Enough: Probabilistic Variants of Dimensional and Margin Complexity". In: *Proceedings of Thirty Third Conference on Learning Theory*. Vol. 125. Proceedings of Machine Learning Research. 2020, pp. 2236–2262 (cit. on p. 18).

[96] Stefani Karp, Behnam Neyshabur, and Mehryar Mohri. "On the Algorithmic Stability of SGD in Deep Learning". In: (2020) (cit. on p. 7).

[97] Aitor Lewkowycz and Guy Gur-Ari. "On the training dynamics of deep networks with L_2 regularization". In: *arXiv preprint arXiv:2006.08643* (2020). URL: https://github.com/google/autol2 (cit. on p. 111).

[98] Yuanzhi Li and Zehao Dou. "When can Wasserstein GANs minimize Wasserstein Distance?" In: *arXiv preprint arXiv:2003.04033* (2020) (cit. on p. 20).

[99] Yuanzhi Li, Tengyu Ma, and Hongyang R Zhang. "Learning over-parametrized two-layer relu neural networks beyond ntk". In: *arXiv preprint arXiv:2007.04596* (2020) (cit. on p. 20).

[100] Yuanzhi Li, Tengyu Ma, and Hongyang R. Zhang. "Learning Over-Parametrized Two-Layer Neural Networks beyond NTK". In: *Proceedings of Thirty Third Conference on Learning Theory*. Vol. 125. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2613–2682 (cit. on p. 18).

[101] Philip M. Long and Hanie Sedghi. "Generalization Bounds for Deep Convolutional Neural Networks". In: *Proceedings of ICLR*. 2020 (cit. on p. 7).

[102] Philip M. Long and Hanie Sedghi. "Generalization bounds for deep convolutional neural networks". In: *International Conference on Learning Representations*. 2020 (cit. on p. 2).

[103] Edward Moroshko, Blake E Woodworth, Suriya Gunasekar, Jason D Lee, Nati Srebro, and Daniel Soudry. "Implicit Bias in Deep Linear Classification: Initialization Scale vs Training Accuracy". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 22182–22193 (cit. on p. 18).

[104] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. "Neural Tangents: Fast and Easy Infinite Neural Networks in Python". In: *International Conference on Learning Representations*. 2020. URL: https://github.com/google/neural-tangents (cit. on p. 111).

[105] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html (cit. on p. 113).

[106] Taiji Suzuki and Shunta Akiyama. *Benefit of deep learning with non-convex noisy gradient descent: Provable excess risk bound and superiority to kernel methods*. 2020. URL: https://arxiv.org/abs/2012.03224 (cit. on p. 20).

[107] Xiang Wang, Chenwei Wu, Jason D Lee, Tengyu Ma, and Rong Ge. "Beyond Lazy Training for Over-parameterized Tensor Decomposition". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. 2020, pp. 21934–21944 (cit. on p. 18).

[108] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. "Kernel and Rich Regimes in Over-parametrized Models". In: *Proceedings of Thirty Third Conference on Learning Theory*. Vol. 125. 2020, pp. 3635–3673 (cit. on p. 18).

[109] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. "Gradient descent optimizes over-parameterized deep ReLU networks". In: *Machine Learning*. Vol. 109. 2020 (cit. on p. 2).

[110] Zixiang Chen, Yuan Cao, Difan Zou, and Quanquan Gu. "How Much Over-parameterization Is Sufficient to Learn Deep Re{LU} Networks?" In: *International Conference on Learning Representations*. 2021 (cit. on p. 2).

[111] Carles Domingo-Enrich, Alberto Bietti, Eric Vanden-Eijnden, and Joan Bruna. "On Energy-Based Models with Overparametrized Shallow Neural Networks". In: *Proceedings of ICML*. 2021 (cit. on p. 18).

[112] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. "A Mathematical Framework for Transformer Circuits". In: *Transformer Circuits Thread* (2021). https://transformer-circuits.pub/2021/framework/index.h (cit. on p. 31).

[113] Stefani Karp, Ezra Winston, Yuanzhi Li, and Aarti Singh. "Local Signal Adaptivity: Provable Feature Learning in Neural Networks Beyond Kernels". In: *Advances in Neural Information Processing Systems*. 2021 (cit. on p. 18).

[114] Eran Malach, Pritish Kamath, Emmanuel Abbe, and Nathan Srebro. "Quantifying the Benefit of Using Differentiable Learning over Tangent Kernels". In: *Proceedings of ICML*. 2021 (cit. on p. 18).

