Towards an Empirical Theory of Deep Learning

A DISSERTATION PRESENTED

BY

PREETUM NAKKIRAN

TO

THE SCHOOL OF ENGINEERING AND APPLIED SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SUBJECT OF

COMPUTER SCIENCE

HARVARD UNIVERSITY

CAMBRIDGE, MASSACHUSETTS

JULY 2021

Thesis advisors: Professors Boaz Barak and Madhu Sudan Preetum Nakkiran

Towards an Empirical Theory of Deep Learning

Abstract

In this thesis, we take an empirical approach to the theory of deep learning. We treat deep learning systems as black boxes, with inputs we can control (train samples, architecture, model size, optimizer, etc.) and outputs we can observe (the neural network function, its test error, its parameters, etc.). Our goal is to characterize how our choice of inputs affects the outputs. As an empirical theory, we aim to describe this behavior quantitatively, if not prove it rigorously. We hope for theories that are as general and universal as possible, applying in a wide range of deep learning settings, including those in practice.

We present three empirical theories towards this goal. (1) Deep Double Descent demonstrates that the relationship between inputs and outputs in deep learning is not always monotonic in natural ways: there is a predictable "critical regime" where, for example, training on more data can actually hurt performance, but models are well-behaved outside this regime. (2) The Deep Bootstrap Framework shows that to understand the *generalization* of the output network, it is sufficient to understand *optimization* aspects of our input choices. (3) Distributional Generalization takes a closer look at the output network, and finds that trained models actually "generalize" in a much broader sense than we classically expect. We introduce a new kind of generalization to capture these behaviors.

Our results shed light on existing topics in learning theory (especially generalization, overparameterization, interpolation), and also reveal new phenomena which require new frameworks to capture. In some cases, our study of deep learning has exposed phenomena that hold even for non-deep methods. We thus hope the results of this thesis will eventually weave into a general theory of learning, deep and otherwise.

Contents

1 Introduction 1
1.1 What is Deep Learning? 2
1.2 Guiding Questions 3
1.2.1 Objects of Study 4
1.3 Our Approach 5
1.3.1 Fully Rigorous Theory 6
1.3.2 Obstacles to a Fully Rigorous Theory 6
1.3.3 The Way Forward 8
1.4 Our Results 9
1.4.1 Overparameterization and Interpolation 10
1.5 Organization 11
1.5.1 Relations to Published Work 11

2 Background 12

3 Summary of Results 16
3.1 Deep Double Descent 17
3.2 The Deep Bootstrap Framework 18
3.3 Distributional Generalization 21

4 Deep Double Descent 24
4.1 Introduction 25
4.2 Our results 27
4.3 Related work 30
4.4 Experimental Setup 31
4.5 Model-wise Double Descent 32
4.6 Epoch-wise Double Descent 35
4.7 Sample-wise Non-monotonicity 37
4.8 Conclusion and Discussion 39

5 The Deep Bootstrap Framework 41
5.1 Introduction 43
5.2 The Deep Boostrap 46

5.3 Experimental Setup 48
5.3.1 Datasets 49
5.3.2 Methodology 49
5.4 Main Experiments 50
5.5 Deep Phenomena through the Bootstrap Lens 52
5.6 Differences between the Worlds 56
5.7 Conclusion and Discussion 57

6 Distributional Generalization 58
6.1 Introduction 60
6.1.1 Distributional Generalization 62
6.1.2 Summary of Contributions 64
6.1.3 Related Work and Significance 65
6.2 Preliminaries 67
6.2.1 Distributional Closeness 68
6.2.2 Framework for Indistinguishability 69
6.3 Feature Calibration 70
6.3.1 Experiments 73
6.3.2 Discussion 78
6.3.3 1-Nearest-Neighbors Connection 79
6.3.4 Pointwise Density Estimation 81
6.4 Agreement Property 81
6.4.1 Experiments 83
6.4.2 Potential Mechanisms 84
Bimodal Samples 84
Pointwise Agreement 85
6.5 Limitations and Ensembles 86
6.5.1 Ensembles 86
6.6 Distributional Generalization: Beyond Interpolating Methods 87
6.7 Conclusion and Discussion 90
6.7.1 Open Questions 91

7 Conclusion 93
7.1 Open Questions 94

Appendix A Additional Works 97
A.1 On Adversarial Examples 97
A.2 On Dynamics of SGD and Representation Learning 97
A.3 On Theoretical Models 98

Appendix B Deep Double Descent 99

B.1 Experimental Details . . . 99
B.1.1 Models . . . 99
B.1.2 Image Classification: Experimental Setup . . . 100
B.1.3 Neural Machine Translation: Experimental Setup . . . 102
B.1.4 Per-section Experimental Details . . . 103
B.2 Extended discussion of related work . . . 104
B.3 Random Features: A Case Study . . . 106
B.4 Appendix: Additional Experiments . . . 107
B.4.1 Epoch-wise Double Descent: Additional results . . . 107
B.4.2 Model-Wise Double Descent: Additional Results . . . 109
Clean Settings With Model-wise Double Descent . . . 109
Weight Decay . . . 110
Early Stopping does not exhibit double descent . . . 112
Training Procedure . . . 113
B.4.3 Ensembling . . . 114

Appendix C The Deep Bootstrap Framework 118
C.1 Toy Example . . . 118
C.2 Additional Figures . . . 120
C.2.1 Introduction Experiment . . . 120
C.2.2 DARTS Architectures . . . 120
C.2.3 Effect of Data Augmentation . . . 120
C.2.4 Adam . . . 122
C.2.5 Pretraining . . . 122
Pretrained MLP . . . 122
Pretrained Vision Transformer . . . 123
C.2.6 Learning Rate . . . 124
C.2.7 Difference Between Worlds . . . 126
C.2.8 Error vs. SoftError . . . 126
Training with MSE . . . 126
C.3 Bootstrap Connection . . . 128
C.4 Appendix: Experimental Details . . . 129
C.4.1 Introduction Experiment . . . 129
C.4.2 Main Experiments . . . 130
C.4.3 Implicit Bias . . . 131
C.4.4 Image-GPT Finetuning . . . 131
C.5 Appendix: Datasets . . . 132
C.5.1 CIFAR-5m . . . 132
C.5.2 Imagenet: DogBird . . . 135

Appendix D Distributional Generalization 137

D.1 Student-Teacher Indistinguishability 137
D.2 Experimental Details 140
D.2.1 Datasets 140
D.2.2 Models 140
Decision Trees 140
Kernels 140
Neural Networks 142
D.3 Feature Calibration: Appendix 143
D.3.1 A guide to reading the plots 143
D.3.2 Experiment 1 144
D.3.3 Constant Partition 145
D.3.4 Class Partition 146
Neural Networks and CIFAR-10 146
Decision Trees 147
D.3.5 Multiple Features 147
D.3.6 Coarse Partition 148
D.3.7 Pointwise Density Estimation 150
Experiment: Train-set Ensemble 151
D.4 Agreement Property: Appendix 153
D.4.1 Experimental Details 153
D.4.2 Additional Plots 154
D.4.3 Alternate Mechanisms 155
Bimodal Samples 155
Pointwise Agreement 156
D.5 Non-interpolating Classifiers: Appendix 158
D.6 Nearest-Neighbor Proofs 160
D.6.1 Agreement Property 160

References 179

Acknowledgments

This thesis, and my experience in grad school, would not have been the same without the support, guidance, lessons, collaborations, friendship and company of many people.

First, I want to thank my advisors: Boaz Barak and Mahdu Sudan. I did not start grad school in machine learning— I started in the Theory Group, advised by Madhu. I am grateful for Madhu in this time for allowing me to explore many exciting areas of theory, teaching me many things, and helping refine my taste in research. I learnt from Madhu to think deeply about the motivations for research questions, and to be ambitious in defining new paradigms. But I am especially grateful to Madhu for supporting me as my interests branched beyond the traditional realm of Theory: for being patient with me as I explored my interests, and for encouraging me when I started more seriously thinking about machine learning. Around this time I ran into Boaz in Hi-Rise Coffee, and was very happy to see another theorist excited about (among other things) understanding deep learning. This gave me confidence to continue pursing this direction, at a time when not many TCS folks were thinking about deep learning. Boaz's intense support in both organization and research were crucial to spinning up the Harvard ML Theory Group: from the reading group (which evolved into a seminar series), to the research meetings, to the compute resources required. In research, I am very glad for all the energy and insight that comes out of any meeting with Boaz— this is the highlight of every project. Finally, this goes without saying, but even outside of research, Boaz and Madhu have been extremely warm and supportive people.

Research is a social endeavor, and I want to thank everyone at Harvard who made this such an enjoyable and productive experience. Especially those involved in the early days of the ML Theory Group, when we were starting from scratch, learning the field, and developing our research agendas: Gal Kaplun, Yamini Bansal, Dimitris Kalimeris, Nikhil Vyas, Ben Edelman, Fred Zhang, Kelly Zhang. It has been a lot of fun learning from, writing papers, and discussing with all of you. I want to thank everyone from the Harvard Theory group, and also other friends around Cambridge, for making my time in Cambridge so enjoyable, both inside and outside the office. Including: Elad Haramaty, Julien Fageot, Boriana Gjura, Meena Jagadeesan, Aditya Vignesh Venkatramani, Chenyang Yuan. I especially thank Jarosław Błasiok and Thibaut Horel, for teaching me so much during my TCS days, and for keeping me honest during my ML days.

I want to thank all of my teachers, at Harvard, Berkeley, and earlier. Many of these ideas were later useful in my research, in unexpected ways. Including: Salil Vadhan, Jelani Nelson, Scott Kominers, Luca Trevisan, Lap Chi Lau, Prasad Raghavendra, Sangam Garg, Anant Sahai, Babak Ayazifar, Laurent El Ghaoui, Paul Vojta, Matthias Reinsch, Satish Rao, Peter Saxby, John Frank.

I want to thank all of my collaborators. And especially senior collaborators and mentors, who were very generous with their time when I was getting started in machine learning, including: Ilya Sutskever, Chris Olah, Jacob Steinhardt, Tengyu Ma, Sham Kakade, Raziel Alvarez. I am grateful that they treated my ideas seriously even when I had no prior experience. I also thank my colleagues at OpenAI during the summer of 2019, which was a formative time for my research. Including: Aditya Ramesh, Mark Chen, Johannes Otterbach, Rewon Child.

Finally, I thank my parents, for everything they have done in their lives to support me. I am deeply grateful for their love and support.

Mathematical theory is not critical to the development of machine learning. But scientific inquiry is.

Leo Breiman, 1995.

1 Introduction

The motivation for this thesis is to develop a theory of deep learning. First, we discuss the definition of deep learning, and why we need a theory. We then describe obstacles to a fully rigorous mathematical theory, and outline our empirical approach.

Briefly, we will take an "empirical science" approach to theory, which is closest to the approach taken in physics. We will treat deep learning as an empirical phenomena– an aspect of Nature that we can observe and experiment with, and whose behavior we want to characterize and understand. We will try to identify universal phenomena in deep learning in the real world (i.e. in practice). We will try to capture these phenomena in conjectures as formally as possible, and then test these conjectures through experiment. In some cases, our conjectures suggest entirely new phenomena, which we then validate in the real world.

1.1 What is Deep Learning?

There is no formal definition of deep learning (DL) which captures all of its current and potential future instantiations. What the Turing machine did for defining computation, we do not yet have for defining deep learning— and this is one goal of theory. But informally, "deep learning" refers not to a single method, but to a collection of *ingredients* which can be combined in various ways to produce learning systems for certain classes of learning problems. These ingredients typically include certain parametric function families ("neural network architectures"), optimization algorithms (typically Stochastic Gradient Descent and variants), objective functions, and datasets (typically high-dimensional "natural" distributions, such as images or language). We cannot formally specify which sets of ingredients are allowed, which ways they can be combined, or which classes of problems they can solve. However, in practice we have a growing set of "recipes," which are successful ways of combining ingredients that yield desirable behavior in the real world.

This paradigm of deep learning has been enormously successful in practice: it has addressed long-standing learning problems which posed obstacles to prior methods (e.g. in image classification Krizhevsky, Sutskever, and Hinton (2012)), and it has made progress on entirely new types of problems, which we were not even attempting to solve before. Its success is now fairly robust: it is clear that DL is not simply an "accidental" success in a few settings, but rather a versatile toolbox that has hope of solving many real-world learning problems, given enough data and computational resources.

The Need for Theory Despite the tremendous empirical success of deep learning, we are very far from a scientific understanding of the field. The state-of-the-art in deep learning steadily advances, but in unpredictable ways: it is unclear if a new advance will be due to a new architecture, dataset, optimizer, training methodology, etc. And we continue to discover new capabilities and application domains of deep learning— we do not yet know which problems deep learning can or cannot solve. Practitioners and theorists alike are regularly *surprised* by advances in deep learning: cases where changing the *ingredients* lead to unexpected gains in the final system. One goal of theory in deep learning is to eliminate such surprises from the practice of deep learning. In doing so, we expect to gain a more thorough understanding of the nature of computational learning. This motivates our guiding question below.

1.2 Guiding Questions

In developing a theory, we must decide: what is the goal of the theory? What questions should it answer? In this thesis, the following question will guide our work:

“How does what we *do* affect what we *get*?”

“Doing” deep learning involves many moving parts– choices of architecture, dataset, sample size, initialization, loss function, pretraining, etc. At the end, we “get” a trained model, which we can study under various lenses (its test error, its parameters, its robustness, etc). All of the different inputs affect the model produced, but in highly coupled ways, which we do not adequately understand. This is not only a matter of theoretical discomfort: Practitioners are regularly surprised when jointly changing the inputs to deep learning systems lead to unexpected capabilities (e.g. ViT Dosovitskiy, Beyer, Kolesnikov, Weissenborn, Zhai, Unterthiner, Dehghani, Minderer, Heigold, Gelly, Uszkoreit, and Houlsby (2021), GPT-3 Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, Neelakantan, Shyam, Sastry, Askell, Agarwal, Herbert-Voss, Krueger, Henighan, Child, Ramesh, Ziegler, Wu, Winter, Hesse, Chen, Sigler, Litwin, Gray, Chess, Clark, Berner, McCandlish, Radford, Sutskever, and Amodei (2020a), CLIP Radford, Kim, Hallacy, Ramesh, Goh, Agarwal, Sastry, Askell, Mishkin, Clark, Krueger, and Sutskever (2021), AlphaFold2 Jumper, Evans, Pritzel, Green, Figurnov, Ronneberger, Tunyasuvunakool, Bates, Žídek, Potapenko, et al. (2021)).

Existing frameworks from learning theory are essentially attempts to answer the above question in certain settings. For example, consider the "uniform convergence" framework for supervised learning. If we learn with a hypothesis class that satisfies uniform convergence of its error, then we are guaranteed that: no matter what we do, if we output a model with small train error, then it will also have small test error. In this sense, what we *do* on the train set is exactly what we *get* on the test set (w.r.t. error).

Unfortunately, this particular framework does not always apply for deep learning– but the situation is subtle. There are settings in deep learning which are well-captured by "uniform convergence" (e.g. when data is large and models are small). But there are closely related settings where uniform convergence fails severely (e.g. with small data or large models). For example, we can train large *over-parameterized* models which have essentially arbitrary behavior on their train sets, regardless of their test performance. A large network with 0% error on its train set could have anywhere between 0-100% error at test time, depending on what we *do*. This example reflects the tantalizing state of DL theory: existing theories of learning often capture deep learning behaviors in *some* regimes, but fail in others. This motivates the search for better frameworks to reason about deep learning in a unified way[*].

Finally, deep learning is especially interesting because we often *get* much more than we asked for. We simply ask for an empirical-risk-minimizer on the train set, but we get a function with structured internal representations (*representation learning*), which improves sample-complexity on related learning problems (*transfer learning*), and can be combined modularly with other neural networks, trained in different ways. We will see a new example of getting behaviors "for free" later in this thesis,

[*]The following works survey the state of the fully rigorous approach to deep learning theory: Bartlett, Montanari, and Rakhlin (2021); Berner, Grohs, Kutyniok, and Petersen (2021); Belkin (2021).

in Distributional Generalization (Chapter 6).

1.2.1 Objects of Study

For the works presented in this thesis, we will simplify the above guiding question in the following ways:

First, we will consider deep learning only in the context of *supervised classification*. This is both convenient and appropriate: supervised classification was one of the first "breakthrough applications" of deep learning, and thus methods for supervised classification are among the most robust and well-understood in practice. Moreover, supervised learning in practice is the closest setting to problems studied in theory– and so this is a natural first step to bridge theory and practice. Here, we can at least define the problem formally (if not the solution). Finally, many of the other applications or behaviors in DL reduce to supervised learning as a special case (e.g. reinforcement learning, semi-supervised learning, self-supervised learning)– and so we hope that understanding supervised learning is the key to understanding many other settings of deep learning.

Second, we will consider the models we *get* only in terms of their black-box and on-distribution behavior. Deep learning systems output *parameters* θ, which determine a function f_θ. We will only consider the input-output behavior of this function f, and not open the black-box of parameters θ. Moreover, we will not consider arbitrary inputs x, but only inputs $x \sim D$ drawn from the distribution D on which f was trained. This is in some sense the simplest non-trivial level of abstraction, but there is already much to be said about behaviors at this level.

1.3 Our Approach

To address our guiding question, the next step is to decide *what type* of theory is appropriate for deep learning. There are many types of theories in the sciences, from quantitative theories about mathematical objects (e.g. theory of computation, or of differential manifolds), to quantitative theories about the real world (often called "laws" in physics), to qualitative theories about the real world (e.g. evolution, or the central dogma in biology). We will search for a *quantitative* theory about deep learning as it is used in the real world. Many of the objects involved have quantitative descriptions— model architecture, optimizer, test error, etc— and so we hope their interactions can be described quantitatively as well. Among quantitative theories, there are fully rigorous theories, and then there are empirical theories.

1.3.1 Fully Rigorous Theory

The fully rigorous mathematical theory is the "gold standard" of theory: a theorem with assumptions which hold in the real world, and whose conclusions capture behaviors we care about. The field of *compressed sensing* provides an example of such a theory (Candès and Tao, 2006; Tropp and Gilbert, 2007). The theory here states that, *if* real world signals satisfy a certain mathematical property (e.g. approximate sparsity in the Haar wavelet basis), then a certain explicit mathematical procedure (ℓ_1-minimization with Gaussian measurements) will "work" in a formal sense (i.e. will recover the signal, up to quantified error bounds). The sparsity assumption in this theorem can in principle be verified, by making sufficient observations about the real world.

It is tempting to pursue a fully rigorous theory of deep learning. However, this approach quickly runs into obstacles.

1.3.2 Obstacles to a Fully Rigorous Theory

The primary obstacle to a fully rigorous theory is definitional: we cannot yet formally define the methods we use, nor formally define the problems we expect them to solve. It is not even a matter of proofs being too difficult— we are not even able to formally state conjectures which capture real world behaviors.

It is instructive to explicitly encounter these obstacles as we try to formalize a candidate theorem. For example, we could hope to at least describe the existing practical success of deep learning (and perhaps even predict its future successes), via a theorem of the following form:

"Deep neural nets work on natural distributions, when trained on enough data"

To do this formally, we must formally define each object. What do we mean by "deep neural nets"? We mean some family of architectures (MLPs, CNNs, etc) coupled with some family of optimizers (SGD, Adam, etc). Certainly not all architectures can be allowed: Too restrictive a family of architectures is not interesting (we do not want a theory of one particular neural network; we want a general theory of neural networks). And too general a family is also not interesting: Stochastic Gradient Descent on a sufficiently contrived architecture can simulate an arbitrary circuit (Abbe and Sandon, 2020). There is no existing definition of the set of "reasonable architectures"— which should include all the ones that are successful in practice now, and which could be successful in the future.

We encounter similar problems in specifying each object (the set of allowable optimizers, activation functions, loss functions, etc). There are many moving parts in deep learning, and learning systems are robust to many choices of these parts, but we cannot precisely specify which choices are allowed. Moreover, we may not be working at the right level of abstraction: what we think of as "deep learning" today may be a subset of some much larger unifying object which we have not yet discovered (e.g. "locally-optimizable online learning systems").

Next, what do we mean by "natural distributions"? Deep learning is not an appropriate tool for *every* learning problem. Non-deep methods are still used in many places in practice, and theory gives us explicit examples of problems which are "hard" for deep learning but easy for other methods (Shalev-Shwartz, Shamir, and Shammah, 2017). Yet, there is a certain family of problems for which deep methods tend to be more successful than other methods. These often share characteristics such as: high-dimensional inputs, no semantic input coordinates, poorly understood generative processes, and large datasets. Representative examples are many vision problems (image classification, segmentation, generation, etc.) and certain natural language processing (NLP) problems (Krizhevsky et al., 2012; He, Zhang, Ren, and Sun, 2016a; Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio, 2014; Devlin, Chang, Lee, and Toutanova, 2018; Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin, 2017b). But we have no explicit definition of a set of "natural distributions" for which deep learning works. Such a definition, to be relevant, must include both certain image distributions and certain natural-language distributions— two a priori very different kinds of inputs— which highlights the difficulty in definition.

However, just because we cannot yet formally define these objects does not mean they do not exist. The state of practice strongly suggests that some version of the statement "deep learning works" is true, with an appropriate choice of definitions. We have a certain intuitive sense of what a "reasonable architecture" on a "natural distribution" means— and we can allow ourselves to use these informal terms in our conjectures now, in the hopes that they can be formalized in the future.

1.3.3 The Way Forward

In light of the above obstacles, there are essentially two ways to proceed:

1. "Bottom up": We start with a simple mathematical theory which we fully understand (but is unrealistic), and gradually build up complexity to better approximate the real world. This approach is being pursued by many scientists, and has lead to richer theories of regression, interpolation, kernel methods, non-convex optimization, infinite-width limits, etc. However, such theories are always at risk of describing objects that are detached from the deep learning

in the real world— and may or may not have similar behaviors. In fact, we will see several examples where this approach to theory has suggested incorrect intuitions about the real world.

2. "Top Down": We start with the real world, and gradually abstract away pieces of it which we can describe (if not prove) mathematically. This is the empirical theory approach: we attempt to *quantitatively describe* behaviors in the real world, as formally and generally as possible, even if we cannot derive them from first-principles. This approach is underutilized in machine learning, but has a long and fruitful history in physics, where it often leads to "laws" (Kepler's laws, Ampère's law, Boyle's law, etc). Such examples in physics abound, where behaviors that are universal and empirically characterized often lead to deeper understanding, and eventually to unified mechanistic theories.

The former approach lacks realism; the latter lacks mathematical justification. We chose to do the latter, because (1) We are primarily interested in the real world, and (to us) formally describing it is almost as good as having proofs, (2) It is an underutilized approach in the current scientific community, and (3) In our experience, models for deep learning constructed from the "bottom up" approach rarely hold beyond the specific phenomena they were designed to explain. However, models constructed from the "top down" approach often hold far more generally, and can suggest *new* deep learning phenomena beyond what they were designed to capture.

But it is always useful to have an eye for "both ends", in order to eventually join them.

1.4 Our Results

Informally, the three works presented in this thesis address our guiding question by showing:

1. The relationship between what we *do* and what we *get* is not always a continuous and well-behaved one. There is "critical regime" where models become very sensitive to their inputs,

and behave poorly— but outside this regime, behavior is monotonic and predictable (Deep Double Descent, Nakkiran, Kaplun, Bansal, Yang, Barak, and Sutskever (2020)).