[115] Maria Refinetti, Sebastian Goldt, Florent Krzakala, and Lenka Zdeborová. "Classifying high-dimensional Gaussian mixtures: Where kernel methods fail and neural networks succeed". In: (2021). URL: https://arxiv.org/abs/2102.11742 (cit. on pp. 18, 20, 21).

[116] Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. "bert2BERT: Towards Reusable Pretrained Language Models". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 2022 (cit. on p. 57).

[117] Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Fabian Pedregosa, and Max Vladymyrov. "GradMax: Growing Neural Networks using Gradient Information". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=qjN4h_wwUO (cit. on p. 57).

[118] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. "What Can Transformers Learn In-Context? A Case Study of Simple Function Classes". In: *Neural Information Processing Systems*. 2022 (cit. on p. 31).

[119] Samy Jelassi, Michael Sander, and Yuanzhi Li. "Vision Transformers provably learn spatial structure". In: *Advances in Neural Information Processing Systems*. 2022 (cit. on p. 31).

[120] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. "In-context Learning and Induction Heads". In: *Transformer Circuits Thread* (2022). https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html (cit. on p. 31).

[121] Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. "Staged training for transformer language models". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 19893–19908 (cit. on p. 57).

[122] Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, et al. "Ul2: Unifying language learning paradigms". In: *The Eleventh International Conference on Learning Representations*. 2022 (cit. on pp. 56, 113).

[123] Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. "Learning to Grow Pretrained Models for Efficient Transformer Training". In: *The Eleventh International Conference on Learning Representations*. 2022 (cit. on pp. 47, 49).

[124] Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. "Transformers learn to implement preconditioned gradient descent for in-context learning". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cit. on p. 31).

[125] Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. "Birth of a Transformer: A Memory Viewpoint". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023 (cit. on p. 31).

[126] Stefani Karp, Pranjal Awasthi, and Satyen Kale. "Provable Gradient-Descent-Based Learning of Decision Lists by Transformers". In: *DeepMath*. 2023 (cit. on p. 31).

[127] Xiang Li, Yiqun Yao, Xin Jiang, Xuezhi Fang, Xuying Meng, Siqi Fan, Peng Han, Jing Li, Li Du, Bowen Qin, et al. "Flm-101b: An open llm and how to train it with $100 k budget". In: *arXiv preprint arXiv:2309.03852* (2023) (cit. on pp. 47, 57).

[128] Yuchen Li, Yuanzhi Li, and Andrej Risteski. "How do transformers learn topic structure: towards a mechanistic understanding". In: *Proceedings of the 40th International Conference on Machine Learning*. 2023 (cit. on p. 31).

[129] Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. "Transformers Learn In-Context by Gradient Descent". In: *International Conference on Machine Learning*. 2023 (cit. on p. 31).

[130] Sashank Reddi, Sobhan Miryoosefi, Stefani Karp, Shankar Krishnan, Satyen Kale, Seungyeon Kim, and Sanjiv Kumar. "Efficient Training of Language Models using Few-Shot Learning". In: *Proceedings of the 40th International Conference on Machine Learning*. 2023 (cit. on pp. 47, 49, 50, 55–58, 114, 115).

[131] Yuandong Tian, Yiping Wang, Beidi Chen, and Simon Shaolei Du. "Scan and Snap: Understanding Training Dynamics and Token Composition in 1-layer Transformer". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cit. on p. 31).

[132] Deqing Fu, Tianqi CHEN, Robin Jia, and Vatsal Sharan. *Transformers Learn Higher-Order Optimization Methods for In-Context Learning: A Study with Linear Models*. 2024. URL: https://openreview.net/forum?id=YKzGrt3m2g (cit. on p. 31).

[133] Stefani Karp, Nikunj Saunshi, Sobhan Miryoosefi, Sashank J. Reddi, and Sanjiv Kumar. "Landscape-Aware Growing: The Power of a Little LAG". In: *arXiv preprint arXiv:2406.02469* (2024) (cit. on p. 47).

[134] Yite Wang, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. "LEMON: Lossless model expansion". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=3Vw7DQqq7U (cit. on pp. 57, 58).

[135] Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. "Masked Structural Growth for 2x Faster Language Model Pre-training". In: *The Twelfth International Conference on Learning Representations*. 2024 (cit. on p. 57).

[136] Wikimedia Foundation. *Wikimedia Downloads*. URL: https://dumps.wikimedia.org (cit. on p. 113).