2. With respect to test error, what we *do* matters essentially only through two factors: its online optimization speed (when training on an infinite stream of fresh samples) and its offline optimization speed (when training to fit a fixed train set). This reduces understanding generalization to understanding certain aspects of optimization (Deep Bootstrap Framework, Nakkiran, Neyshabur, and Sedghi (2021a)).

3. Although what we *do* is designed to minimize a single metric (test error), the models we *get* have much richer structure beyond just their test error. Our work reveals, for example, that even models with high test error can have certain "good behavior" at a coarser scale. These behaviors are captured by a new, broader, definition of generalization (Distributional Generalization, Nakkiran and Bansal (2020)).

1.4.1 Overparameterization and Interpolation

Much of our work will focus on understanding *interpolating* classifiers– that is, classifiers which fit their train sets exactly (with 0% train error). This setting presents perhaps the largest gap in our knowledge, since it is in conflict with the spirit of many existing theories of learning. Many theories (including empirical risk minimization, and kernel methods) suggest that we need some form of "capacity control" or "regularization" in order to learn. Without such explicit regularization, learning methods may "overfit" and perform poorly, especially in the presence of label noise (Belkin, Ma, and Mandal, 2018b; Bartlett et al., 2021). In contrast, deep neural nets can be trained to interpolation, without explicit regularization, and still achieve good test performance (Zhang, Bengio, Hardt, Recht, and Vinyals, 2017b; Belkin, Hsu, Ma, and Mandal, 2019a). The issues of overparameterization and interpolation are related, since interpolation is often only possible with large overparameterized networks.

This disconnect motivates a rich body of work on overparameterized and interpolating methods, including the "implicit bias" and "benign overfitting" programs (Zhang et al., 2017b; Belkin, Hsu, and Mitra, 2018a; Belkin et al., 2018b, 2019a; Liang and Rakhlin, 2018; Hastie, Montanari, Rosset, and Tibshirani, 2019; Schapire, Freund, Bartlett, Lee, et al., 1998; Breiman, 1995; Mei and Montanari, 2019; Gunasekar, Lee, Soudry, and Srebro, 2018a; Soudry, Hoffer, Nacson, and Srebro, 2018; Bartlett, Long, Lugosi, and Tsigler, 2020). It motivates us as well. In this context, the works in this thesis will:

1. Highlight "pathologies" of models when passing between underparameterized and overparameterized regimes (Deep Double Descent, Nakkiran et al. (2020)).

2. Demonstrate a close connection between underparameterized and overparameterized regimes (Deep Bootstrap Framework, Nakkiran et al. (2021a)). In a certain sense, models which are overparameterized (fitting their finite train sets) "behave as though" they were underparameterized (trained on effectively infinite data). This converts questions of "implicit bias" into questions about *explicit* properties of optimization.

3. Demonstrate a significant *difference* between underparameterized and overparameterized regimes (Distributional Generalization, Nakkiran and Bansal (2020)). Along the way, we will show realistic settings where interpolation is *not* "benign," but rather "malignant" in predictable ways. In particular, our conjectures imply that interpolating neural networks are *not consistent* in settings with nonzero Bayes risk.

1.5 Organization

In the remainder of this thesis, we will focus only on the three representative papers above.

We will first describe necessary background and notation (Chapter 2). Then we briefly summarize the contributions of these papers (Chapter 3), in relation to our guiding question and areas of overparameterization & interpolation. The three subsequent chapters are based on the content of each of the papers. We conclude with reflections and open questions.

1.5.1 Relations to Published Work

Chapter 4 is based on:

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., Sutskever, I. (2020). Deep Double Descent: Where bigger models and more data hurt. *In International Conference on Learning Representations.*

Chapter 5 is based on:

Nakkiran, P., Neyshabur, B., Sedghi, H. (2021). The deep bootstrap framework: Good online learners are good offline generalizers. *In International Conference on Learning Representations.*

Chapter 6 is based on:

Nakkiran, P. & Bansal, Y. (2020). Distributional generalization: A new kind of generalization. *ArXiv, abs/2009.08092.*

2 Background

What do we *do* in practice? In this chapter we set up some formalism and describe the objects of study, which will give us context for our results.

We consider deep learning for *supervised classification*. The following description is standard and present in many textbooks, for example Goodfellow, Bengio, and Courville (2016). We are given access to samples from a distribution $\mathcal{D}$ over domain $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input-space and $\mathcal{Y}$ is label-space. The input-space will often be $\mathbb{R}^d$, and high-dimensional, while the label-space will often be finite and small. For example, we can consider the distribution $\mathcal{D}$ given by images of certain objects in certain settings, labeled by their object type (ImageNet is a set of samples from such a distribution). Our goal[*] is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ with small *test classification error*:

$$\text{TestError}(f) := \Pr_{x,y\sim\mathcal{D}}[f(x) \neq y] = \mathop{\mathbb{E}}_{x,y\sim\mathcal{D}}[1\{f(x) \neq y\}] \tag{2.1}$$

That is, small probability of misclassifying a new example from the distribution. To achieve this goal, what we often *do* is the following: First, we take N independent samples $(x_i, y_i) \sim \mathcal{D}$ (for $i = 1, \ldots, N$) as our *train set*. Then, we pick a certain parametric family of functions $\mathcal{F}$ mapping $\mathcal{X}$ to $\mathcal{Y}$. We call an element $f_\theta \in \mathcal{F}$ a "neural network" $f_\theta : \mathcal{X} \to \mathcal{Y}$, and it is specified by parameters θ ("weights"). Examples of parametric families include the family of Multi-layer Perceptrons (MLPs), CNNs (Convolutional Neural Networks), ResNets (Residual Networks), etc.

We sometimes also think of neural networks as outputting "soft decisions" – that is, distributions over $\mathcal{Y}$, which we denote by $\Delta(\mathcal{Y})$. The soft-decision classifier $\overline{f_\theta} : \mathcal{X} \to \Delta(\mathcal{Y})$, induces the hard-decision classifier $f_\theta : \mathcal{X} \to \mathcal{Y}$ via argmax. That is, $f_\theta(x) := \text{argmax}_{y\in\mathcal{Y}}[\overline{f_\theta}(x)]_y$. For such classifiers, can consider their "test soft error", which is defined as their expected probability mass on incorrect labels:

$$\text{TestSoftError}(\bar{f}) := \mathop{\mathbb{E}}_{x,y\sim\mathcal{D}}[1 - [\bar{f}(x)]_y]$$

We then attempt to find a setting of parameters θ which minimizes the train error:

$$\text{TrainError}(f_\theta) := \frac{1}{N}\sum_i 1\{f_\theta(x_i) \neq y_i\} \tag{2.2}$$

Crucially, we do this in a particular way: First, we construct a differentiable proxy of the train error, called the train *loss*:

$$\text{TrainLoss}(f_\theta) := \frac{1}{N}\sum_i \ell(y_i, \overline{f_\theta}(x_i)) \tag{2.3}$$

[*]This is the simplest goal. We could have other goals, such as learning a function that is *robust* to distribution shifts, or to adversarial perturbations. Or one that satisfies certain notions of fairness, or composes well with other downstream functions, etc.

where the inner loss function ℓ can be a number of things— most often it is cross-entropy or L2 loss (Hui and Belkin, 2021). With reasonable choice of ℓ, minimizing the train loss suffices to minimize the train error

$$\underset{\theta}{\operatorname{argmin}} \operatorname{TrainLoss}(f_\theta) \subseteq \underset{\theta}{\operatorname{argmin}} \operatorname{TrainError}(f_\theta)$$

To minimize the train loss, we start at a random[†] point in parameter space θ, and use mini-batch Stochastic Gradient Descent (SGD) or variants thereof. Specifically, at each step of SGD, we estimate gradients of the train loss as

$$\nabla_\theta \operatorname{TrainLoss}(f_\theta) \approx \frac{1}{|B|} \sum_{x_i, y_i \in B} \nabla_\theta \ell(y_i, \overline{f_\theta}(x_i))$$

using a mini-batch B of random samples from the train set. In practice, we sample without replacement until all samples are extinguished ("one epoch"), and then repeat the process for multiple epochs.

The "generalization problem" is to understand the following disconnect: why does this particular way of minimizing the train error often lead to a classifier with small *test error*?

Additional Notation

Model Size. We can crudely describe the size of models by their number of parameters: the dimension of θ. For realistic models, this measure is monotonically related to more intrinsic notions of model capacity (for example, as we propose in Nakkiran et al. (2020)).

Interpolation. A classifier is *interpolating* if it fits its train set exactly. That is, if $\operatorname{TrainError}(f) = 0$. Interpolation is related to *overparameterization*, since interpolation often requires model size to be much larger than the size of the train set.

[†]The details of how to sample an initialization are part of the specification of the neural network.

Bulk Landscape. The following object will be central to several of our works. Fix a certain class of model family $\mathcal{F}$ and distribution $\mathcal{D}$. Then, define the "bulk loss landscape" $L_{\mathcal{D},\mathcal{F}}(N, S, T)$ as: the expected test error of a model of size S, when trained on N samples from the distribution for T steps of SGD.

3 Summary of Results

We now briefly summarize our work, as progress towards answering our guiding question: *how does what we do affect what we get?* We have chosen to include only 3 representative papers in this thesis, which present 3 conceptually very different results. All of these works can be seen as claims about the structure of the map between design choices in deep learning ("what we do") and the trained model ("what we get"), as illustrated in Figure 3.1: Deep Double Descent shows that this map can be poorly behaved in a certain "critical regime". The Deep Bootstrap Framework shows that, with respect to test error, this map approximately "factorizes" into online optimization and offline optimization aspects. And Distributional Generalization takes a closer look at the trained models, as entire functions, and shows they have rich structure beyond just their test error.

The first two works involve primarily the "bulk landscape" $L(N, S, T)$: the test error of a model

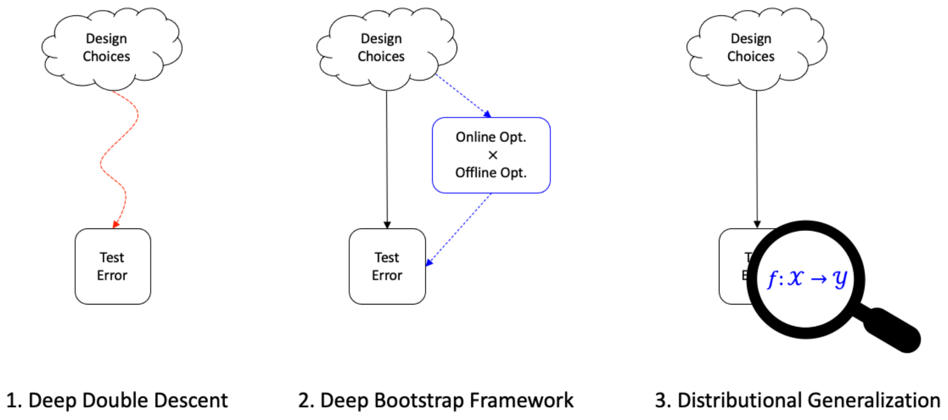

Design Choices
Test Error
Design Choices
Online Opt. × Offline Opt.
Test Error
Design Choices
$f: \mathcal{X} \to \mathcal{Y}$
1. Deep Double Descent
2. Deep Bootstrap Framework
3. Distributional Generalization

Figure 3.1: Schematic representation of the three works in this thesis, as different statements about the map between what we do (design choices) and what we get (trained model). Deep Double Descent shows that this map can be poorly behaved in a certain "critical regime". The Deep Bootstrap Framework shows that, with respect to test error, this map approximately "factorizes" into online optimization and offline optimization aspects. And Distributional Generalization takes a closer look at the trained models, as entire functions, and shows they have rich structure beyond just their test error.

of size S, trained on N samples for T step steps. The last work takes a more fine-grained view of classifiers, beyond just their error. The description here is informal and partial; a more precise and complete description appears in the subsequent chapters.

3.1 Deep Double Descent

This concerns the paper of Nakkiran, Kaplun, Bansal, Yang, Barak, and Sutskever (2020). This paper identifies a systematic obstacle to certain types of theories about the bulk landscape $L(N, S, T)$. Before we fully understand L, we can hope to understand (or even prove) certain structural statements about L. Perhaps the simplest non-trivial statement is *monotonicity*: can we at least show that $L(N, S, T)$ is monotonic in its parameters? That is, can we show that training on more data at least does not hurt performance? (And similarly for larger models, and longer train time). It turns out this is empirically false: L is in practice often non-monotonic. Moreover, it is non-monotonic in a

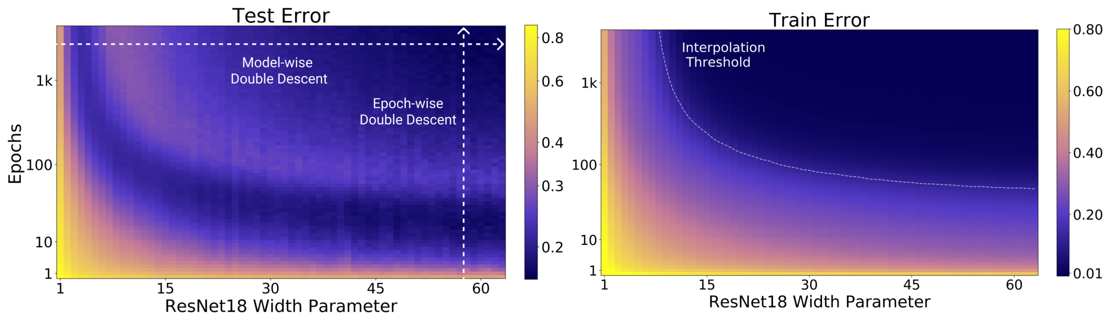


Figure 3.2: Left: Test error as a function of model size and train time. A peak in test error occurs at the "interpolation threshold," when models are barely able to fit their train sets. **Right:** Train error of the corresponding models. Details in Chapter 4.

structured way: the test error peaks at the "interpolation threshold", which is roughly the boundary of the set of (N, S, T) such that the *train loss* is nearly zero. That is, when models are trained to barely fit their train sets, they encounter pathologies which lead to poor test performance (see Figure 3.2). In other words: the relation between what we *do* and what we *get* is not always a continuous and well-behaved one.

These results also reinforce the existence of different "regimes" in deep learning: the boundary between over and under-parameterized regimes is not merely a matter of when a certain theory fails to apply – rather, the boundary shows up as a kind of "phase transition" in the actual behavior of models. It is thus helpful to partition deep learning systems into "overparameterized" vs. "underparameterized" regimes, and only study behaviors far from the boundary.

3.2 The Deep Bootstrap Framework

This concerns the paper of Nakkiran, Neyshabur, and Sedghi (2021a). This paper introduces a conjecture which reduces the problem of generalization to two problems in pure optimization. The main idea is, instead of comparing the test error of classifiers to their train error (as is often done), we rather compare the test error in the real world to test error in a certain "Ideal World." This Ideal

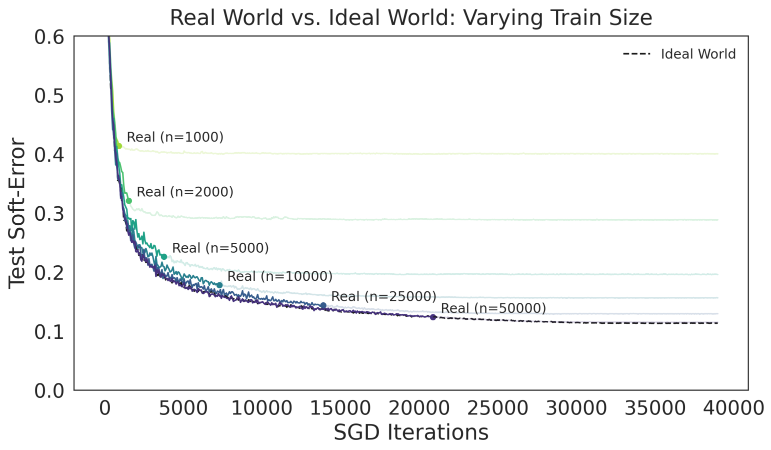


Figure 3.3: Performance in the Ideal World (dashed line) remains close to performance in the Real World (colored lines), until the Real World has converged. The Ideal World effectively has an infinite train set, while the Real World has a finite train set (of varying sizes, shown in different colors). Details in Chapter 5.

World, informally, corresponds to an online optimization setting where we have an infinite stream of samples and optimize directly on the population loss. In such a world, there is no "generalization problem," since there is no discrepancy between train set and test set. It turns out, perhaps surprisingly, that in a certain regime classifiers perform almost identically whether they are trained in the real world (re-using samples from a finite train set) or in the Ideal World (using fresh samples each gradient step) — and we formalize this as our main conjecture.

Our conjecture implies that to understand generalization, it is sufficient to understand empirical optimization and population optimization in isolation. These two optimization problems together determine the generalization performance. Concretely, fix a distribution and model. We consider mini-batch SGD as the optimizer here, for simplicity.[*] Our result relates the following key quantities:

1. $\ell(N)$: The test soft-error on N samples, when trained to convergence.

   That is, $\ell(N) = L(N, S, T = \infty)$.

[*]Our conjecture can be applied to any online optimizer, which is "lifted" to the offline setting by repeating samples.

2. $T(N)$: The time required to converge on N samples, in number of SGD steps.
   Where convergence is defined as reaching some small ε train error, and $T(N) := \infty$ if the model fails to converge (e.g. for small models).

3. $\widetilde{\ell}(T)$: The test soft-error of the model trained for T steps in the "online optimization" setting—with fresh samples in each minibatch, instead of re-using samples from a train set.
   That is, $\widetilde{\ell}(T) = L(N = \infty, S, T)$.

We then claim that, for all S and all N:

$$\forall t \leq T(N): \quad L(N, S, t) \approx L(\infty, S, t) \tag{3.1}$$

That is, up until the point they converge, deep networks do not distinguish whether they are trained with re-used samples (i.e. finite N) or fresh samples in each epoch (i.e. infinite N). See Figure 3.3. This statement connects the "overparameterized" and "underparameterized" regimes: Overparameterized models (with finite N) "behave like" underparameterized models (with large/infinite N) in their test soft-errors. Equation (3.1) is especially interesting at the convergence time, $t = T(N)$. Here, it reduces to the claim that:

$$\ell(N) \approx \widetilde{\ell}(T(N)) \tag{3.2}$$

The left hand side is the quantity we often care about: the generalization performance of models trained to fit their train set. The right hand side involves two purely *optimization*-related quantities. This means the test soft-error of the model we *get* is entirely determined by the following two factors of what we *do*:

1. How quickly online-SGD optimizes the population loss of the model (faster is better).
   This determines $\widetilde{\ell}(\cdot)$.

2. How quickly multi-pass SGD optimizes the empirical loss of the model (slower is better).

This determines $T(\cdot)$.

We do not prove this conjecture[†], but we empirically verify it in a number of diverse settings in deep learning. In fact, we can view all advances in deep learning through their effect on these two factors. Some advances (e.g. convnets for vision, batch norm, residual connections) help primarily because they improve the first factor. Other advances (e.g. regularization, data augmentation) help primarily because they improve the second factor.

These two factors are purely "optimization questions", not generalization ones. This may seem surprising, since in the past optimization was seen an insufficient language[‡] to study generalization—after all, many models can optimize well on their train sets, but only some will generalize well to the test set. The resolution is that we consider two kinds of optimization: on the empirical risk and on the population risk. Each of these alone are insufficient to determine generalization, but taken together are sufficient. We thus hope this work helps encourage the community to better understand online optimization aspects of deep learning.

Equation (3.2) also has implications for scaling laws (Kaplan, McCandlish, Henighan, Brown, Chess, Child, Gray, Radford, Wu, and Amodei, 2020; Hestness, Narang, Ardalani, Diamos, Jun, Kianinejad, Patwary, Yang, and Zhou, 2017): empirically all three quantities $(\ell, T, \widetilde{\ell})$ obey power-law scalings in their parameters. Thus, Equation 3.2 relates the scaling-exponents of data-scaling, time-scaling, and convergence-time scaling.

3.3 Distributional Generalization

This concerns the paper of Nakkiran and Bansal (2020).

This paper introduces a new kind of generalization. Usually, to study classifier how well a clas-

[†]And we cannot prove it, since it is not fully formally specified, for reasons mentioned in the Introduction.

[‡]E.g. http://www.offconvex.org/2019/06/03/trajectories/

sifier "generalizes", we consider a single metric – its test error or test loss (Shalev-Shwartz and Ben-David, 2014). However, there could be many different classifiers with the same test error, and reducing classifiers to a single number misses these rich aspects of their behavior. In this work, we shift the object of study from the *test error* of classifiers, to the entire classification function $f : \mathcal{X} \to \mathcal{Y}$. It turns out that the functions produced by typical machine learning methods have predictable structure, beyond just their test error, which we formally characterize.

Our notion of "distributional generalization" subsumes the classical notion of generalization, and is capable of capturing behaviors that previous notions could not. For example, we show many instances where classical generalization fails— classifiers interpolate and have high test error— but distributional generalization holds in a certain sense. In other words, even classifiers with high test error can be "well behaved" in other senses.

Informally, distributional generalization claims that the input-output behavior of a classifier on its train set is close *in distribution* to its behavior on the test distribution. That is,

$$(x_i, f(x_i)) \approx (x, f(x)) \tag{3.3}$$

where (x_i, y_i) is a sample from the train set of f, and $(x, y) \sim \mathcal{D}$ is a sample from the test distribution. Both distributions include all the randomness involved in sampling a train set and training f. The exact form of distributional closeness in Equation (3.3) depends on problem parameters: the type of model, number of samples, distribution, etc. In our work, we give several formal conjectures which make Equation (3.3) precise.

One of the implications of our work is: even though we use learning methods designed for classification, the object we *get* does not always converge to the Bayes-optimal classifier. Rather, interpolating methods approximate an optimal *sampler*: a sample from $f(x)$ "looks like" a sample from $p(y|x)$ in a certain sense. This suggests, somewhat counterintuitively, that if we want to build a theory of classification methods used in practice, we may want to pursue a theory of *samplers*, as opposed to a theory of *classifiers*.

We defer a full discussion of the implications of this work to Chapter 6, in Section 6.1.3.

4 Deep Double Descent

In this chapter, we show that a variety of modern deep learning tasks exhibit a "double-descent" phenomenon where, as we increase model size, performance first gets *worse* and then gets better. Moreover, we show that double descent occurs not just as a function of model size, but also as a function of the number of training epochs. We unify the above phenomena by defining a new complexity measure we call the *effective model complexity* and conjecture a generalized double descent with respect to this measure. Furthermore, our notion of model complexity allows us to identify certain regimes where increasing (even quadrupling) the number of train samples actually *hurts* test performance.

4.1 Introduction

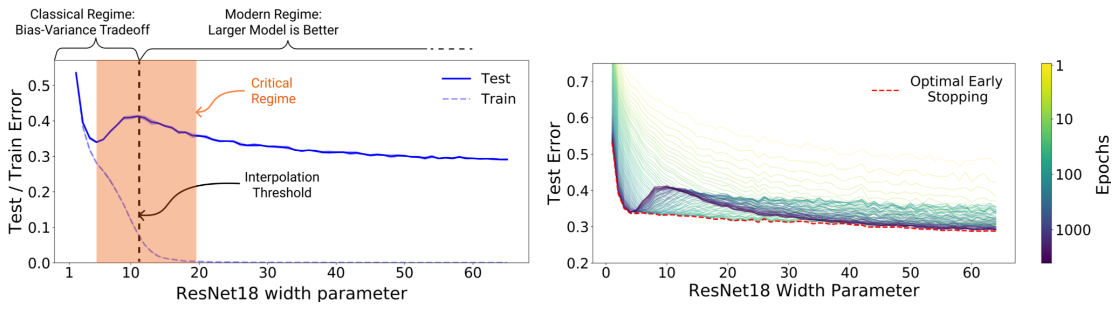


Figure 4.1: **Left:** Train and test error as a function of model size, for ResNet18s of varying width on CIFAR-10 with 15% label noise. **Right:** Test error, shown for varying train epochs. All models trained using Adam for 4K epochs. The largest model (width 64) corresponds to standard ResNet18.

The *bias-variance trade-off* is a fundamental concept in classical statistical learning theory (e.g., Hastie, Tibshirani, Friedman, and Franklin (2005)). The idea is that models of higher complexity have lower bias but higher variance. According to this theory, once model complexity passes a certain threshold, models "overfit" with the variance term dominating the test error, and hence from this point onward, increasing model complexity will only *decrease* performance (i.e., increase test error). Hence conventional wisdom in classical statistics is that, once we pass a certain threshold, *"larger models are worse."*

However, modern neural networks exhibit no such phenomenon. Such networks have millions of parameters, more than enough to fit even random labels (Zhang, Bengio, Hardt, Recht, and Vinyals (2017a)), and yet they perform much better on many tasks than smaller models. Indeed, conventional wisdom among practitioners is that *"larger models are better"* (Krizhevsky et al. (2012), Huang, Cheng, Bapna, Firat, Chen, Chen, Lee, Ngiam, Le, Wu, and Chen (2019), Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke, and Rabinovich (2015), Radford, Wu, Child, Luan, Amodei, and Sutskever (2019)). The effect of training time on test performance is also up for debate. In some settings, "early stopping" improves test performance, while in other set-

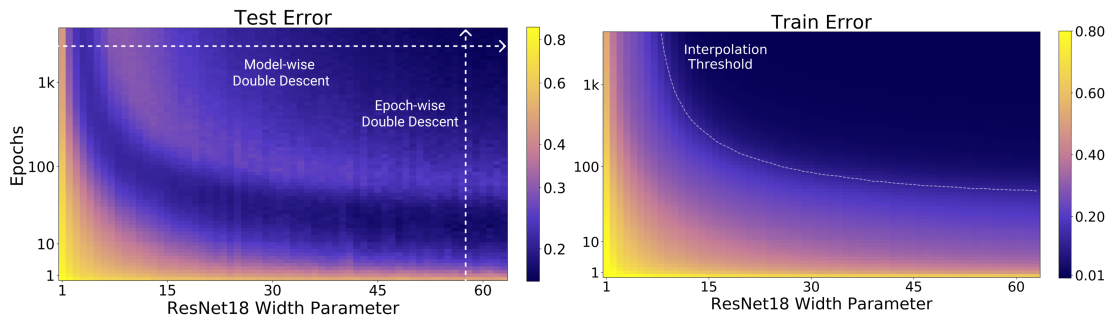


Figure 4.2: **Left:** Test error as a function of model size and train epochs. The horizontal line corresponds to model-wise double descent–varying model size while training for as long as possible. The vertical line corresponds to epoch-wise double descent, with test error undergoing double-descent as train time increases. **Right** Train error of the corresponding models. All models are Resnet18s trained on CIFAR-10 with 15% label noise, data-augmentation, and Adam for up to 4K epochs.

tings training neural networks to zero training error only improves performance. Finally, if there is one thing both classical statisticians and deep learning practitioners agree on is *"more data is always better"*.

In this paper, we present empirical evidence that both reconcile and challenge some of the above "conventional wisdoms." We show that many deep learning settings have two different regimes. In the *under-parameterized* regime, where the model complexity is small compared to the number of samples, the test error as a function of model complexity follows the U-like behavior predicted by the classical bias/variance tradeoff. However, once model complexity is sufficiently large to *interpolate* i.e., achieve (close to) zero training error, then increasing complexity only *decreases* test error, following the modern intuition of "bigger models are better". Such a phenomenon was first postulated by Belkin et al. (2019a) who named it "double descent", and demonstrated it for decision trees, random features, and 2-layer neural networks with ℓ_2 loss, on a variety of learning tasks including MNIST and CIFAR-10.

Main contributions. We show that double descent is a robust phenomenon that occurs in a variety of tasks, architectures, and optimization methods (see Figure 4.1 and Section 4.5). Moreover, we propose a much more general notion of "double descent" that goes beyond varying the number of parameters. We define the *effective model complexity (EMC)* of a training procedure as the maximum number of samples on which it can achieve close to zero training error. The EMC depends not just on the data distribution and the architecture of the classifier but also on the training procedure—and in particular increasing training time will increase the EMC.

We hypothesize that for many natural models and learning algorithms, double descent occurs as a function of the EMC. Indeed we observe "epoch-wise double descent" when we keep the model fixed and increase the training time, with performance following a classical U-like curve in the underfitting stage (when the EMC is smaller than the number of samples) and then improving with training time once the EMC is sufficiently larger than the number of samples (see Figure 4.2). As a corollary, early stopping only helps in the relatively narrow parameter regime of critically parameterized models.

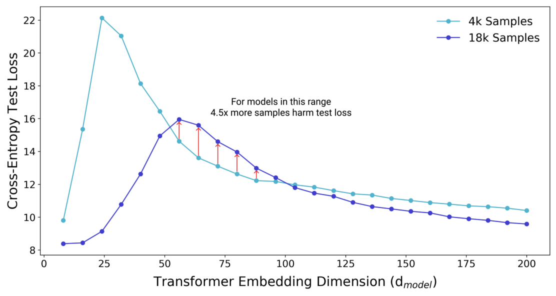


Figure 4.3: Test loss (per-token perplexity) as a function of Transformer model size (embedding dimension d_{model}) on language translation (IWSLT'14 German-to-English). The curve for 18k samples is generally lower than the one for 4k samples, but also shifted to the right, since fitting 18k samples requires a larger model. Thus, for some models, the performance for 18k samples is *worse* than for 4k samples.

Sample non-monotonicity. Finally, our results shed light on test performance as a function of the number of train samples. Since the test error peaks around the point where EMC matches the number of samples (the transition from the under- to over-parameterization), increasing the number of samples has the effect of shifting this peak to the right. While in most settings increasing the number of samples decreases error, this shifting effect can sometimes result in a setting where *more data is worse!* For example, Figure 4.3 demonstrates cases in which increasing the number of samples by a factor of 4.5 results in worse test performance.

4.2 Our results

To state our hypothesis more precisely, we define the notion of *effective model complexity*. We define a *training procedure* $\mathcal{T}$ to be any procedure that takes as input a set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of labeled training samples and outputs a classifier $\mathcal{T}(S)$ mapping data to labels. We define the *effective model complexity* of $\mathcal{T}$ (w.r.t. distribution $\mathcal{D}$) to be the maximum number of samples n on which $\mathcal{T}$ achieves on average ≈ 0 *training error*.

Definition 1 (Effective Model Complexity). *The* Effective Model Complexity *(EMC) of a training procedure $\mathcal{T}$, with respect to distribution $\mathcal{D}$ and parameter $\varepsilon > 0$, is defined as:*

$$\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{T}) := \max\left\{n \mid \mathop{\mathbb{E}}_{S\sim\mathcal{D}^n}[\mathrm{Error}_S(\mathcal{T}(S))] \leq \varepsilon\right\}$$

where $\mathrm{Error}_S(M)$ *is the mean error of model M on train samples S.*

Our main hypothesis can be informally stated as follows:

Hypothesis 1 (Generalized Double Descent hypothesis, informal). *For any natural data distribution $\mathcal{D}$, neural-network-based training procedure $\mathcal{T}$, and small $\varepsilon > 0$, if we consider the task of predicting labels based on n samples from $\mathcal{D}$ then:*

Under-paremeterized regime. *If* $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{T})$ *is sufficiently smaller than n, any perturbation of $\mathcal{T}$ that increases its effective complexity will decrease the test error.*

Over-parameterized regime. *If* $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{T})$ *is sufficiently larger than n, any perturbation of $\mathcal{T}$ that increases its effective complexity will decrease the test error.*

Critically parameterized regime. *If* $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{T}) \approx n$, *then a perturbation of* $\mathcal{T}$ *that increases its effective complexity might decrease* **or** **increase** *the test error.*

Hypothesis 1 is informal in several ways. We do not have a principled way to choose the parameter ε (and currently heuristically use $\varepsilon = 0.1$). We also are yet to have a formal specification for "sufficiently smaller" and "sufficiently larger". Our experiments suggest that there is a *critical interval* around the *interpolation threshold* when $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{T}) = n$: below and above this interval increasing complexity helps performance, while within this interval it may hurt performance. The width of the critical interval depends on both the distribution and the training procedure in ways we do not yet completely understand.

We believe Hypothesis 1 sheds light on the interaction between optimization algorithms, model size, and test performance and helps reconcile some of the competing intuitions about them. The main result of this paper is an experimental validation of Hypothesis 1 under a variety of settings, where we considered several natural choices of datasets, architectures, and optimization algorithms, and we changed the "interpolation threshold" by varying the number of model parameters, the length of training, the amount of label noise in the distribution, and the number of train samples.

Model-wise Double Descent. In Section 4.5, we study the test error of models of increasing size, for a fixed large number of optimization steps. We show that "model-wise double-descent" occurs for various modern datasets (CIFAR-10, CIFAR-100, IWSLT'14 de-en, with varying amounts of label noise), model architectures (CNNs, ResNets, Transformers), optimizers (SGD, Adam), number of train samples, and training procedures (data-augmentation, and regularization). Moreover, the peak in test error systematically occurs at the interpolation threshold. In particular, we demonstrate realistic settings in which *bigger models are worse*.

Epoch-wise Double Descent. In Section 4.6, we study the test error of a fixed, large architecture over the course of training. We demonstrate, in similar settings as above, a corresponding peak in test performance when models are trained just long enough to reach ≈ 0 train error. The test error of a large model first decreases (at the beginning of training), then increases (around the critical regime), then decreases once more (at the end of training)—that is, *training longer can correct overfitting*.

Sample-wise Non-monotonicity. In Section 4.7, we study the test error of a fixed model and training procedure, for varying number of train samples. Consistent with our generalized double-descent hypothesis, we observe distinct test behavior in the "critical regime", when the number of samples is near the maximum that the model can fit. This often manifests as a long plateau region, in which taking significantly more data might not help when training to completion (as is the case for CNNs on CIFAR-10). Moreover, we show settings (Transformers on IWSLT'14 en-de), where this manifests as a peak—and for a fixed architecture and training procedure, *more data actually hurts*.

Preliminary results suggest that generalized double descent also holds as we vary the EMC by varying the amount of regularization, for a fixed model (see Figure B.10 in the Appendix).

Remarks on Label Noise. We observe all forms of double descent most strongly in settings with label noise in the train set (as is often the case when collecting train data in the real-world). However, we also show several realistic settings with a test-error peak even without label noise: ResNets (Figure 4.4a) and CNNs (Figure B.8) on CIFAR-100; Transformers on IWSLT'14 (Figure 4.8). Moreover, all our experiments demonstrate distinctly different test behavior in the critical regime—often manifesting as a "plateau" in the test error in the noiseless case which develops into a peak with added label noise. See Section 4.8 for further discussion.

4.3 Related work

Model-wise double descent was first proposed as a general phenomenon by Belkin et al. (2019a). Similar behavior had been observed in Advani and Saxe (2017) and Geiger, Spigler, d'Ascoli, Sagun, Baity-Jesi, Biroli, and Wyart (2019b). Subsequently, there has been a large body of work studying the double descent phenomenon. A growing list of papers that theoretically analyze it in the tractable setting of linear least squares regression includes Belkin, Hsu, and Xu (2019b); Hastie et al. (2019); Bartlett et al. (2020); Muthukumar, Vodrahalli, and Sahai (2019); Bibas, Fogel, and Feder (2019); Mitra (2019); Mei and Montanari (2019). Moreover, Geiger, Jacot, Spigler, Gabriel, Sagun, d'Ascoli, Biroli, Hongler, and Wyart (2019a) provide preliminary results for model-wise double descent in convolutional networks trained on CIFAR-10. Our work differs from the above papers in two crucial aspects: First, we extend the idea of double-descent beyond the number of parameters to incorporate the training procedure under a unified notion of "Effective Model Complexity", leading to novel insights like epoch-wise double descent and sample non-monotonicity. The notion that increasing train time corresponds to increasing complexity was also presented in Nakkiran, Kaplun, Kalimeris, Yang, Zhang, Edelman, and Barak (2019). Second, we provide an extensive and rigorous demonstration of double-descent for modern practices spanning a variety of architectures, datasets optimization procedures. An extended discussion of the related work is provided in Appendix B.2.

4.4 Experimental Setup

We briefly describe the experimental setup here; full details are in Appendix B.1 [*]. We consider three families of architectures: ResNets, standard CNNs, and Transformers. **ResNets:** We parameterize a family of ResNet18s (He, Zhang, Ren, and Sun (2016b)) by scaling the width (number of filters) of convolutional layers. Specifically, we use layer widths $[k, 2k, 4k, 8k]$ for varying k. The standard ResNet18 corresponds to $k = 64$. **Standard CNNs:** We consider a simple family of 5-layer CNNs, with 4 convolutional layers of widths $[k, 2k, 4k, 8k]$ for varying k, and a fully-connected layer. For context, the CNN with width $k = 64$, can reach over 90% test accuracy on CIFAR-10 with data-

[*]The raw data from our experiments are available at: `https://gitlab.com/harvard-machine-learning/double-descent/tree/master`

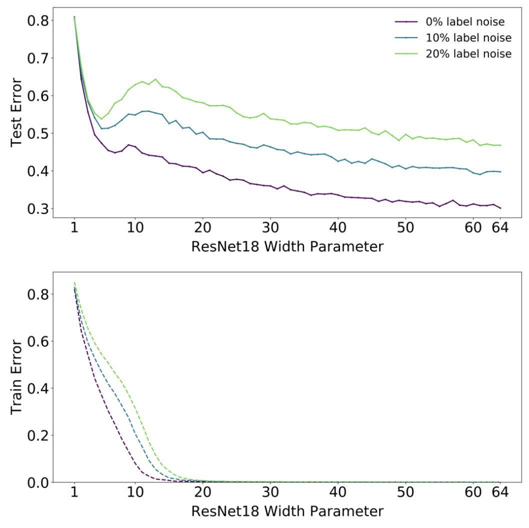
augmentation. **Transformers:** We consider the 6 layer encoder-decoder from Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin (2017a), as implemented by Ott, Edunov, Baevski, Fan, Gross, Ng, Grangier, and Auli (2019). We scale the size of the network by modifying the embedding dimension d_{model}, and setting the width of the fully-connected layers proportionally ($d_{\text{ff}} = 4 \cdot d_{\text{model}}$). For ResNets and CNNs, we train with cross-entropy loss, and the following optimizers: (1) Adam with learning-rate 0.0001 for 4K epochs; (2) SGD with learning rate $\propto \frac{1}{\sqrt{T}}$ for 500K gradient steps. We train Transformers for 80K gradient steps, with 10% label smoothing and no drop-out.

Label Noise. In our experiments, label noise of probability p refers to training on a samples which have the correct label with probability $(1-p)$, and a uniformly random incorrect label otherwise (label noise is sampled only once and not per epoch). Figure 4.1 plots test error on the noisy distribution, while the remaining figures plot test error with respect to the clean distribution (the two curves are just linear rescaling of one another).

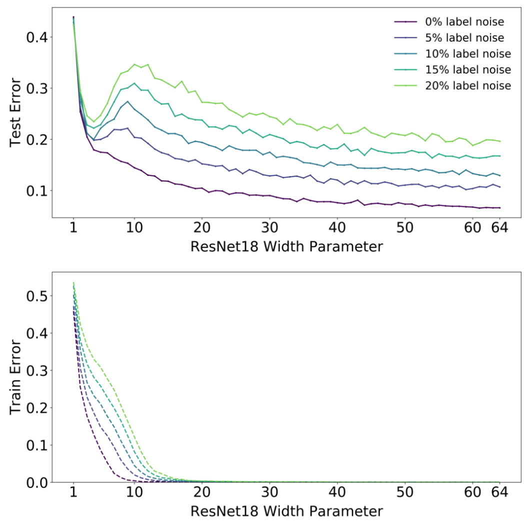
4.5 Model-wise Double Descent

In this section, we study the test error of models of increasing size, when training to completion (for a fixed large number of optimization steps). We demonstrate model-wise double descent across different architectures, datasets, optimizers, and training procedures. The critical region exhibits distinctly different test behavior around the interpolation point and there is often a peak in test error that becomes more prominent in settings with label noise.

For the experiments in this section (Figures 4.4, 4.5, 4.6, 4.7, 4.8), notice that all modifications which increase the interpolation threshold (such as adding label noise, using data augmentation, and increasing the number of train samples) also correspondingly shift the peak in test error towards larger models. Additional plots showing the early-stopping behavior of these models, and addi-



(a) CIFAR-100. There is a peak in test error even with no label noise.



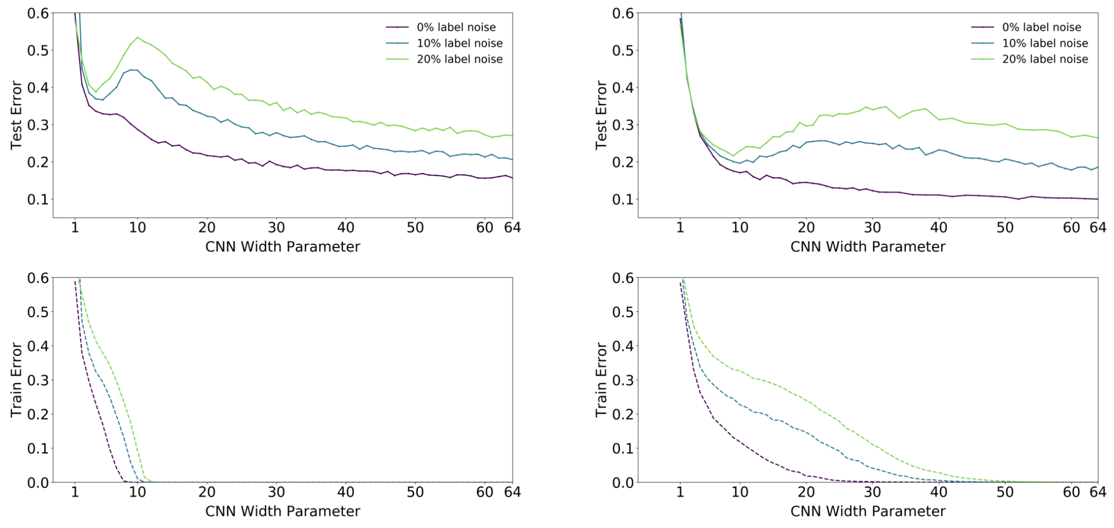
(b) CIFAR-10. There is a "plateau" in test error around the interpolation point with no label noise, which develops into a peak for added label noise.

Figure 4.4: Model-wise double descent for ResNet18s. Trained on CIFAR-100 and CIFAR-10, with varying label noise. Optimized using Adam with LR 0.0001 for 4K epochs, and data-augmentation.

tional experiments showing double descent in settings with no label noise (e.g. Figure B.7) are in Appendix B.4.2. We also observed model-wise double descent for adversarial training, with a prominent robust test error peak even in settings without label noise. See Figure B.14 in Appendix B.4.2.

Discussion. Fully understanding the mechanisms behind model-wise double descent in deep neural networks remains an important open question. However, an analog of model-wise double descent occurs even for linear models. A recent stream of theoretical works analyzes this setting (Bartlett et al. (2020); Muthukumar et al. (2019); Belkin et al. (2019b); Mei and Montanari (2019); Hastie et al. (2019)). We believe similar mechanisms may be at work in deep neural networks.

Informally, our intuition is that for model-sizes at the interpolation threshold, there is effectively only one model that fits the train data and this interpolating model is very sensitive to noise in the train set and/or model mis-specification. That is, since the model is just barely able to fit the



(a) Without data augmentation.

(b) With data augmentation.

Figure 4.5: Effect of Data Augmentation. 5-layer CNNs on CIFAR10, with and without data-augmentation. Data-augmentation shifts the interpolation threshold to the right, shifting the test error peak accordingly. Optimized using SGD for 500K steps. See Figure B.15 for larger models.

train data, forcing it to fit even slightly-noisy or mis-specified labels will destroy its global structure, and result in high test error. (See Figure B.16 in the Appendix for an experiment demonstrating this noise sensitivity, by showing that ensembling helps significantly in the critically-parameterized regime). However for over-parameterized models, there are many interpolating models that fit the train set, and SGD is able to find one that "memorizes" (or "absorbs") the noise while still performing well on the distribution.

The above intuition is theoretically justified for linear models. In general, this situation manifests even without label noise for linear models (Mei and Montanari (2019)), and occurs whenever there is *model mis-specification* between the structure of the true distribution and the model family. We believe this intuition extends to deep learning as well, and it is consistent with our experiments.

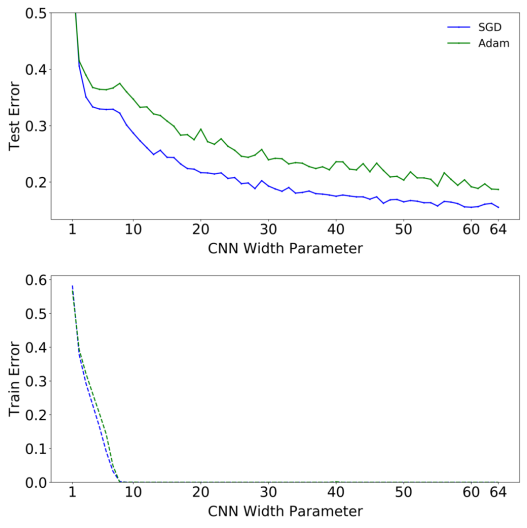


Figure 4.6: SGD vs. Adam. 5-Layer CNNs on CIFAR-10 with no label noise, and no data augmentation. Optimized using SGD for 500K gradient steps, and Adam for 4K epochs.

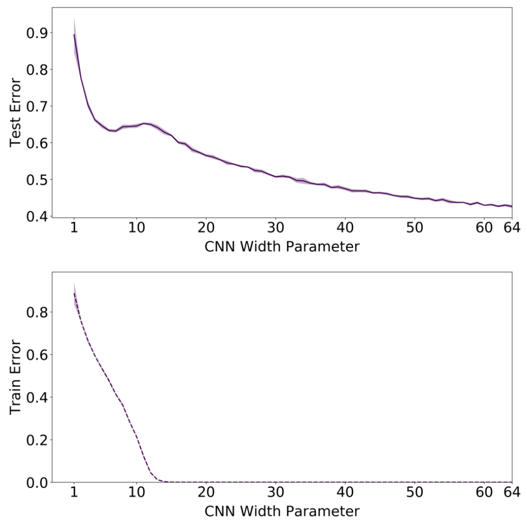


Figure 4.7: Noiseless settings. 5-layer CNNs on CIFAR-100 with no label noise; note the peak in test error. Trained with SGD and no data augmentation. See Figure B.8 for the early-stopping behavior of these models.

4.6 Epoch-wise Double Descent

In this section, we demonstrate a novel form of double-descent with respect to training epochs, which is consistent with our unified view of effective model complexity (EMC) and the generalized double descent hypothesis. Increasing the train time increases the EMC—and thus a sufficiently large model transitions from under- to over-parameterized over the course of training.

As illustrated in Figure 4.9, sufficiently large models can undergo a "double descent" behavior where test error first decreases then increases near the interpolation threshold, and then decreases again. In contrast, for "medium sized" models, for which training to completion will only barely reach ≈ 0 error, the test error as a function of training time will follow a classical U-like curve where it is better to stop early. Models that are too small to reach the approximation threshold will remain in the "under parameterized" regime where increasing train time monotonically decreases test

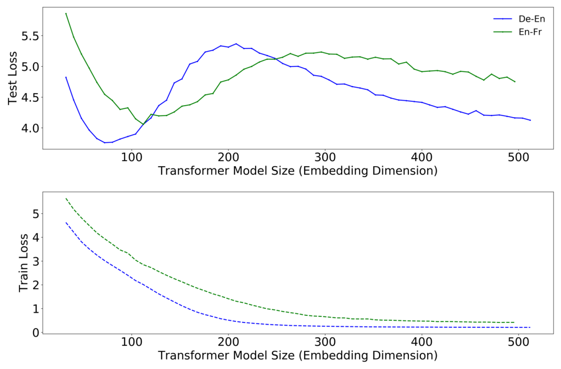


Figure 4.8: Transformers on language translation tasks: Multi-head-attention encoder-decoder Transformer model trained for 80k gradient steps with labeled smoothed cross-entropy loss on IWSLT'14 German-to-English (160K sentences) and WMT'14 English-to-French (subsampled to 200K sentences) dataset. Test loss is measured as per-token perplexity.

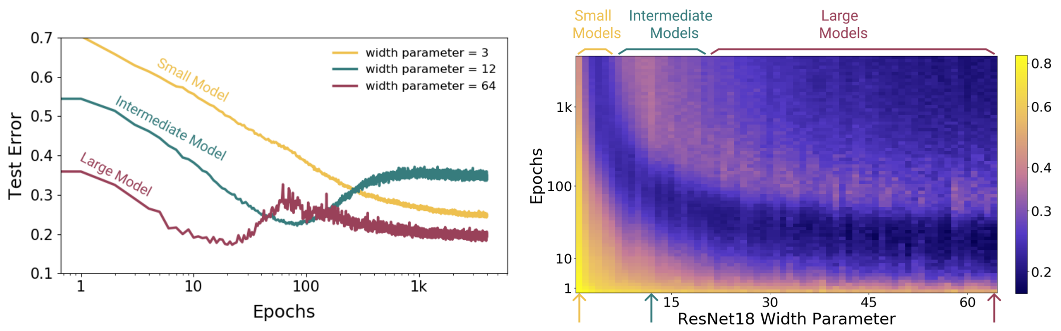
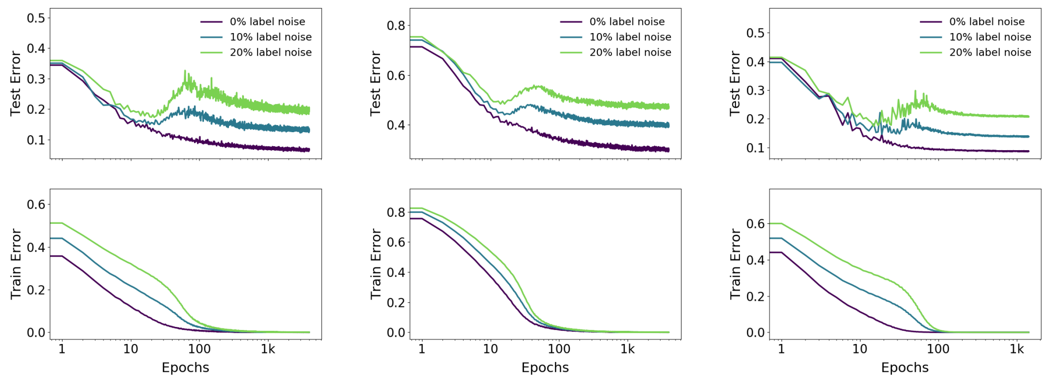


Figure 4.9: Left: Training dynamics for models in three regimes. Models are ResNet18s on CIFAR10 with 20% label noise, trained using Adam with learning rate 0.0001, and data augmentation. Right: Test error over (Model size $\times$ Epochs). Three slices of this plot are shown on the left.

error. Our experiments (Figure 4.10) show that many settings of dataset and architecture exhibit epoch-wise double descent, in the presence of label noise. Further, this phenomenon is robust across optimizer variations and learning rate schedules (see additional experiments in Appendix B.4.1). As in model-wise double descent, the test error peak is accentuated with label noise.

Conventional wisdom suggests that training is split into two phases: (1) In the first phase, the network learns a function with a small generalization gap (2) In the second phase, the network starts to over-fit the data leading to an increase in test error. Our experiments suggest that this is not the complete picture—in some regimes, the test error decreases again and may achieve a lower value at the end of training as compared to the first minimum (see Fig 4.10 for 10% label noise).



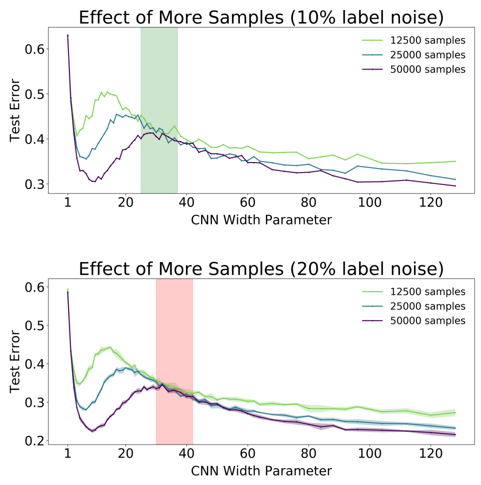
(a) ResNet18 on CIFAR10. **(b)** ResNet18 on CIFAR100. **(c)** 5-layer CNN on CIFAR 10.

Figure 4.10: Epoch-wise double descent for ResNet18 and CNN (width=128). ResNets trained using Adam with learning rate 0.0001, and CNNs trained with SGD with inverse-squareroot learning rate.

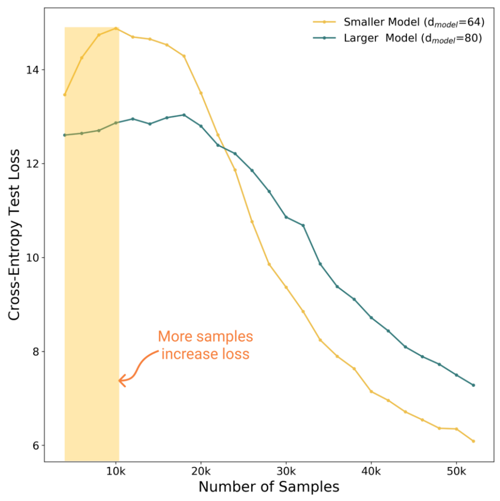
4.7 Sample-wise Non-monotonicity

In this section, we investigate the effect of varying the number of train samples, for a fixed model and training procedure. Previously, in model-wise and epoch-wise double descent, we explored behavior in the critical regime, where $\mathrm{EMC}_{\mathcal{D},\varepsilon}(\mathcal{T}) \approx n$, by varying the EMC. Here, we explore the critical regime by varying the number of train samples n. By increasing n, the same training procedure $\mathcal{T}$ can switch from being effectively over-parameterized to effectively under-parameterized.

We show that increasing the number of samples has two different effects on the test error vs. model complexity graph. On the one hand, (as expected) increasing the number of samples shrinks the area under the curve. On the other hand, increasing the number of samples also has the effect of "shifting the curve to the right" and increasing the model complexity at which test error peaks.



(a) Model-wise double descent for 5-layer CNNs on CIFAR-10, for varying dataset sizes. **Top:** There is a range of model sizes (shaded green) where training on $2\times$ more samples does not improve test error. **Bottom:** There is a range of model sizes (shaded red) where training on $4\times$ more samples does not improve test error.



(b) Sample-wise non-monotonicity. Test loss (per-word perplexity) as a function of number of train samples, for two transformer models trained to completion on IWSLT'14. For both model sizes, there is a regime where more samples hurt performance. Compare to Figure 4.3, of model-wise double-descent in the identical setting.

Figure 4.11: Sample-wise non-monotonicity.

These twin effects are shown in Figure 4.11a. Note that there is a range of model sizes where the effects "cancel out"—and having $4\times$ more train samples does not help test performance when training to completion. Outside the critically-parameterized regime, for sufficiently under- or over-parameterized models, having more samples helps. This phenomenon is corroborated in Figure 4.12, which shows test error as a function of both model and sample size, in the same setting as Figure 4.11a.

In some settings, these two effects combine to yield a regime of model sizes where more data actually hurts test performance as in Figure 4.3 (see also Figure 4.11b). Note that this phenomenon is not unique to DNNs: more data can hurt even for linear models (see Appendix B.3).

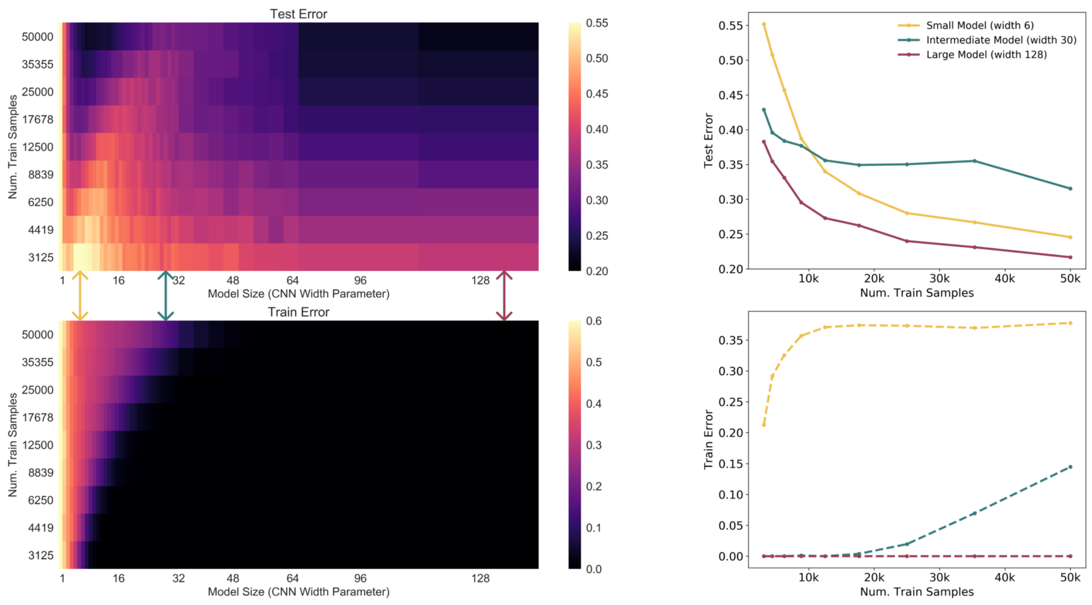


Figure 4.12: Left: Test Error as a function of model size and number of train samples, for 5-layer CNNs on CIFAR-10 + 20% noise. Right: Three slices of the left plot, showing the effect of more data for models of different sizes. Note that, when training to completion, more data helps for small and large models, but does not help for near-critically-parameterized models (green).

4.8 Conclusion and Discussion

We introduce a generalized double descent hypothesis: models and training procedures exhibit atypical behavior when their Effective Model Complexity is comparable to the number of train samples. We provide extensive evidence for our hypothesis in modern deep learning settings, and show that it is robust to choices of dataset, architecture, and training procedures. In particular, we demonstrate "model-wise double descent" for modern deep networks and characterize the regime where bigger models can perform worse. We also demonstrate "epoch-wise double descent," which, to the best of our knowledge, has not been previously proposed. Finally, we show that the double descent phenomenon can lead to a regime where training on more data leads to worse test performance. Preliminary results suggest that double descent also holds as we vary the amount of regularization for a fixed model (see Figure B.10).

We also believe our characterization of the critical regime provides a useful way of thinking for practitioners—if a model and training procedure are just barely able to fit the train set, then small changes to the model or training procedure may yield unexpected behavior (e.g. making the model slightly larger or smaller, changing regularization, etc. may hurt test performance).

Early stopping. We note that many of the phenomena that we highlight often do not occur with optimal early-stopping. However, this is consistent with our generalized double descent hypothesis: if early stopping prevents models from reaching 0 train error then we would not expect to see double-descent, since the EMC does not reach the number of train samples. Further, we show at least one setting where model-wise double descent can still occur even with optimal early stopping (ResNets on CIFAR-100 with no label noise, see Figure B.7). We have not observed settings where more data hurts when optimal early-stopping is used. However, we are not aware of reasons which preclude this from occurring. We leave fully understanding the optimal early stopping behavior of double descent as an important open question for future work.

Label Noise. In our experiments, we observe double descent most strongly in settings with label noise. However, we believe this effect is not fundamentally about label noise, but rather about *model mis-specification*. For example, consider a setting where the label noise is not truly random, but rather pseudorandom (with respect to the family of classifiers being trained). In this setting, the performance of the Bayes optimal classifier would not change (since the pseudorandom noise is deterministic, and invertible), but we would observe an identical double descent as with truly random label noise. Thus, we view adding label noise as merely a proxy for making distributions "harder"— i.e. increasing the amount of model mis-specification.

Other Notions of Model Complexity. Our notion of *Effective Model Complexity* is related to classical complexity notions such as Rademacher complexity, but differs in several crucial ways: (1) EMC depends on the *true labels* of the data distribution, and (2) EMC depends on the training procedure, not just the model architecture.

Other notions of model complexity which do not incorporate features (1) and (2) would not suffice to characterize the location of the double-descent peak. Rademacher complexity, for example, is determined by the ability of a model architecture to fit a randomly-labeled train set. But Rademacher complexity and VC dimension are both insufficient to determine the model-wise double descent peak location, since they do not depend on the distribution of labels— and our experiments show that adding label noise shifts the location of the peak.

Moreover, both Rademacher complexity and VC dimension depend only on the model family and data distribution, and not on the training procedure used to find models. Thus, they are not capable of capturing train-time double-descent effects, such as "epoch-wise" double descent, and the effect of data-augmentation on the peak location.

5 The Deep Bootstrap Framework

Here we propose a new framework for reasoning about generalization in deep learning. The core idea is to couple the Real World, where samples are re-used in multiple epochs, to an "Ideal World", where we see fresh samples in every batch. It turns out that in practice, the gap between these two worlds is often universally small. This reduces the problem of *generalization* in offline learning to the problem of *optimization* in online learning.

In particular, our results imply that a good deep learning method is one which (1) optimizes quickly on the population loss, and (2) optimizes slowly on the empirical loss.

We can use this to gain a new optimization-perspective on many phenomena and design choices in deep learning. For example, CNNs often generalize better than MLPs on image distributions in the Real World, but this is "because" they optimize faster on the population loss in the Ideal World.

We can similarly give optimization-perspectives for the effect of pre-training, data-augmentation, regularization, learning rate, etc. We thus hope our framework encourages researchers to consider generalization through the (perhaps simpler) lens of optimization.

5.1 Introduction

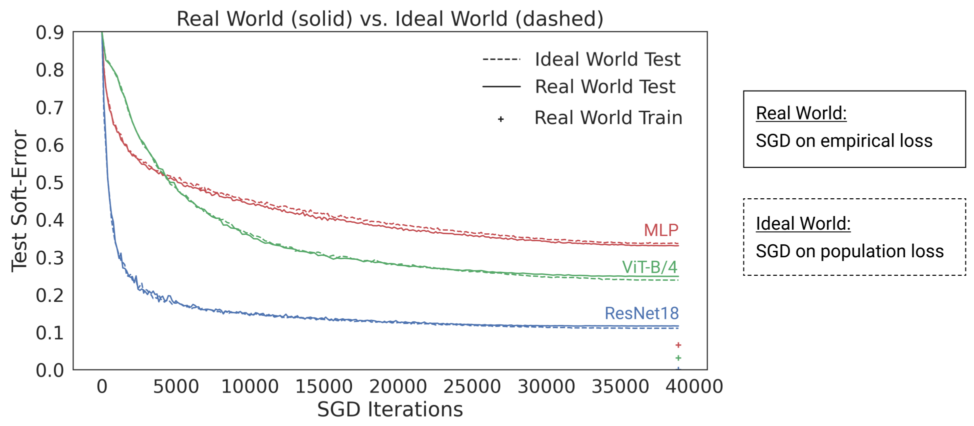


Figure 5.1: Three architectures trained from scratch on CIFAR-5m, a CIFAR-10-like task. The Real World is trained on 50K samples for 100 epochs, while the Ideal World is trained on 5M samples in 1 pass. The Real World Test remains close to Ideal World Test, despite a large generalization gap.

The goal of a generalization theory in supervised learning is to understand when and why trained models have small test error. The classical framework of generalization decomposes the test error of a model f_t as:

$$\mathrm{TestError}(f_t) = \mathrm{TrainError}(f_t) + \underbrace{[\mathrm{TestError}(f_t) - \mathrm{TrainError}(f_t)]}_{\text{Generalization gap}} \tag{5.1}$$

and studies each part separately (e.g. Vapnik and Chervonenkis (1971); Blumer, Ehrenfeucht, Haussler, and Warmuth (1989); Shalev-Shwartz and Ben-David (2014)). Many works have applied this framework to study generalization of deep networks (e.g. Bartlett (1997); Bartlett, Maiorov, and Meir (1999); Bartlett and Mendelson (2001); Anthony and Bartlett (2009); Neyshabur, Tomioka, and Srebro (2015b); Dziugaite and Roy (2017); Bartlett, Foster, and Telgarsky (2017); Neyshabur, Bhojanapalli, and Srebro (2018a); Harvey, Liaw, and Mehrabian (2017); Golowich, Rakhlin, and Shamir (2018); Arora, Ge, Neyshabur, and Zhang (2018); Arora, Du, Hu, Li, and Wang (2019);

Allen-Zhu, Li, and Liang (2019); Long and Sedghi (2020); Wei and Ma (2020)). However, there are at least two obstacles to understanding generalization of modern neural networks via the classical approach.

1. Modern methods can *interpolate*, reaching TrainError ≈ 0, while still performing well. In these settings, the decomposition of Equation (5.1) does not actually reduce test error into two different subproblems: it amounts to writing TestError = $0 +$ TestError. That is, understanding the generalization gap here is exactly equivalent to understanding the test error itself.
2. Most if not all techniques for understanding the generalization gap (e.g. uniform convergence, VC-dimension, regularization, stability, margins) remain vacuous (Zhang et al., 2017b; Belkin et al., 2018a,b; Nagarajan and Kolter, 2019) and not predictive (Nagarajan and Kolter, 2019; Jiang, Neyshabur, Mobahi, Krishnan, and Bengio, 2020; Dziugaite, Drouin, Neal, Rajkumar, Caballero, Wang, Mitliagkas, and Roy, 2020) for modern networks.

In this work, we propose an alternate approach to understanding generalization to help overcome these obstacles. The key idea is to consider an alternate decomposition:

$$\mathrm{TestError}(f_t) = \underbrace{\mathrm{TestError}(f_t^{\mathrm{iid}})}_{\text{A: Online Learning}} + \underbrace{[\mathrm{TestError}(f_t) - \mathrm{TestError}(f_t^{\mathrm{iid}})]}_{\text{B: Bootstrap error}} \tag{5.2}$$

where f_t is the neural-network after t optimization steps (the "Real World"), and f_t^{iid} is a network trained identically to f_t, but using fresh samples from the distribution in each mini-batch step (the "Ideal World"). That is, f_t^{iid} is the result of optimizing on the *population loss* for t steps, while f_t is the result of optimizing on the *empirical loss* as usual (we define this more formally later).

This leads to a different decoupling of concerns, and proposes an alternate research agenda to understand generalization. To understand generalization in the bootstrap framework, it is sufficient to understand:

(A) **Online Learning:** How quickly models optimize on the population loss, in the infinite-data regime (the Ideal World).

(B) **Finite-Sample Deviations:** How closely models behave in the finite-data vs. infinite-data regime (the bootstrap error).

Although neither of these points are theoretically understood for deep networks, they are closely related to rich areas in optimization and statistics, whose tools have not been brought fully to bear on the problem of generalization. The first part (A) is purely a question in online stochastic optimization: We have access to a stochastic gradient oracle for a population loss function, and we are interested in how quickly an online optimization algorithm (e.g. SGD, Adam) reaches small population loss. This problem is well-studied in the online learning literature for convex functions (Bubeck, 2011; Hazan, 2019; Shalev-Shwartz et al., 2011), and is an active area of research in non-convex settings (Jin, Ge, Netrapalli, Kakade, and Jordan, 2017; Lee, Simchowitz, Jordan, and Recht, 2016; Jain and Kar, 2017; Gao, Li, and Zhang, 2018; Yang, Deng, Hajiesmaili, Tan, and Wong, 2018; Maillard and Munos, 2010). In the context of neural networks, optimization is usually studied on the empirical loss landscape (Arora et al., 2019; Allen-Zhu et al., 2019), but we propose studying optimization on the population loss landscape directly. This highlights a key difference in our approach: we never compare test and train quantities— we only consider test quantities.

The second part (B) involves approximating fresh samples with "reused" samples, and reasoning about behavior of certain functions under this approximation. This is closely related to the *non-parametric bootstrap* in statistics (Efron, 1979; Efron and Tibshirani, 1986), where sampling from the population distribution is approximated by sampling with replacement from an empirical distribution. Bootstrapped estimators are widely used in applied statistics, and their theoretical properties are known in certain cases (e.g. Hastie, Tibshirani, and Friedman (2009); James, Witten, Hastie, and Tibshirani (2013); Efron and Hastie (2016); Van der Vaart (2000)). Although current bootstrap theory does not apply to neural networks, it is conceivable that these tools could eventually be extended to our setting.

Experimental Validation. Beyond the theoretical motivation, our main experimental claim is that the bootstrap decomposition is actually useful: in realistic settings, the bootstrap error is often small, and the performance of real classifiers is largely captured by their performance in the Ideal World. Figure 5.1 shows one example of this, as a preview of our more extensive experiments in Section 5.4. We plot the test error of a ResNet (He et al., 2016a), an MLP, and a Vision Transformer (Dosovitskiy et al., 2021) on a CIFAR-10-like task, over increasing minibatch SGD iterations. The Real World is trained on 50K samples for 100 epochs. The Ideal World is trained on 5 million samples with a single pass. Notice that the bootstrap error is small for all architectures, although the generalization gap can be large. In particular, the convnet generalizes better than the MLP on finite data, but this is "because" it optimizes faster on the population loss with infinite data. See Appendix C.4.1 for details.

Our Contributions.

- **Framework:** We propose the Deep Bootstrap framework for understanding generalization in deep learning, which connects offline generalization to online optimization. (Section 5.2).
- **Validation:** We give evidence that the bootstrap error is small in realistic settings for supervised image classification, by conducting extensive experiments on large-scale tasks (including variants of CIFAR-10 and ImageNet) for many architectures (Section 5.4). Thus,

  The generalization of models in offline learning is largely determined by their optimization speed in online learning.

- **Implications:** We highlight how our framework can unify and yield insight into important phenomena in deep learning, including implicit bias, model selection, data-augmentation and pretraining (Section 5.5). For example, we observe a surprising phenomena in practice, which is captured by our framework:

The same techniques (architectures and training methods) are used in practice in both over- and under-parameterized regimes.

Additional Related Work. The bootstrap error is also related to algorithmic stability (e.g. Bousquet and Elisseeff (2000); Hardt, Recht, and Singer (2016)), since both quantities involve replacing samples with fresh samples. However, stability-based generalization bounds cannot tightly bound the bootstrap error, since there are many settings where the generalization gap is high, but bootstrap error is low.

Our work is similar in spirit to the theorhetical work of Bottou and LeCun (2003), which shows that online SGD performs equivalently to a version of offline SGD, in a certain convex setting. Pillaud-Vivien, Rudi, and Bach (2018) compares multi-pass and single-pass versions of SGD, and explores settings when multiple passes are optimal.

5.2 The Deep Boostrap

Here we more formally describe the Deep Bootstrap framework and our main claims. Let $\mathcal{F}$ denote a learning algorithm, including architecture and optimizer. We consider optimizers which can be used in online learning, such as stochastic gradient descent and variants. Let $\text{Train}_{\mathcal{F}}(\mathcal{D}, n, t)$ denote training in the "Real World": using the architecture and optimizer specified by $\mathcal{F}$, on a train set of n samples from distribution $\mathcal{D}$, for t optimizer steps. Let $\text{Train}_{\mathcal{F}}(\mathcal{D}, \infty, t)$ denote this same optimizer operating on the population loss (the "Ideal World"). Note that these two procedures use identical architectures, learning-rate schedules, mini-batch size, etc – the only difference is, the Ideal World optimizer sees a fresh minibatch of samples in each optimization step, while the Real World reuses samples in minibatches. Let the Real and Ideal World trained models be:

$$\text{Real World:} \quad f_t \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}, n, t)$$
$$\text{Ideal World:} \quad f_t^{\text{iid}} \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}, \infty, t)$$

We now claim that for all t until the Real World converges, these two models f_t, f_t^{iid} have similar test performance. In our main claims, we differ slightly from the presentation in the Introduction in that we consider the "soft-error" of classifiers instead of their hard-errors. The soft-accuracy of classifiers is defined as the softmax probability on the correct label, and (soft-error) $:= 1 -$ (soft-accuracy). Equivalently, this is the expected error of temperature-1 samples from the softmax distribution. Formally, define ε as the bootstrap error – the gap in soft-error between Real and Ideal worlds at time t:

$$\text{TestSoftError}_{\mathcal{D}}(f_t) = \text{TestSoftError}_{\mathcal{D}}(f_t^{\text{iid}}) + \varepsilon(n, \mathcal{D}, \mathcal{F}, t) \tag{5.3}$$

Our main experimental claim is that the bootstrap error ε is uniformly small in realistic settings.

Claim 1 (Bootstrap Error Bound, informal). *For choices of $(n, \mathcal{D}, \mathcal{F})$ corresponding to realistic settings in deep learning for supervised image classification, the bootstrap error $\varepsilon(n, \mathcal{D}, \mathcal{F}, t)$ is small for all $t \leq T_0$. The "stopping time" T_0 is defined as the time when the Real World reaches small training error (we use 1%) – that is, when Real World training has essentially converged.*

The restriction on $t \leq T_0$ is necessary, since as $t \to \infty$ the Ideal World will continue to improve, but the Real World will at some point essentially stop changing (when train error ≈ 0). However, we claim that these worlds are close for "as long as we can hope"— as long as the Real World optimizer is still moving significantly.

Error vs. Soft-Error. We chose to measure soft-error instead of hard-error in our framework for both empirical and theoretically-motivated reasons. Empirically, we found that the bootstrap gap is often smaller with respect to soft-errors. Theoretically, we want to define the bootstrap gap such that it converges to 0 as data and model size are scaled to infinity. Specifically, if we consider an *overparameterized* scaling limit where the Real World models always interpolate the train data, then Distributional Generalization (Nakkiran and Bansal, 2020) implies that the bootstrap gap for test error will *not* converge to 0 on distributions with non-zero Bayes risk. Roughly, this is because the Ideal World classifier will converge to the Bayes optimal one ($\text{argmax}_y\, p(y|x)$), while the Real World interpolating classifier will converge to a *sampler* from $p(y|x)$. Considering soft-errors instead of errors nullifies this issue. We elaborate further on the differences between the worlds in Section 5.6. See also Appendix C.3 for relations to the nonparametric bootstrap (Efron, 1979).

5.3 Experimental Setup

Our bootstrap framework could apply to any setting where an iterative optimizer for online learning is applied in an offline setting. In this work we primarily consider stochastic gradient descent (SGD) applied to deep neural networks for image classification. This setting is well-developed both in practice and in theory, and thus serves as an appropriate first step to vet theories of generalization, as done in many recent works (e.g. Jiang et al. (2020); Neyshabur, Li, Bhojanapalli, LeCun, and Srebro (2018b); Zhang et al. (2017b); Arora et al. (2019)). Our work does not depend on overparameterization— it holds for both under and over parameterized networks, though it is perhaps most interesting in the overparameterized setting. We now describe our datasets and experimental methodology.

5.3.1 Datasets

Measuring the bootstrap error in realistic settings presents some challenges, since we do not have enough samples to instantiate the Ideal World. For example, for a Real World CIFAR-10 network trained on 50K samples for 100 epochs, the corresponding Ideal World training would require 5 million samples (fresh samples in each epoch). Since we do not have 5 million samples for CIFAR-10, we use the following datasets as proxies. More details, including sample images, in Appendix C.5.

CIFAR-5m. We construct a dataset of 6 million synthetic CIFAR-10-like images, by sampling from the CIFAR-10 Denoising Diffusion generative model of Ho, Jain, and Abbeel (2020), and labeling the unconditional samples by a 98.5% accurate Big-Transfer model (Kolesnikov, Beyer, Zhai, Puigcerver, Yung, Gelly, and Houlsby, 2019). These are synthetic images, but close to CIFAR-10 for research purposes. For example, a WideResNet28-10 trained on 50K samples from CIFAR-5m reaches 91.2% test accuracy on CIFAR-10 test set. We use 5 million images for training, and reserve the rest for the test set. We plan to release this dataset.

ImageNet-DogBird. To test our framework in more complex settings, with real images, we construct a distribution based on ImageNet ILSVRC-2012 (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, and Fei-Fei, 2015). Recall that we need a setting with a large number of samples relative to the difficulty of the task: if the Real World performs well with few samples and few epochs, then we can simulate it in the Ideal World. Thus, we construct a simpler binary classification task out of ImageNet by collapsing classes into the superclasses "hunting dog" and "bird." This is a roughly balanced task with 155K total images.

5.3.2 Methodology

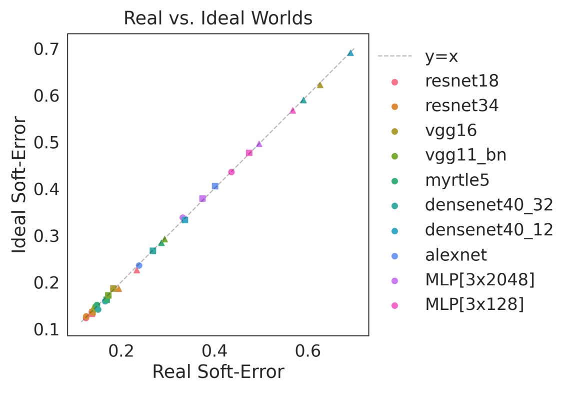
For experiments on CIFAR-5m, we exactly simulate the Real and Ideal worlds as described in Section 5.2. That is, for every Real World architecture and optimizer we consider, we construct the corresponding Ideal World by executing the exact same training code, but using fresh samples in each epoch. The rest of the training procedure remains identical, including data-augmentation and learning-rate schedule. For experiments on ImageNet-DogBird, we do not have enough samples to exactly simulate the Ideal World. Instead, we approximate the Ideal World by using the full training set ($N = 155K$) and data-augmentation. Formally, this corresponds to approximating $\text{Train}_{\mathcal{F}}(\mathcal{D}, \infty, t)$ by $\text{Train}_{\mathcal{F}}(\mathcal{D}, 155K, t)$. In practice, we train the Real World on $n = 10K$ samples for 120 epochs, so we can approximate this with less than 8 epochs on the full $155K$ train set. Since we train with data augmentation (crop+resize+flip), each of the 8 repetitions of each sample will undergo different random augmentations, and thus this plausibly approximates fresh samples.

Stopping time. We stop both Real and Ideal World training when the Real World reaches a small value of train error (which we set as 1% in all experiments). This stopping condition is necessary, as described in Section 5.2. Thus, for experiments which report test performance "at the end of training", this refers to either when the target number of epochs is reached, or when Real World training has converged ($< 1\%$ train error). We always compare Real and Ideal Worlds after the exact same number of train iterations.

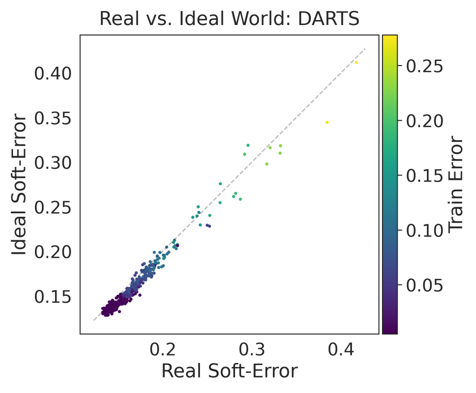
5.4 Main Experiments

We now give evidence to support our main experimental claim, that the bootstrap error ε is often small for realistic settings in deep learning for image classification. In all experiments in this section, we instantiate the same model and training procedure in the Real and Ideal Worlds, and observe that the test soft-error is close at the end of training. Full experimental details are in Appendix C.4.2.

CIFAR-5m. In Figure 5.2a we consider a variety of standard architectures on CIFAR-5m, from fully-connected nets to modern convnets. In the Real World, we train these architectures with SGD on $n = 50K$ samples from CIFAR-5m, for 100 total epochs, with varying initial learning rates.



(a) Standard architectures.



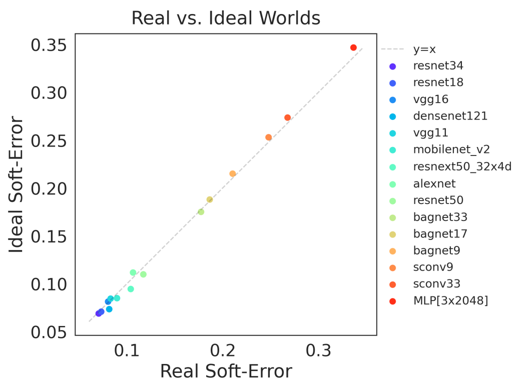
(b) Random DARTS architectures.

Figure 5.2: Real vs Ideal World: CIFAR-5m. SGD with 50K samples. (a): Varying learning-rates 0.1 (●), 0.01 (■), 0.001 (▲). (b): Random architectures from DARTS space (Liu et al., 2019).

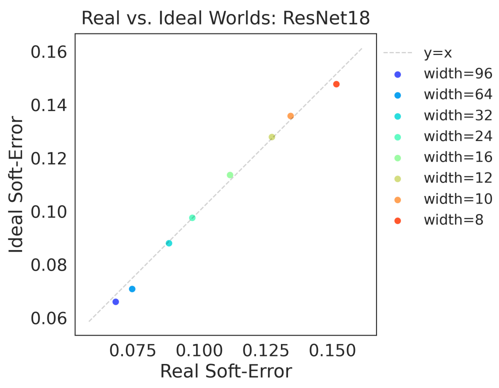
We then construct the corresponding Ideal Worlds for each architecture and learning rate, trained in the same way with fresh samples each epoch. Figure 5.2a shows the test soft-error of the trained classifiers in the Real and Ideal Worlds at the end of training. Observe that test performance is very close in Real and Ideal worlds, although the Ideal World sees $100\times$ unique samples during training.

To test our framework for more diverse architectures, we also sample 500 random architectures from the DARTS search space (Liu, Simonyan, and Yang, 2019). These are deep convnets of varying width and depth, which range in size from 70k to 5.5 million parameters. Figure 5.2b shows the Real and Ideal World test performance at the end of training— these are often within 3%.

ImageNet: DogBird. We now test various ImageNet architectures on ImageNet-DogBird. The Real World models are trained with SGD on $n = 10K$ samples with standard ImageNet data augmentation. We approximate the Ideal World by training on $155K$ samples as described in Section 5.3.2. Figure 5.3a plots the Real vs. Ideal World test error at the end of training, for various architectures. Figure 5.3b shows this for ResNet-18s of varying widths.



(a) Standard architectures.



(b) ResNet-18s of varying width.

Figure 5.3: ImageNet-DogBird. Real World models trained on 10K samples.

5.5 Deep Phenomena through the Bootstrap Lens

Here we show that our Deep Bootstrap framework can be insightful to study phenomena and design choices in deep learning. For example, many effects in the Real World can be seen through their corresponding effects in the Ideal World. Full details for experiments provided in Appendix C.4.

Model Selection in the Over- and Under-parameterized Regimes. Much of theoretical work in deep learning focuses on overparameterized networks, which are large enough to fit their train sets. However, in modern practice, state-of-the-art networks can be either over or *under*-parameterized, depending on the scale of data. For example, SOTA models on 300 million JFT images or 1 billion Instagram images are underfitting, due to the massive size of the train set (Sun, Shrivastava, Singh, and Gupta, 2017; Mahajan, Girshick, Ramanathan, He, Paluri, Li, Bharambe, and van der Maaten, 2018). In NLP, modern models such as GPT-3 and T5 are trained on massive internet-text datasets, and so are solidly in the underparameterized regime (Kaplan et al., 2020; Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, Neelakantan, Shyam, Sastry, Askell, Agarwal, Herbert-Voss, Krueger, Henighan, Child, Ramesh, Ziegler, Wu, Winter, Hesse, Chen, Sigler, Litwin, Gray, Chess, Clark, Berner, McCandlish, Radford, Sutskever, and Amodei, 2020b; Raffel, Shazeer, Roberts, Lee, Narang, Matena, Zhou, Li, and Liu, 2019). We highlight one surprising aspect of this situation:

The same techniques (architectures and training methods)
are used in practice in both over- and under-parameterized regimes.

For example, ResNet-101 is competitive both on 1 billion images of Instagram (when it is underparameterized) and on 50k images of CIFAR-10 (when it is overparameterized). This observation was made recently in Bornschein, Visin, and Osindero (2020) for overparameterized architectures, and is also consistent with the conclusions of Rosenfeld, Rosenfeld, Belinkov, and Shavit (2020). It is apriori surprising that the same architectures do well in both over and underparameterized regimes, since there are very different considerations in each regime. In the overparameterized regime, architecture matters for generalization reasons: there are many ways to fit the train set, and some architectures lead SGD to minima that generalize better. In the underparameterized regime, architecture matters for purely optimization reasons: all models will have small generalization gap with 1 billion+ samples, but we seek models which are capable of reaching low values of test loss, and which do so quickly (with few optimization steps). Thus, it should be surprising that in practice, we use similar architectures in both regimes.

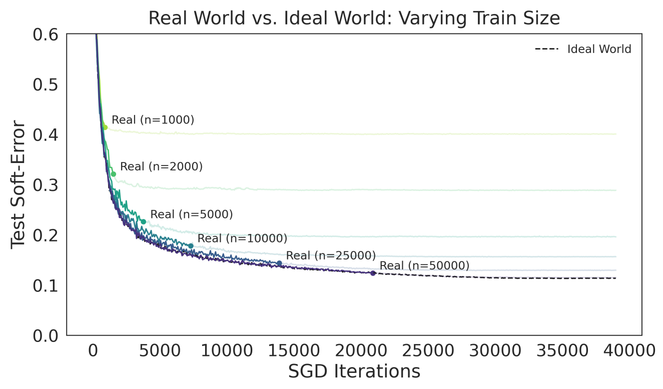
Our work suggests that these phenomena are closely related: If the boostrap error is small, then we should expect that architectures which optimize well in the infinite-data (underparameterized) regime also generalize well in the finite-data (overparameterized) regime. This unifies the two apriori different principles guiding model-selection in over and under-parameterized regimes, and helps understand why the same architectures are used in both regimes.

Implicit Bias via Explicit Optimization. Much recent theoretical work has focused on the *implicit bias* of gradient descent (e.g. Neyshabur, Tomioka, and Srebro (2015a); Soudry et al. (2018);

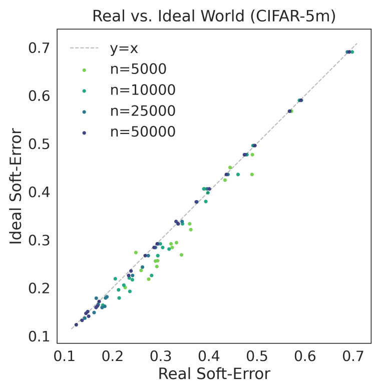
Gunasekar, Lee, Soudry, and Srebro (2018b); Gunasekar et al. (2018a); Ji and Telgarsky (2019); Chizat and Bach (2020)). For overparameterized networks, there are many minima of the empirical loss, some which have low test error and others which have high test error. Thus studying why interpolating networks generalize amounts to studying why SGD is "biased" towards finding empirical minima with low population loss. Our framework suggests an alternate perspective: instead of directly trying to characterize which empirical minima SGD reaches, it may be sufficient to study why SGD optimizes quickly on the population loss. That is, instead of studying implicit bias of optimization on the empirical loss, we could study explicit properties of optimization on the population loss.

The following experiment highlights this approach. Consider the D-CONV and D-FC architectures introduced recently by Neyshabur (2020). D-CONV is a deep convolutional network and D-FC is its fully-connected counterpart: an MLP which subsumes the convnet in expressive capacity. That is, D-FC is capable of representing all functions that D-CONV can represent, since it replaces all conv layers with fully-connected layers and unties all the weights. Both networks reach close to 0 train error on 50K samples from CIFAR-5m, but the convnet generalizes much better. The traditional explanation for this is that the "implicit bias" of SGD biases the convnet to a better-generalizing minima than the MLP. We show that, in fact, this generalization is captured by the fact that D-CONV optimizes much faster on the population loss than D-FC. Figure 5.5c shows the test and train errors of both networks when trained on 50K samples from CIFAR-5m, in the Real and Ideal Worlds. Observe that the Real and Ideal world test performances are nearly identical.

Sample Size. In Figure 5.4, we consider the effect of varying the train set size in the Real World. Note that in this case, the Ideal World does not change. There are two effects of increasing n: (1) The stopping time extends— Real World training continues for longer before converging. And (2) the bootstrap error decreases. Of these, (1) is the dominant effect. Figure 5.4a illustrates this behavior in detail by considering a single model: ResNet-18 on CIFAR-5m. We plot the Ideal World



(a) ResNet-18, varying samples.

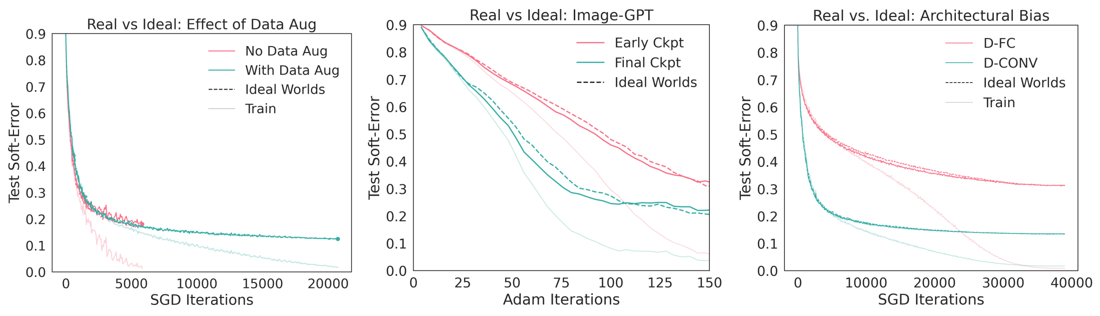


(b) All architectures, varying samples.

Figure 5.4: Effect of Sample Size.

behavior of ResNet-18, as well as different Real Worlds for varying n. All Real Worlds are stopped when they reach $< 1\%$ train error, as we do throughout this work. After this point their test performance is essentially flat (shown as faded lines). However, until this stopping point, all Real Worlds are roughly close to the Ideal World, becoming closer with larger n. These learning curves are representative of most architectures in our experiments. Figure 5.4b shows the same architectures of Figure 5.2a, trained on various sizes of train sets from CIFAR-5m. The Real and Ideal worlds may deviate from each other at small n, but become close for realistically large values of n.

Data Augmentation. Data augmentation in the Ideal World corresponds to randomly augmenting each fresh sample before training on it (as opposed to re-using the same sample for multiple augmentations). There are 3 potential effects of data augmentation in our framework: (1) it can affect the Ideal World optimization, (2) it can affect the bootstrap gap, and (3) it can affect the Real World stopping time (time until training converges). We find that the dominant factors are (1) and (3), though data augmentation does typically reduce the bootstrap gap as well. Figure 5.5a shows the effect of data augmentation on ResNet-18 for CIFAR-5m. In this case, data augmentation does not change the Ideal World much, but it extends the time until the Real World training converges. This view suggests that good data augmentations should (1) not hurt optimization in the



(a) Data Aug: ResNet-18. **(b)** Pretrain: Image-GPT ($n = 2K$). **(c)** Implicit Bias.

Figure 5.5: Deep Phenomena in Real vs. Ideal Worlds.

Ideal World (i.e., not destroy true samples much), and (2) obstruct optimization in the Real World (so the Real World can improve for longer before converging). This is aligned with the "affinity" and "diversity" view of data augmentations in Gontijo-Lopes, Smullin, Cubuk, and Dyer (2020). See Appendix C.2.3 for more figures, including examples where data augmentation hurts the Ideal World.

Pretraining. Figure 5.5b shows the effect of pretraining for Image-GPT (Chen, Radford, Child, Wu, Jun, Luan, and Sutskever, 2020), a transformer pretrained for generative modeling on ImageNet. We fine-tune iGPT-S on 2K samples of CIFAR-10 (not CIFAR-5m, since we have enough samples in this case) and compare initializing from an early checkpoint vs. the final pretrained model. The fully-pretrained model generalizes better in the Real World, and also optimizes faster in the Ideal World. Additional experiments including ImageNet-pretrained Vision Transformers (Dosovitskiy et al., 2021) are in Appendix C.2.5.

Random Labels. Our approach of comparing Real and Ideal worlds also captures generalization in the random-label experiment of Zhang et al. (2017b). Specifically, if we train on a distribution with purely random labels, both Real and Ideal world models will have trivial test error.

5.6 Differences between the Worlds

In our framework, we only compare the test soft-error of models in the Real and Ideal worlds. We do not claim these models are close in all respects— in fact, this is not true. For example, Figure 5.6 shows the same ResNet-18s trained in the Introduction (Figure 6.1), measuring three different metrics in both worlds. Notably, the test *loss* diverges drastically between the Real and Ideal worlds, although the test soft-error (and to a lesser extent, test error) remains close. This is because training to convergence in the Real World will cause the network weights to grow unboundedly, and the softmax distribution to concentrate (on both train and test). In contrast, training in the Ideal World will generally not cause weights to diverge, and the softmax will remain diffuse. This phenomenon also means that the Error and Soft-Error are close in the Real World, but can be slightly different in the Ideal World, which is consistent with our experiments.

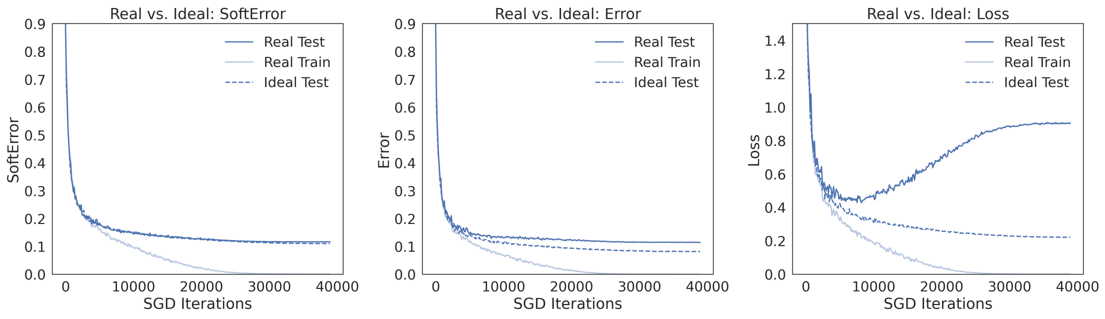


Figure 5.6: SoftError vs. Error vs. Loss: ResNet-18.

5.7 Conclusion and Discussion

We propose the Deep Bootstrap framework for understanding generalization in deep learning. Our approach is to compare the Real World, where optimizers take steps on the empirical loss, to an Ideal World, where optimizers have infinite data and take steps on the population loss. We find that in modern settings, the test performance of models is close between these worlds. This establishes a new connection between the fields of generalization and online learning: models which learn quickly (online) also generalize well (offline). Our framework thus provides a new lens on deep phenomena, and lays a promising route towards theoretically understanding generalization in deep learning.

Limitations. Our work takes first steps towards characterizing the bootstrap error ε, but fully understanding this, including its dependence on problem parameters $(n, \mathcal{D}, \mathcal{F}, t)$, is an important area for future study. The bootstrap error is not universally small for all models and learning tasks: for example, we found the gap was larger at limited sample sizes and without data augmentation. Moreover, it can be large in simple settings like linear regression (Appendix C.1), or settings when the Real World test error is non-monotonic (e.g. due to epoch double-decent (Nakkiran et al., 2020)). Nevertheless, the gap appears to be small in realistic deep learning settings, and we hope that future work can help understand why.

6
Distributional Generalization

We now introduce a new notion of generalization— Distributional Generalization— which roughly states that outputs of a classifier at train and test time are close *as distributions*, as opposed to close in just their average error. For example, if we mislabel 30% of dogs as cats in the train set of CIFAR-10, then a ResNet trained to interpolation will in fact mislabel roughly 30% of dogs as cats on the *test set* as well, while leaving other classes unaffected. This behavior is not captured by classical generalization, which would only consider the average error and not the distribution of errors over the input domain. Our formal conjectures, which are much more general than this example, characterize the form of distributional generalization that can be expected in terms of problem parameters: model architecture, training procedure, number of samples, and data distribution. We give empirical evidence for these conjectures across a variety of domains in machine learning, including neural networks, kernel machines, and decision trees. Our results thus advance our empirical understanding of interpolating classifiers.

6.1 Introduction

In learning theory, when we study how well a classifier "generalizes", we usually consider a single metric – its test error (Shalev-Shwartz and Ben-David, 2014). However, there could be many different classifiers with the same test error that differ substantially in, say, the subgroups of inputs on which they make errors or in the features they use to attain this performance. Reducing classifiers to a single number misses these rich aspects of their behavior.

In this work, we propose formally studying the entire *joint distribution* of classifier inputs and outputs. That is, the distribution $(x, f(x))$ for samples from the distribution $x \sim D$ for a classifier $f(x)$. This distribution reveals many structural properties of the classifier beyond test error (such as *where* the errors occur). In fact, we discover new behaviors of modern classifiers that can only be understood in this framework. As an example, consider the following experiment (Figure 6.1).

Experiment 1. *Consider a binary classification version of CIFAR-10, where CIFAR-10 images x have binary labels* `Animal/Object`. *Take 50K samples from this distribution as a train set, but apply the following label noise: flip the label of cats to* `Object` *with probability 30%. Now train a WideResNet f to 0 train error on this train set. How does the trained classifier behave on test samples? Options below:*

(1) The test error is low across all classes, since there is only 3% overall label noise in the train set.

(2) Test error is "spread" across the animal class. After all, the classifier is not explicitly told what a cat or a dog is, just that they are all animals.

(3) The classifier misclassifies roughly 30% of test cats as "objects", but all other animals are largely unaffected.

The reality is closest to option (3) as shown in Figure 6.1. The left panel shows the joint density of train inputs x with train labels `Object/Animal`. Since the classifier is interpolating, the classifier outputs on the train set are identical to the left panel. The right panel shows the *classifier predictions* $f(x)$ on *test inputs x*.

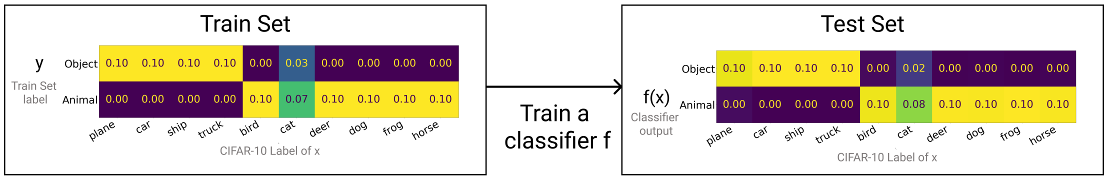


Figure 6.1: The setup and result of Experiment 1. The CIFAR-10 train set is labeled as either Animals or Objects, with label noise affecting only cats. A WideResNet-28-10 is then trained to 0 train error on this train set, and evaluated on the test set.

There are several notable things about this experiment. First, the error is *localized* to cats in the test set as it was in the train set, even though no explicit cat labels were provided. The interpolating model is thus sensitive to subgroup-structures in the distribution. Second, the *amount* of error on the cat class is close to the noise applied on the train set. Thus, the behavior of the classifier on the train set *generalizes* to the test set in a stronger sense than just average error. Specifically, when *conditioned on a subgroup* (cat), the *distribution* of the true labels is close to that of the classifier outputs. Third, this is not the behavior of the Bayes-optimal classifier, which would always output the maximum-likelihood label instead of reproducing the noise in the distribution. The network is thus behaving poorly from the perspective of Bayes-optimality, but behaving well in a certain distributional sense (which we will formalize soon).

Now, consider a seemingly unrelated experimental observation. Take an AlexNet trained on ImageNet, a 1000-way classification problem with 116 varieties of dogs. AlexNet only achieves 56.5% test accuracy on ImageNet. However, it at least classifies most dogs as *some* variety of dog (with 98.4% accuracy), though it may mistake the exact breed.

In this work, we show that both of these experiments are examples of the same underlying phenomenon. We empirically show that for an interpolating classifier, its classification outputs are close in distribution to the true labels — even when conditioned on many subsets of the domain. For example, in Figure 1, the distribution of $p(f(x)|x = \text{cat})$ is close to the true label distribution of $p(y|x = \text{cat})$. We propose a formal conjecture (Feature Calibration), that predicts which subgroups of the domain can be conditioned on for the above distributional closeness to hold.

We emphasize that these experimental behaviors could not have been captured solely by looking at average test error, as is done in the classical theory of generalization. In fact, they are special cases of a new kind of generalization, which we call "Distributional Generalization".

Informally, Distributional Generalization states that the outputs of classifiers f on their train sets and test sets are close *as distributions* (as opposed to close in just error). That is, the following joint distributions[*] are close:

$$(x, f(x))_{x \sim \text{TestSet}} \approx (x, f(x))_{x \sim \text{TrainSet}} \tag{6.1}$$

The remainder of this paper is devoted to making the above statement precise, and empirically checking its validity on real-world tasks. Specifically, we want to formally define the notion of approximation ($\approx$), and understand how it depends on the problem parameters (the type of classifier, number of train samples, etc). We focus primarily on interpolating methods, where we formalize Equation (6.1) through our Feature Calibration Conjecture.

6.1.1 Distributional Generalization

We now present our notion of "Distributional Generalization", and relate it to the classical theory of generalization. A trained model f obeys classical generalization (with respect to test error) if its error on the train set is close to its error on the test distribution. We first re-write this in a form better suited for our extension.

Classical Generalization: *Let f be a trained classifier. Then f generalizes if:*

$$\mathop{\mathbb{E}}_{\substack{x \sim \mathit{TrainSet} \\ \hat{y} \leftarrow f(x)}} [\mathbb{1}\{\hat{y} \neq y(x)\}] \approx \mathop{\mathbb{E}}_{\substack{x \sim \mathit{TestSet} \\ \hat{y} \leftarrow f(x)}} [\mathbb{1}\{\hat{y} \neq y(x)\}] \tag{6.2}$$

[*]These distributions also include the randomness in sampling the train and test sets, and in training the classifier, as we define more precisely in Section 6.2.2.

Above, $y(x)$ is the true class of x and $\widehat{y}$ is the predicted class. The LHS of Equation 6.2 is the train error of f, and the RHS is the test error. Crucially, both sides of Equation 6.2 are expectations of the same function ($T_{\text{err}}(x, \widehat{y}) := 1\{\widehat{y} \neq y(x)\}$) under different distributions. The LHS of Equation 6.2 is the expectation of T_{err} under the "Train Distribution" $\mathcal{D}_{\text{tr}}$, which is the distribution over $(x, \widehat{y})$ given by sampling a train point x along with its classifier-label $f(x)$. Similarly, the RHS is under the "Test Distribution" $\mathcal{D}_{\text{te}}$, which is this same construction over the test set. These two distributions are the central objects in our study, and are defined formally in Section 6.2.2. We can now introduce Distributional Generalization, which is a property of trained classifiers. It is parameterized by a set of bounded functions ("tests"): $\mathcal{T} \subseteq \{T : \mathcal{X} \times \mathcal{Y} \to [0, 1]\}$.

Distributional Generalization: *Let f be a trained classifier. Then f satisfies Distributional Generalization with respect to tests $\mathcal{T}$ if:*

$$\forall T \in \mathcal{T} : \quad \underset{\substack{x \sim \mathit{TrainSet} \\ \widehat{y} \leftarrow f(x)}}{\mathbb{E}} [T(x, \widehat{y})] \approx \underset{\substack{x \sim \mathit{TestSet} \\ \widehat{y} \leftarrow f(x)}}{\mathbb{E}} [T(x, \widehat{y})] \tag{6.3}$$

We write this property as $\mathcal{D}_{\text{tr}} \approx^{\mathcal{T}} \mathcal{D}_{\text{te}}$. This states that the train and test distribution have similar expectations for all functions in the family $\mathcal{T}$. For the singleton set $\mathcal{T} = \{T_{\text{err}}\}$, this is equivalent to classical generalization, but it may hold for much larger sets $\mathcal{T}$. For example in Experiment 1, the train and test distributions match with respect to the test function "*Fraction of true cats labeled as object*." In fact, we find that the family $\mathcal{T}$ is so large in practice that it is best to think of Distributional Generalization as stating that the distributions $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{te}}$ are close *as distributions*.

This property becomes especially interesting for interpolating classifiers, which fit their train sets exactly. Here, the Train Distribution $(x_i, f(x_i))$ is exactly equal[†] to the original distribution $(x, y) \sim \mathcal{D}$, since $f(x_i) = y_i$ on the train set. In this case, distributional generalization claims that

[†]The formal definition of Train Distribution, in Section 6.2.2, includes the randomness of sampling the train set as well. We consider a fixed train set in the Introduction for sake of exposition.

the output distribution $(x, f(x))$ of the model on test samples is close to the *true* distribution (x, y). The following conjecture specializes Distributional Generalization to interpolating classifiers, and will be the main focus of our work.

Interpolating Indistinguishability Meta-Conjecture (informal): *For interpolating classifiers f, and a large family $\mathcal{T}$ of test functions, the distributions:*

$$\boxed{(x, f(x))_{x \in \mathit{TestSet}} \;\approx^{\mathcal{T}}\; (x, f(x))_{x \in \mathit{TrainSet}} \;\equiv\; (x, y)_{x,y \sim \mathcal{D}}} \tag{6.4}$$

This is a "meta-conjecture", which becomes a concrete conjecture once the family of tests $\mathcal{T}$ is specified. One of the main contributions of our work is formally stating two concrete instances of this conjecture— specifying exactly the family of tests $\mathcal{T}$ and their dependence on problem parameters (the distribution, model family, training procedure, etc). It captures behaviors far more general than Experiment 1, and empirically applies across a variety of natural settings in machine learning.

6.1.2 Summary of Contributions

We extend the classical framework of generalization by introducing Distributional Generalization, in which the train and test behavior of models are close *as distributions*. Informally, for trained classifiers f, its outputs on the train set $(x, f(x))_{x \in \text{TrainSet}}$ are close in distribution to its outputs on the test set $(x, f(x))_{x \in \text{TestSet}}$, where the form of this closeness depends on specifics of the model, training procedure, and distribution. This notion is more fine-grained than classical generalization, since it considers the entire distribution of model outputs instead of just the test error. In particular, Distributional Generalization can hold even when classical generalization fails: even classifiers with high test error can have good structure in other ways.

We initiate the study of Distributional Generalization across various domains in machine learning. For interpolating classifiers, we state two formal conjectures which predict the form of distributional closeness that can be expected for a given model and task:

1. **Feature Calibration Conjecture** (Section 6.3): The outputs of interpolating classifiers match the statistics of their training distribution, even when conditioned on certain subgroups (given by "distinguishable features").
2. **Agreement Conjecture** (Section 6.4): For two interpolating classifiers of the same type, trained independently on the same distribution, their *agreement probability* with each other on test samples roughly matches their *test accuracy*.

We perform a number of experiments surrounding these conjectures, which reveal new behaviors of standard interpolating classifiers (e.g. ResNets, MLPs, kernels, decision trees). We prove our conjectures for 1-Nearest-Neighbors (Theorem 1), which suggests some form of "locality" as the underlying mechanism. Finally, we discuss extending these results to non-interpolating methods in Section 6.6. Our experiments and conjectures shed new light on the structure of interpolating classifiers, which are extensively studied in recent years yet still poorly understood.

6.1.3 Related Work and Significance

Our work has connections to, and implications for many existing research programs in deep learning, described below.

Implicit Bias and Overparameterization. There has been a long line of recent work towards understanding overparameterized and interpolating methods, since these pose challenges for classical theories of generalization (e.g. Zhang et al. (2017b); Belkin et al. (2018a,b, 2019a); Liang and Rakhlin (2018); Nakkiran et al. (2020); Schapire et al. (1998); Breiman (1995); Soudry et al. (2018); Gunasekar et al. (2018a)). The "implicit bias" program here aims to answer: *Among all models with 0 train error, which model is actually produced by SGD?* Most existing work seeks to characterize the exact implicit bias of models under certain (sometimes strong) assumptions on the model, training method or the data distribution. In contrast, our conjecture applies across many different interpolating models (from neural nets to decision trees) as they would be used in practice, and thus form a sort of "universal implicit bias" of these methods. Moreover, our results place constraints on potential future theories of implicit bias, and guide us towards theories that better capture practice.

Benign Overfitting. Most prior works on interpolating classifiers attempt to explain why training to interpolation "does not harm" the the model. This has been dubbed "benign overfitting" (Bartlett et al., 2020) and "harmless interpolation" (Muthukumar, Vodrahalli, Subramanian, and Sahai, 2020), reflecting the widely-held belief that interpolation does not harm the decision boundary of classifiers. In contrast, we find that interpolation actually does "harm" classifiers, in predictable ways: fitting the label noise on the train set causes similar noise to be reproduced at test time. Our results thus indicate that interpolation can significantly affect the decision boundary of classifiers, and should not be considered a purely "benign" effect.

Classical Generalization and Scaling Limits. Our framework of Distributional Generalization is insightful even to study classical generalization, since it reveals much more about models than just their test error. For example, statistical learning theory attempts to understand if and when models will asymptotically converge to Bayes optimal classifiers, in the limit of large data ("asymptotic consistency" e.g. Shalev-Shwartz and Ben-David (2014); Wasserman (2006)). In deep learning, there are at least two distinct ways to scale model and data to infinity together: the *underparameterized* scaling limit, where data-size $\gg$ model-size always, and the *overparameterized* scaling limit, where data-size $\ll$ model-size always. The underparameterized scaling limit is well-understood: when data is essentially infinite, neural networks will converge to the Bayes-optimal classifier (provided the model-size is large enough, and the optimization is run for long enough, with enough noise to escape local minima). On the other hand, our work suggests that in the *overparameterized* scaling limit, models will *not* converge to the Bayes-optimal classifier. Specifically, our Feature Calibration Conjecture implies that in the limit of large data, interpolating models will approach a *sampler* from the distribution. That is, the limiting model f will be such that the output $f(x)$ is a sample from $p(y|x)$, as opposed to the Bayes-optimal $f^*(x) = \text{argmax}_y\, p(y|x)$. This claim— that overparameterized models do not converge to Bayes-optimal classifiers— is unique to our work as far as we know, and highlights the broad implications of our results.

Locality and Manifold Learning. Our intuition for the behaviors in this work is that they arise due to some form of "locality" of the trained classifiers, in an appropriate embedding space. For example, the behavior observed in Experiment 1 would be consistent with that of a 1-Nearest-Neighbor classifier in a embedding that separates the CIFAR-10 classes well. This intuition that classifiers learn good embeddings is present in various forms in the literature, for example: the so-called called "manifold hypothesis," that natural data lie on a low-dimensional manifold (e.g. Narayanan and Mitter (2010); Sharma and Kaplan (2020)), as well as works on local stiffness of the loss landscape (Fort, Nowak, Jastrzebski, and Narayanan, 2019b), and works showing that over-parameterized neural networks can learn hidden low-dimensional structure in high-dimensional settings (Gerace, Loureiro, Krzakala, Mézard, and Zdeborová, 2020; Bach, 2017; Chizat and Bach, 2020). It is open to more formally understand connections between our work and the above.

Label Noise. Our conjectures also describe the behavior of neural networks under label noise, which has been empirically and theoretically studied in the past, though not formally characterized before (Zhang et al., 2017b; Belkin et al., 2018b; Rolnick, Veit, Belongie, and Shavit, 2017; Natarajan, Dhillon, Ravikumar, and Tewari, 2013; Thulasidasan, Bhattacharya, Bilmes, Chennupati, and Mohd-Yusof, 2019; Ziyin, Chen, Wang, Liang, Salakhutdinov, Morency, and Ueda, 2020; Chatterji and Long, 2020). Prior works have noticed that vanilla interpolating networks are sensitive to label noise (e.g. Figure 1 in Zhang et al. (2017b), and Belkin et al. (2018b)), and there are many works on making networks more robust to label noise via modifications to the training procedure or objective (Rolnick et al., 2017; Natarajan et al., 2013; Thulasidasan et al., 2019; Ziyin et al., 2020). In contrast, we claim this sensitivity to label noise is not necessarily a problem to be fixed, but rather a consequence of a stronger property: distributional generalization.

6.2 Preliminaries

Notation. We consider joint distributions $\mathcal{D}$ on $x \in \mathcal{X}$ and discrete $y \in \mathcal{Y} = [k]$. Let $\mathcal{D}^n$ denote n iid samples from $\mathcal{D}$ and $S = \{(x_i, y_i)\}$ denote a train set. Let $\mathcal{F}$ denote the training procedure of a classifier family (including architecture and training algorithm), and let $f \leftarrow \text{Train}_{\mathcal{F}}(S)$ denote training a classifier f on train set S. We consider classifiers which output hard decisions $f : \mathcal{X} \to \mathcal{Y}$. Let $\text{NN}_S(x) = x_i$ denote the nearest neighbor to x in train-set S, with respect to a distance metric d. Our theorems will apply to any distance metric, and so we leave this unspecified. Let $\text{NN}_S^{(y)}(x)$ denote the nearest neighbor estimator itself, that is, $\text{NN}_S^{(y)}(x) := y_i$ where $x_i = \text{NN}_S(x)$.

Experimental Setup. Full experimental details are provided in Appendix D.2. Briefly, we train all classifiers to interpolation unless otherwise specified— that is, to 0 train error. We use standard-practice training techniques for all methods with minor hyperparameter modifications for training to interpolation. In all experiments, we consider only the hard-classification decisions, and not e.g. the softmax probabilities. Neural networks (MLPs and ResNets (He et al., 2016a)) are trained with Stochastic Gradient Descent. Interpolating decision trees are trained using the growth rule from Random Forests (Breiman, 2001), growing until all leafs have a single sample. For kernel classification, we consider both kernel regression on one-hot labels and kernel SVM, with small or 0 values of regularization (which is often optimal, as in Shankar, Fang, Guo, Fridovich-Keil, Ragan-Kelley, Schmidt, and Recht (2020)). Section 6.6 considers non-interpolating versions of the above methods (via early-stopping or regularization).

6.2.1 Distributional Closeness

For two distributions P, Q over $\mathcal{X} \times \mathcal{Y}$, let $P \approx_\varepsilon Q$ denote ε-closeness in total variation distance; that is, $TV(P, Q) = \frac{1}{2}||P - Q||_1 \leq \varepsilon$. Recall that TV-distance has an equivalent variational characterization: For distributions P, Q over $\mathcal{X} \times \mathcal{Y}$, we have

$$TV(P, Q) = \sup_{T:\mathcal{X}\times\mathcal{Y}\to[0,1]} \left| \mathop{\mathbb{E}}_{(x,y)\sim P}[T(x, y)] - \mathop{\mathbb{E}}_{(x,y)\sim Q}[T(x, y)] \right|$$

A "test" (or "distinguisher") here is a function $T : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ which accepts a sample from either distribution, and is intended to classify the sample as either from distribution P or Q. TV distance is then the advantage of the best distinguisher among all bounded tests. More generally, for any family $\mathcal{T} \subseteq \{T : \mathcal{X} \times \mathcal{Y} \to [0, 1]\}$ of tests, we say distributions P and Q are "ε-indistinguishable up to $\mathcal{T}$-tests" if they are close with respect to all tests in class $\mathcal{T}$. That is,

$$P \approx_\varepsilon^{\mathcal{T}} Q \iff \sup_{T\in\mathcal{T}} \left| \mathop{\mathbb{E}}_{(x,y)\sim P}[T(x, y)] - \mathop{\mathbb{E}}_{(x,y)\sim Q}[T(x, y)] \right| \leq \varepsilon \tag{6.5}$$

This notion of closeness is also known as an Integral Probability Metric (Müller, 1997). Throughout this work, we will define specific families of distinguishers $\mathcal{T}$ to characterize the sense in which the output distribution $(x, f(x))$ of classifiers is close to their input distribution $(x, y) \sim \mathcal{D}$. When we write $P \approx Q$, we are making an informal claim in which we mean $P \approx_\varepsilon Q$ for some small but unspecified ε.

6.2.2 Framework for Indistinguishability

Here we setup the formal objects studied in the remainder of the paper. This formal description of Train and Test distributions differs slightly from the informal description in the Introduction, because we want to study the generalization properties of an entire end-to-end training procedure

($\text{Train}_{\mathcal{F}}$), and not just properties of a fixed classifier (f). We thus consider the following three distributions over $\mathcal{X} \times \mathcal{Y}$.

1. Source $\mathcal{D}$: (x, y) where $x, y \sim \mathcal{D}$.
2. Train $\mathcal{D}_{\text{tr}}$: $(x_{\text{tr}}, f(x_{\text{tr}}))$ where $S \sim \mathcal{D}^n, f \leftarrow \text{Train}_{\mathcal{F}}(S)$, and $(x_{\text{tr}}, y_{\text{tr}}) \sim S$
3. Test $\mathcal{D}_{\text{te}}$: $(x, f(x))$ where $S \sim \mathcal{D}^n, f \leftarrow \text{Train}_{\mathcal{F}}(S)$, and $x, y \sim \mathcal{D}$

The **Source Distribution** $\mathcal{D}$ is simply the original distribution. To sample from the **Train Distribution** $\mathcal{D}_{\text{tr}}$, we first sample a train set $S \sim \mathcal{D}^n$, train a classifier f on it, then output $(x_{\text{tr}}, f(x_{\text{tr}}))$ for a random *train point* x_{tr}. That is, $\mathcal{D}_{\text{tr}}$ is the distribution of input and outputs of a trained classifier f on its train set. To sample from the **Test Distribution** $\mathcal{D}_{\text{te}}$, do we this same procedure, but output $(x, f(x))$ for a random *test point* x. That is, the $\mathcal{D}_{\text{te}}$ is the distribution of input and outputs of a trained classifier f at test time. The only difference between the Train Distribution and Test Distribution is that the point x is sampled from the train set or the test set, respectively.[‡]

For interpolating classifiers, $f(x_{\text{tr}}) = y_{\text{tr}}$ on the train set, and so the Source and Train distributions are equivalent:

$$\text{For interpolating classifiers } f\text{:} \quad \mathcal{D} \equiv \mathcal{D}_{\text{tr}} \tag{6.6}$$

Our general thesis is that the Train and Test Distributions are indistinguishable under a variety of test families $\mathcal{T}$. Formally, we argue that for certain families of tests $\mathcal{T}$ and interpolating classifiers $\mathcal{F}$,

$$\text{Indistinguishability Conjecture:} \quad \boxed{\mathcal{D} \equiv \mathcal{D}_{\text{tr}} \approx^{\mathcal{T}}_{\varepsilon} \mathcal{D}_{\text{te}}} \tag{6.7}$$

Sections 6.3 and 6.4 give specific families of tests $\mathcal{T}$ for which these distributions are indistinguishable. The quality of this distributional closeness will depend on details of the classifier family and distribution, in ways which we will specify.

[‡]Technically, these definitions require training a fresh classifier for each sample, using independent train sets. We use this definition because we believe it is natural, although for practical reasons most of our experiments train a single classifier f and evaluate it on the entire train/test set.

6.3 Feature Calibration

We now formally describe the Feature Calibration Conjecture. At a high level, we argue that the distributions $\mathcal{D}_{\text{te}}$ and $\mathcal{D}$ are statistically close for interpolating classifiers if we first "coarsen" the domain of x by some partition $L : \mathcal{X} \to [M]$ in to M parts. That is, for certain partitions L, the following distributions are statistically close:

$$(L(x), f(x))_{x \sim \mathcal{D}} \approx_{\varepsilon} (L(x), y)_{x \sim \mathcal{D}}$$

We think of L as defining subgroups over the domain— for example, $L(x) \in \{\text{dog, cat, horse...}\}$. Then, the above statistical closeness is essentially equivalent to requiring that for all subgroups $\ell \in [M]$, the conditional distribution of classifier output on the subgroup—$p(f(x)|L(x) = \ell)$ — is close to the true conditional distribution: $p(y|L(x) = \ell)$.

The crux of our conjecture lies in defining exactly which subgroups L satisfy this distributional closeness, and quantifying the ε approximation. This is subtle, since it must depend on almost all parameters of the problem. For example, consider a modification to Experiment 1, where we use a fully-connected network (MLP) instead of a ResNet. An MLP cannot properly distinguish cats even when it is actually provided the real CIFAR-10 labels, and so (informally) it has no hope of behaving differently on cats in the setting of Experiment 1, where the cats are not labeled explicitly (See Figure D.3.2 for results with MLPs). Similarly, if we train the ResNet with very few samples from the distribution, the network will be unable to recognize cats. Thus, the allowable partitions must depend on the classifier family and the training method, including the number of samples.

We conjecture that allowable partitions are those which can themselves be learnt to good test performance with an identical training procedure, but trained with the labels of the partition L instead of y. To formalize this, we define a *distinguishable feature*: a partition of the domain $\mathcal{X}$ that is learnable for a given training procedure. Thus, in Experiment 1, the partition into CIFAR-10 classes would be a distinguishable feature for ResNets (trained with SGD with 50K or more samples), but not for MLPs. The definition below depends on the training procedure $\mathcal{F}$, the data distribution $\mathcal{D}$, number of train samples n, and an approximation parameter ε (which we think of as $\varepsilon \approx 0$).

Definition 2 ($(\varepsilon, \mathcal{F}, \mathcal{D}, n)$-Distinguishable Feature). *For a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, number of samples n, training procedure $\mathcal{F}$, and small $\varepsilon \geq 0$, an* $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$-distinguishable feature *is a partition $L : \mathcal{X} \to [M]$ of the domain $\mathcal{X}$ into M parts, such that training a model using $\mathcal{F}$ on n samples labeled by L works to classify L with high test accuracy. Precisely, L is a $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$-distinguishable feature if:*

$$\Pr_{\substack{S=\{(x_i, L(x_i))\}_{x_1,\dots,x_n \sim \mathcal{D}} \\ f \leftarrow \mathrm{Train}_{\mathcal{F}}(S);\ x \sim \mathcal{D}}} [f(x) = L(x)] \geq 1 - \varepsilon$$

This definition depends only on the marginal distribution of $\mathcal{D}$ on x, and not on the label distribution $p_{\mathcal{D}}(y|x)$. To recap, this definition is meant to capture a labeling of the domain $\mathcal{X}$ that is learnable for a given training procedure $\mathcal{F}$. It must depend on the architecture used by $\mathcal{F}$ and number of samples n, since more powerful classifiers can distinguish more features. Note that there could be many distinguishable features for a given setting $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$ — including features not implied by the class label such as the presence of grass in a CIFAR-10 image. Our main conjecture follows.

Conjecture 1 (Feature Calibration). *For all natural distributions $\mathcal{D}$, number of samples n, interpolating training procedures $\mathcal{F}$, and $\varepsilon \geq 0$, the following distributions are statistically close for all $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$-distinguishable features L:*

$$\underset{f \leftarrow \mathrm{Train}_{\mathcal{F}}(\mathcal{D}^n);\ x,y \sim \mathcal{D}}{(L(x), f(x))} \quad \approx_{\varepsilon} \quad \underset{x,y \sim \mathcal{D}}{(L(x), y)} \tag{6.8}$$

or equivalently:

$$\underset{x,\widehat{y}\sim\mathcal{D}_{\text{te}}}{(L(x),\widehat{y})} \quad \approx_\varepsilon \quad \underset{x,y\sim\mathcal{D}}{(L(x),y)} \tag{6.9}$$

This claims that the TV distance between the LHS and RHS of Equation (6.9) is at most ε, where ε is the error of the distinguishable feature (in Definition 2). We claim that this holds *for all* distinguishable features L "automatically" – we simply train a classifier, without specifying any particular partition. The formal statements of Definition 2 and Conjecture 1 may seem somewhat arbitrary, involving many quantifiers over $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$. However, we believe these statements are natural: In addition to extensive experimental evidence in the following section, we also prove that Conjecture 1 is formally true as stated for 1-Nearest-Neighbor classifiers in Theorem 1.

Connection to Indistinguishability. Conjecture 1 can be equivalently phrased as an instantiation of our general Indistinguishably Conjecture: the source distribution $\mathcal{D}$ and test distribution $\mathcal{D}_{\text{te}}$ are "indistinguishable up to L-tests". That is, Conjecture 1 is equivalent to the statement

$$\mathcal{D}_{\text{te}} \approx_\varepsilon^{\mathcal{L}} \mathcal{D} \tag{6.10}$$

where $\mathcal{L}$ is the family of all tests which depend on x only via a distinguishable feature L. That is, $\mathcal{L} := \{(x,y) \mapsto T(L(x),y) : (\varepsilon, \mathcal{F}, \mathcal{D}, n)\text{-distinguishable feature } L \text{ and } T : [M] \times \mathcal{Y} \to [0,1]\}$. In other words, $\mathcal{D}_{\text{te}}$ is indistinguishable from $\mathcal{D}$ to any distinguisher that only sees the input x via a distinguishable feature $L(x)$.

6.3.1 Experiments

We now empirically validate our conjecture in a variety of settings in machine learning, including neural networks, kernel machines, and decision trees. To do so, we begin by considering the sim-

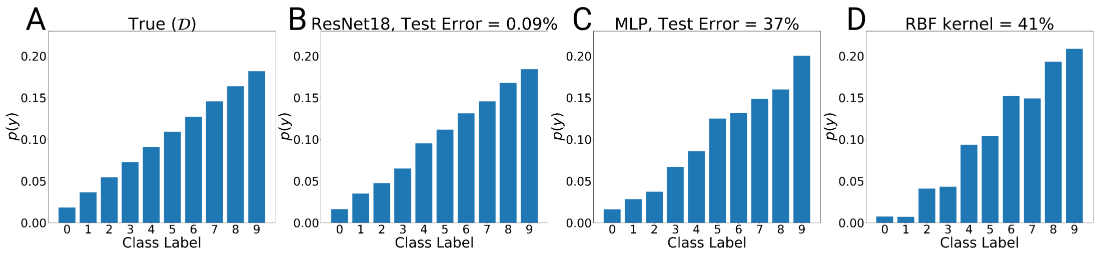


Figure 6.2: Feature Calibration for Constant Partition L: The CIFAR-10 train and test sets are class rebalanced according to (A). Interpolating classifiers are trained on the train set, and we plot the class balance of their outputs on the test set. This roughly matches the class balance of the train set, even for poorly-generalizing classifiers.

plest possible distinguishable feature, and progressively consider more complex ones. Each of the experimental settings below highlights a different aspect of interpolating classifiers, which may be of independent theoretical or practical interest. We summarize the experiments here; detailed descriptions are provided in Appendix D.3.

Constant Partition: Consider the trivially-distinguishable *constant* feature $L(x) = 0$. Then, Conjecture 1 states that the marginal distribution of class labels for any interpolating classifier $f(x)$ is close to the true marginals $p(y)$. That is, irrespective of the classifier's test accuracy, it outputs the "right" proportion of class labels on the test set, even when there is strong class imbalance.

To show this, we construct a dataset based on CIFAR-10 that has class imbalance. For class $k \in \{0...9\}$, sample $(k+1) \times 500$ images from that class. This will give us a dataset where classes will have marginal distribution $p(y = \ell) \propto \ell + 1$ for classes $\ell \in [10]$, as shown in Figure 6.2. We do this both for the training set and the test set, to keep the distribution $\mathcal{D}$ fixed. We then train a variety of classifiers (MLPs, Kernels, ResNets) to interpolation on this dataset, which have varying levels of test errors (9-41%). The class balance of classifier outputs on the (rebalanced) test set is then close to the class balance on the train set, even for poorly generalizing classifiers. Full experimental details and results are described in Appendix D.3. Note that a 1-nearest neighbors classifier would have this property.

Class Partition: We now consider settings (datasets and models) where the original class labels are a distinguishable feature. For instance, the CIFAR-10 classes are distinguishable by ResNets, and MNIST classes are distinguishable by the RBF kernel. Since the conjecture holds for any arbitrary label distribution $p(y|x)$, we consider many such label distributions and show that, for instance, the joint distributions $(\text{Class}(x), y)$ and $(\text{Class}(x), f(x))$ are close. This includes the setting of Experiments 1 and 2 from the Introduction.

In Figure 6.3, we mislabel class $0 \to 1$ with probability p in the CIFAR-10 train set. This gives us the joint distribution shown in Figure 6.3A. We then train a WideResNet-28-10 on this noisy distribution. Figure 6.3B shows the joint distribution on the test set. Figure 6.3C shows the $(1, 0)$ entry of this matrix as we vary $p \in [0, 1]$. The Bayes optimal classifier for this distribution would behave as a step function (shown in red), and a classifier that obeys Conjecture 1 exactly would follow the diagonal (in green). The actual experiment (in blue) is close to the behavior predicted by Conjecture 1.

In fact, our conjecture holds even for a joint density determined by a random confusion matrix on CIFAR-10. In Figure 6.4, we first generate a random sparse confusion matrix on 10 classes, such that each class is preserved with probability 50% and flipped to one of two other classes with probability 20% and 30% respectively. We then apply label noise with this confusion matrix to the train set, and measure the confusion matrix of the trained classifier on the test set. As expected, the train and test confusion matrices are close, and share the same sparsity pattern.

Figure 6.5 shows a version of this experiment for decision trees on the molecular biology UCI task. The molecular biology task is a 3-way classification problem: to classify the type of a DNA splice junction (donor, acceptor, or neither), given the sequence of DNA (60 bases) surrounding the junction. We add varying amounts of label noise that flips class 2 to class 1 with a certain probability, and we observe that interpolating decision trees reproduce this same structured label noise on the test set. We also demonstrate similar experiments with the Gaussian kernel on MNIST (Figure 6.9), and several other UCI tasks (Appendix D.3).

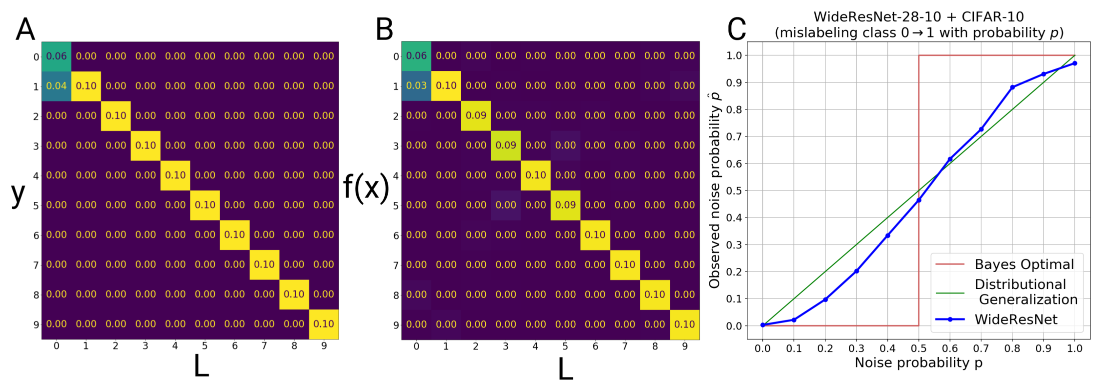


Figure 6.3: Feature Calibration with original classes on CIFAR-10: We train a WRN-28-10 on the CIFAR-10 dataset where we mislabel class $0 \rightarrow 1$ with probability p. (A): Joint density of the distinguishable features L (the original CIFAR-10 class) and the classification task labels y on the train set for noise probability $p = 0.4$. (B): Joint density of the original CIFAR-10 classes L and the network outputs $f(x)$ on the test set. (C): Observed noise probability in the network outputs on the test set (the (1, 0) entry of the matrix in B) for varying noise probabilities p

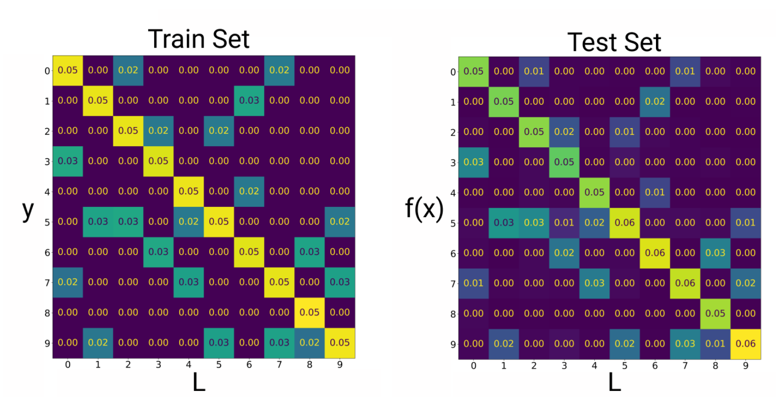


Figure 6.4: Feature Calibration with random confusion matrix on CIFAR-10: Left: Joint density of labels y and original class L on the train set. Right: Joint density of classifier predictions $f(x)$ and original class L on the test set, for a WideResNet28-10 trained to interpolation. These two joint densities are close, as predicted by Conjecture 1.

Multiple features: We now consider a setting where we may have many distinguishable features for a single classification task. The conjecture states that the network should be automatically calibrated for all distinguishable features, even when it is not explicitly provided any information about these features. For this, we use the CelebA dataset (Liu, Luo, Wang, and Tang, 2015), which contains images of celebrities with various labelled binary attributes per-image ("male", "blond hair", etc). Some of these attributes form a distinguishable feature for ResNet50 as they are learnable to high accuracy (Jahandideh, Targhi, and Tahmasbi, 2018). We pick one of the hard attributes as the target classification task, where a ResNet-50 achieves 80% accuracy. Then we confirm that the output distribution is calibrated with respect to the attributes that form distinguishable features. In this setting, the label distribution is deterministic, and not directly dependent on the distinguishable features, unlike the experiments considered before. Yet, as we see in Figure 6.6, the classifier outputs are correctly calibrated for each attribute. Full details of the experiment are described in Appendix D.3.5.

Coarse Partition: Consider AlexNet trained on ImageNet ILSVRC-2012 (Russakovsky et al., 2015), a 1000-class image classification problem including 116 varieties of dogs. The network only achieves 56.5% accuracy on the test set, but it has higher accuracy on coarser label partitions: for example, it will at least classify most dogs as dogs (with 98.4% accuracy), though it may mistake the specific dog variety. In this example, $L(x) \in \{\text{dog}, \text{not-dog}\}$ is the distinguishable feature. Moreover, the network is *calibrated* with respect to dogs: 22.4% of all dogs in ImageNet are Terriers, and indeed, the network classifies 20.9% of all dogs as Terriers (though it has 9% error in which specific dogs it classifies as Terriers). We include similar experiments with ResNets and kernels in Appendix D.3.

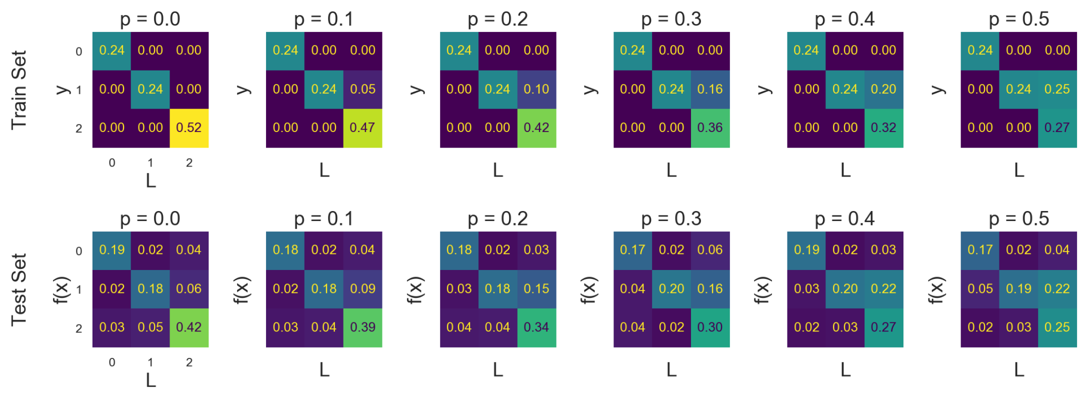


Figure 6.5: Feature Calibration for Decision trees on UCI (molecular biology). We add label noise that takes class 2 to class 1 with probability $p \in [0, 0.5]$. The top row shows the confusion matrix of the true class $L(x)$ vs. the label y on the train set, for varying levels of noise p. The bottom row shows the corresponding confusion matrices of the classifier predictions $f(x)$ on the test set, which closely matches the train set, as predicted by Conjecture 1.

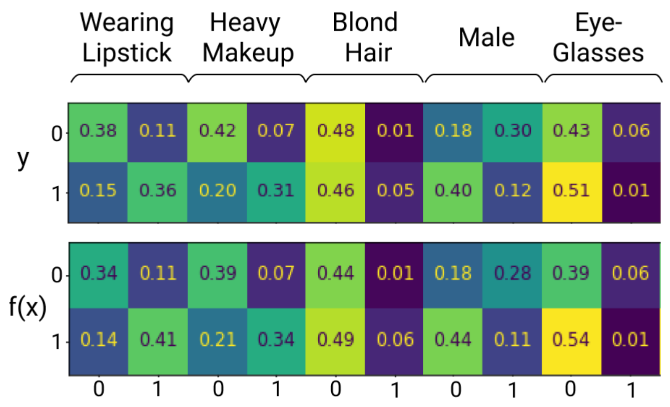


Figure 6.6: Feature Calibration for multiple features on CelebA: We train a ResNet-50 to perform binary classification task on the CelebA dataset. The top row shows the joint distribution of this task label with various other attributes in the dataset. The bottom row shows the same joint distribution for the ResNet-50 outputs on the test set. Note that the network was not given any explicit inputs about these attributes during training.

Model	AlexNet	ResNet50
ImageNet accuracy	0.565	0.761
Accuracy on terriers	0.572	0.775
Accuracy for binary {dog/not-dog}	0.984	0.996
Accuracy on {terrier/not-terrier} among dogs	0.913	0.969
Fraction of real-terriers among dogs	**0.224**	0.224
Fraction of predicted-terriers among dogs	**0.209**	0.229

Table 6.1: **Feature Calibration on ImageNet:** ImageNet classifiers are calibrated with respect to dogs. For example, all classifiers predict terrier for roughly $\sim 22\%$ of all dogs (last row), though they may mistake which specific dogs are terriers. See Table D.2 in the Appendix for more models.

6.3.2 Discussion

Conjecture 1 claims that $\mathcal{D}_{\text{te}}$ is close to $\mathcal{D}$ up to all tests which are *themselves learnable*. That is, if an interpolating method is capable of learning a certain partition of the domain, then it will also produce outputs that are calibrated with respect to this partition, when trained on any problem. This conjecture thus gives a way of quantifying the resolution with which classifiers approximate the source distribution $\mathcal{D}$, via properties of the classification algorithm itself. This is in contrast to many classical ways of quantifying the approximation of density estimators, which rely on *analytic* (rather than *operational*) distributional assumptions (Tsybakov, 2008; Wasserman, 2006).

Proper Scoring Rules. If the loss function used in training is a *strictly-proper scoring rule* such as cross-entropy (Gneiting and Raftery, 2007), then we may expect that in the limit of a large-capacity network and infinite data, training on samples $\{(x_i, y_i)\}$ will yield a good density estimate of $p(y|x)$ at the softmax layer. However, this is not what is happening in our experiments: First, our experiments consider the hard-decisions, not the softmax outputs. Second, we observe Conjecture 1 even in settings without proper scoring rules (e.g. kernel SVM and decision trees).

6.3.3 1-Nearest-Neighbors Connection

Here we show that the 1-nearest neighbor classifier provably satisfies Conjecture 1, under mild assumptions. This is trivially true when the number of train points $n \rightarrow \infty$, such that the train points pack the domain. However, we do not require any such assumptions: the theorem below applies generically to a wide class of distributions, with no assumptions on the ambient dimension of inputs, the underlying metric, or smoothness of the source distribution. All the distributional requirements are captured by the preconditions of Conjecture 1, which require that the feature L is ε-distinguishable to 1-Nearest-Neighbors. The only further assumption is a weak regularity condition: sampling the nearest neighbor train point to a random test point should yield (close to) a uniformly random test point. In the following, $\mathrm{NN}_S(x)$ refers to the nearest neighbor of point x among points in set S.

Theorem 1. *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, and let $n \in \mathbb{N}$ be the number of train samples. Assume the following regularity condition holds: Sampling the nearest neighbor train point to a random test point yields (close to) a uniformly random test point. That is, suppose that for some small $\delta \geq 0$,*

$$\{\mathrm{NN}_S(x)\}_{\substack{S \sim \mathcal{D}^n \\ x \sim \mathcal{D}}} \quad \approx_\delta \quad \{x\}_{x \sim \mathcal{D}} \tag{6.11}$$

Then, Conjecture 1 holds. For all $(\varepsilon, \mathrm{NN}, \mathcal{D}, n)$-distinguishable partitions L, the following distributions are statistically close:

$$\{(y, L(x))\}_{x,y \sim \mathcal{D}} \quad \approx_{\varepsilon+\delta} \quad \{(\mathrm{NN}_S^{(y)}(x), L(x)\}_{\substack{S \sim \mathcal{D}^n \\ x,y \sim \mathcal{D}}} \tag{6.12}$$

Proof of Theorem 1. Recall that L being an $(\varepsilon, \mathrm{NN}, \mathcal{D}, n)$-distinguishable partition means that nearest-neighbors works to classify $L(x)$ from x:

$$\Pr_{\substack{\{x_i,y_i\}\sim\mathcal{D}^n \\ S=\{(x_i,L(x_i))\} \\ x,y\sim\mathcal{D}}} [\mathrm{NN}_S^{(y)}(x) = L(x)] \geq 1-\varepsilon \tag{6.13}$$

Now, we have

$$\{(\mathrm{NN}_S^{(y)}(x), L(x))\}_{\substack{S\sim\mathcal{D}^n \\ x,y\sim\mathcal{D}}} \equiv \{(\widehat{y_i}, L(x))\}_{\substack{S\sim\mathcal{D}^n \\ \widehat{x_i},\widehat{y_i}\leftarrow\mathrm{NN}_S(x) \\ x,y\sim\mathcal{D}}} \tag{6.14}$$

$$\approx_\varepsilon \{(\widehat{y_i}, L(\widehat{x_i}))\}_{\substack{S\sim\mathcal{D}^n \\ \widehat{x_i},\widehat{y_i}\leftarrow\mathrm{NN}_S(x) \\ x,y\sim\mathcal{D}}} \tag{6.15}$$

$$\approx_\delta \{(\widehat{y_i}, L(\widehat{x_i}))\}_{\widehat{x_i},\widehat{y_i}\sim\mathcal{D}} \tag{6.16}$$

Line (6.15) is by distinguishability, since $\Pr[L(x) \neq L(\widehat{x_i})] \leq \varepsilon$. And Line (6.16) is by the regularity condition. □

We view this theorem both as support for our formalism of Conjecture 1, and as evidence that the classifiers we consider in this work have *local* properties similar to 1-Nearest-Neighbors.

Note that Theorem 1 does not hold for the k-nearest neighbor classifier (k-NN), which takes the plurality vote of K neighboring train points. However, it is somewhat more general than 1-NN: for example, it holds for a randomized version of k-NN which, instead of taking the plurality, randomly picks one of the K neighboring train points (potentially weighted) for the test classification.

6.3.4 Pointwise Density Estimation

In fact, we could hope for an even stronger property than Conjecture 1. Consider the familiar example: we mislabel 20% of dogs as cats in the CIFAR-10 training data, and train an interpolating ResNet on this train set. Conjecture 1 predicts that, *on average* over all test dogs, roughly 20% of them are classified as cats. In fact, we may expect this to hold pointwise for each dog: For a single test dog x, if we train a new classifier f (on fresh iid samples from the noisy distribution), then $f(x)$ will be cat roughly 20% of the time. That is, for each test point x, taking an ensemble over independent train sets yields an estimate of the conditional density $p(y|x)$. Informally:

$$\text{With high probability over test } x \sim \mathcal{D} : \quad \Pr_{f \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}^n)}[f(x) = \ell] \approx p(y = \ell | x) \tag{6.17}$$

where the probability on the LHS is over the random sampling of train set, and any randomness in the training procedure. This behavior would be stronger than, and not implied by, Conjecture 1. We give preliminary experiments supporting such a pointwise property in Appendix D.3.7.

6.4 Agreement Property

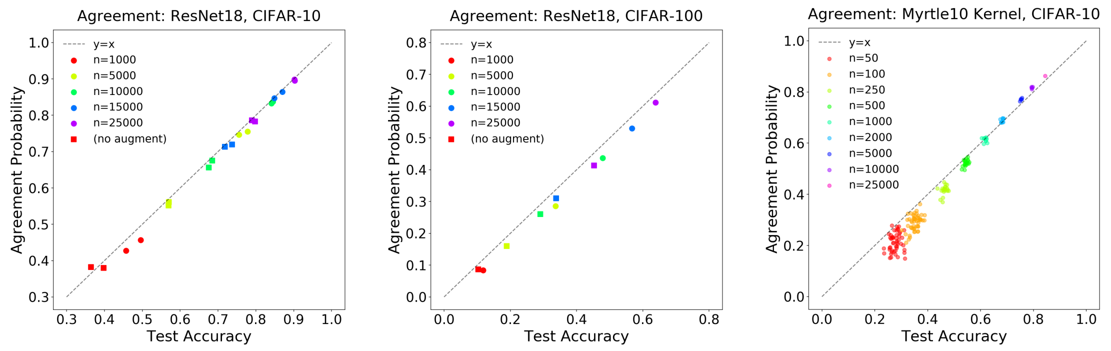
We now present an "agreement property" of various classifiers. This property is independent of the previous section, though both are instantiations of our general indistinguishability conjecture. We claim that, informally, the test accuracy of a classifier is close to the probability that it agrees with an identically-trained classifier on a disjoint train set.

Conjecture 2 (Agreement Property). *For certain classifier families $\mathcal{F}$ and distributions $\mathcal{D}$, the test accuracy of a classifier is close to its* agreement probability *with an independently-trained classifier. That is, let S_1, S_2 be independent train sets sampled from $\mathcal{D}^n$, and let f_1, f_2 be classifiers trained on S_1, S_2 respectively. Then*

$$\Pr_{\substack{S_1 \sim \mathcal{D}^n \\ f_1 \leftarrow \text{Train}_{\mathcal{F}}(S_1) \\ (x,y) \sim \mathcal{D}}}[f_1(x) = y] \approx \Pr_{\substack{S_1, S_2 \sim \mathcal{D}^n \\ f_i \leftarrow \text{Train}_{\mathcal{F}}(S_i) \\ (x,y) \sim \mathcal{D}}}[f_1(x) = f_2(x)] \tag{6.18}$$

Moreover, this holds with high probability over training f_1, f_2: $\Pr_{(x,y)\sim\mathcal{D}}[f_1(x) = y] \approx \Pr_{(x,y)\sim\mathcal{D}}[f_1(x) =$

$f_2(x)]$.



(a) ResNet18 on CIFAR-10.

(b) ResNet18 on CIFAR-100.

(c) Myrtle Kernel on CIFAR-10.

Figure 6.7: Agreement Property on CIFAR-10/100. For two classifiers trained on disjoint train sets, the probability they agree with each other (on the test set) is close to their test accuracy.

The agreement property (Conjecture 2) is surprising for several reasons. First, suppose we have two classifiers f_1, f_2 which were trained on independent train sets, and both achieve test accuracy say 50% on a 10-class problem. That is, they agree with the true label $y(x)$ w.p. 50%. Depending on our intuition, we may expect: (1) They agree with each other much less than they agree with the true label, since each individual classifier is an independently noisy version of the truth, or (2) They agree with each other much more than 50%, since classifiers tend to have "correlated" predictions. However, neither of these are the case in practice.

Second, it may be surprising that the RHS of Equation 6.18 is an estimate of the test error that requires only unlabeled test examples x. This observation is independently interesting, and may be relevant for applications in uncertainty estimation and calibration. Conjecture 2 also provably holds for 1-Nearest-Neighbors in some settings, under stronger assumptions (Theorem 2 in Appendix D.6).

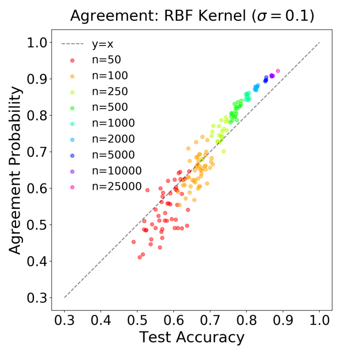
Connection to Indistinguishability. Conjecture 2 is in fact an instantiation of our general indistinguishability conjecture. Informally, we can "swap y for $f_2(x)$" in the LHS of Equation 6.18, since they are indistinguishable. Formally, consider the specific test

$$T_{\text{agree}} : (x, \widehat{y}) \mapsto 1\{f_1(x) = \widehat{y}\} \tag{6.19}$$

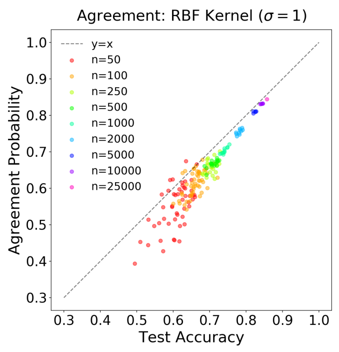
where $f_1 \leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}^n)$. The expectation of this test under the Source Distribution $\mathcal{D}$ is exactly the LHS of Equation 6.18, while the expectation under the Test Distribution $\mathcal{D}_{\text{te}}$ is exactly the RHS. Thus, Conjecture 2 can be equivalently stated as

$$\mathcal{D} \approx^{T_{\text{agree}}} \mathcal{D}_{\text{te}}. \tag{6.20}$$

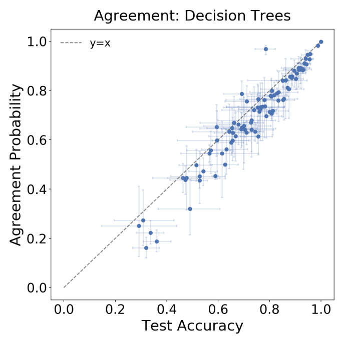
6.4.1 Experiments



(a) RBF on Fashion-MNIST



(b) RBF on Fashion-MNIST



(c) Decision Trees on UCI

Figure 6.8: Agreement Property for RBF and decision trees. For two classifiers trained on disjoint train sets, the probability they agree with each other (on the test set) is close to their test accuracy. For UCI, each point corresponds to one UCI task, and error bars show 95% Clopper-Pearson confidence intervals in estimating population quantities.

In our experiments, we train a pair of classifiers f_1, f_2 on random disjoint subsets of the train set for a given distribution. Both classifiers are otherwise trained identically, using the same architecture, number of train-samples n, and optimizer. We then plot the test error of f_1 against the agreement probability $\Pr_{x \sim \text{TestSet}}[f_1(x) = f_2(x)]$. Figure 6.7 shows experiments with ResNet18 on CIFAR-10 and CIFAR-100, as well as the Myrtle10 kernel from Shankar et al. (2020), with varying number of train samples n. These classifiers are trained with standard-practice training procedures (SGD with standard data-augmentation for ResNets), with no additional hyperparameter tuning. Figure 6.8 shows experiments with the RBF Kernel on Fashion-MNIST, and decision trees on 92 UCI classification tasks. The Agreement Property approximately holds for all pairs of identical classifiers, and continues to hold even for "weak" classifiers (e.g. when f_1, f_2 have high test error). Full experimental details and further experiments are in Appendix D.4.

6.4.2 Potential Mechanisms

We now consider, and refute, several potential mechanisms which could explain the experimental results of Conjecture 2.

Bimodal Samples

A simple model which would exhibit the Agreement Property is the following: Suppose test samples x come in two types: "easy" or "hard." All classifiers get "easy" samples correct, but they output a uniformly random class on "hard" samples. That is, for a fixed x, consider the probability that a freshly-trained classifier gets x correct. "Easy" samples are such that

$$\text{For } x \in \text{EASY:} \quad \Pr_{f \leftarrow \text{Train}(\mathcal{D}^n)}[f(x) = y(x)] = 1$$

while "hard" samples have a uniform distribution on output classes $[K]$:

$$\text{For } x \in \text{HARD:} \quad \Pr_{f \leftarrow \text{Train}(\mathcal{D}^n)}[f(x) = i] = \frac{1}{K} \quad \forall i \in [K]$$

Notice that for HARD samples x, a classifier f_1 agrees with the true label y with exactly the same probability that it agrees with an independent classifier f_2 (because both f_1, f_2 are uniformly random on x). Thus, the agreement property (Conjecture 2) holds exactly under this model. However, this strict decomposition of samples into "easy" and "hard" does not appear to be the case in the experiments (see Appendix D.4.3, Figure D.11).

Pointwise Agreement

We could more generally posit that Conjecture 2 is true because the Agreement Property holds *pointwise* for most test samples x. That is, Equation (6.18) would be implied by:

$$\text{w.h.p. for } (x, y) \sim \mathcal{D} : \quad \Pr_{f_1 \leftarrow \mathrm{Train}(\mathcal{D}^n)}[f_1(x) = y] \approx \Pr_{\substack{f_1 \leftarrow \mathrm{Train}(\mathcal{D}^n) \\ f_2 \leftarrow \mathrm{Train}(\mathcal{D}^n)}}[f_1(x) = f_2(x)] \tag{6.21}$$

This was the case for the EASY/HARD decomposition above, but could be true in more general settings. Equation (6.21) is a "pointwise calibration" property that would allow estimating the probability of making an error on a test point x by simply estimating the probability that two independent classifiers agree on x. However, we find (perhaps surprisingly) that this is not the case. That is, Equation (6.18) holds on average over $(x, y) \sim \mathcal{D}$, but not pointwise for each sample. We give experiments demonstrating this in Appendix D.4.3. Interestingly, 1-nearest neighbors can satisfy the agreement property of Claim 2 without satisfying the "pointwise agreement" of Equation 6.21. It remains an open problem to understand the mechanisms behind Agreement Matching.

6.5 Limitations and Ensembles

The conjectures presented in this work are not fully specified, since they do not exactly specify which classifiers or distributions for which they hold. We experimentally demonstrate instances of these conjectures in various "natural" settings in machine learning, but we do not yet understand which assumptions on the distribution or classifier are required. Some experiments also deviate slightly from the predicted behavior (e.g. the kernel experiments in Figures 6.2 and 6.8). Nevertheless, we believe our conjectures capture the essential aspects of the observed behaviors, at least to first order. It is an important open question to refine these conjectures and better understand their applications and limitations— both theoretically and experimentally.

6.5.1 Ensembles

We could ask if all high-performing interpolating methods used in practice satisfy our conjectures. However, an important family of classifiers which fail our Feature Calibration Conjecture are ensemble methods:

1. Deep ensembles of interpolating neural networks (Lakshminarayanan, Pritzel, and Blundell, 2017).
2. Random forests (i.e. ensembles of interpolating decision trees) (Breiman, 2001).
3. k-nearest neighbors (roughly "ensembles" of 1-Nearest-Neighbors) (Fix and Hodges, 1951).

The pointwise density estimation discussion in Section 6.3.4 sheds some light on these cases. Notice that these are settings where the "base" classifier in the ensemble obeys Feature Calibration, and in particular, acts as an approximate conditional density estimator of $p(y|x)$, as in Section 6.3.4. That is, if individual base classifiers f_i approximately act as samples from

$$f_i(x) \sim p(y|x)$$

then for sufficiently many classifiers $\{f_1, \dots, f_k\}$ trained on independent train sets, the ensembled classifier will act as

$$\text{plurality}(f_1, f_2, \dots, f_k)(x) \approx \operatorname*{argmax}_y p(y|x)$$

Thus, we believe ensembles fail our conjectures because, in taking the plurality vote of base classifiers, they are approximating $\text{argmax}_y p(y|x)$ instead of the conditional density $p(y|x)$ itself. Indeed, in the above examples, we observed that ensemble methods behave much closer to the Bayes-optimal classifier than their underlying base classifiers (especially in settings with label noise).

6.6 Distributional Generalization: Beyond Interpolating Methods

The previous sections have focused primarily on *interpolating* classifiers, which fit their train sets exactly. Here we discuss the behavior of non-interpolating methods, such as early-stopped neural networks and regularized kernel machines, which do not reach 0 train error.

For non-interpolating classifiers, their outputs on the train set $(x, f(x))_{x \sim \text{TrainSet}}$ will *not* match the original distribution $(x, y) \sim \mathcal{D}$. Thus, there is little hope that their outputs on the test set will match the original distribution, and we do not expect the Indistinguishability Conjecture to hold. However, the Distributional Generalization framework does not require interpolation, and we could still expect that the train and test distributions are close ($\mathcal{D}_{\text{tr}} \approx^{\mathcal{T}} \mathcal{D}_{\text{te}}$) for some family of tests $\mathcal{T}$. For example, the following is a possible generalization of Feature Calibration (Conjecture 1).

Conjecture 3 (Generalized Feature Calibration, informal). *For trained classifiers f, the following distributions are statistically close for many partitions L of the domain:*

$$\underset{x_i \sim \text{TrainSet}}{(L(x_i), f(x_i))} \quad \approx \quad \underset{x \sim \text{TestSet}}{(L(x), f(x))} \tag{6.22}$$

We leave unspecified the exact set of partitions L for which this holds— unlike Conjecture 1, where we specified L as the set of all distinguishable features. In this generalized case, we do not yet understand the appropriate notion of "distinguishable feature" [§]. However, we give experimental evidence that suggests some refinement of Conjecture 3 is true.

[§]For example, when considering early-stopped neural networks, it is unclear if the partition L should be distinguishable with respect to the early-stopped network or its fully-trained counterpart.

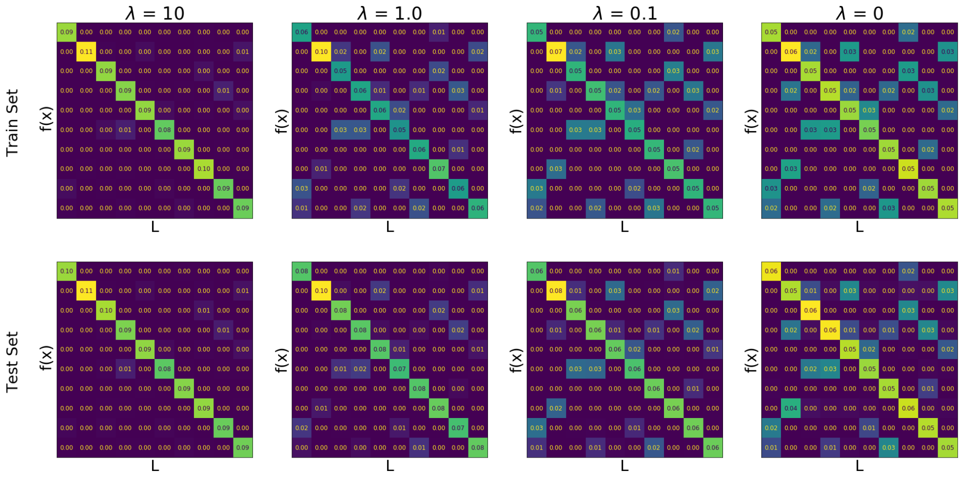


Figure 6.9: Distributional Generalization for Gaussian Kernel on MNIST. We apply label noise from a random sparse confusion to the MNIST train set. We then train a Gaussian Kernel for classification, with varying L_2 regularization λ. The top row shows the confusion matrix of predictions $f(x)$ vs true labels $L(x)$ on the train set, and the bottom row shows the corresponding confusion matrix on the test set. Larger values of regularization prevents the classifier from fitting label noise on the train set, and this behavior is mirrored almost identically on the test set. Note that all classifiers above are trained on the same train set, with the same label noise.

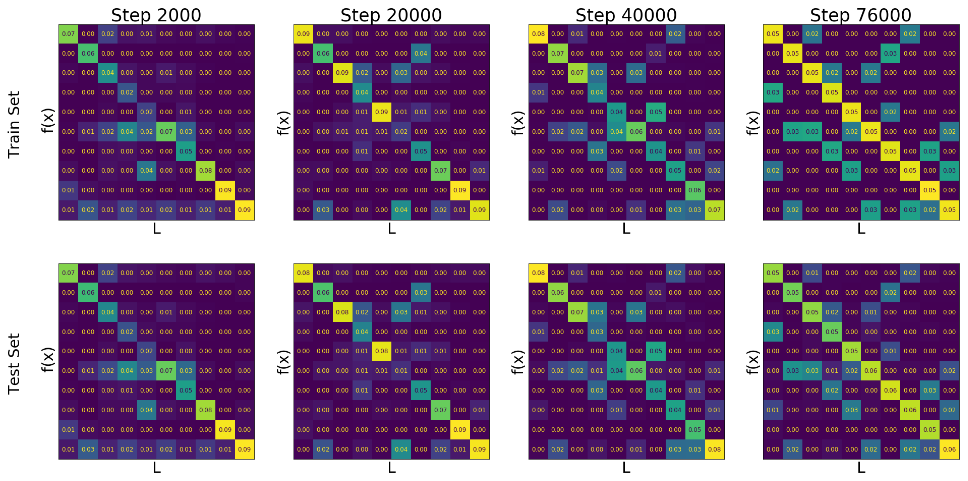


Figure 6.10: Distributional Generalization for WideResNet on CIFAR-10. We apply label noise from a random sparse confusion to the CIFAR-10 train set. We then train a single WideResNet28-10, and measure its predictions on the train and test sets over increasing train time (SGD steps). The top row shows the confusion matrix of predictions $f(x)$ vs true labels $L(x)$ on the train set, and the bottom row shows the corresponding confusion matrix on the test set. As the network is trained for longer, it fits more of the noise on the train set, and this behavior is mirrored almost identically on the test set.

In Figure 6.9 we train Gaussian kernel regression on MNIST, with label noise determined by a random sparse confusion matrix on the train set (analogous to the setting of Figure 6.4; experimental details in Appendix D.2). We vary the amount of ℓ_2 regularization, and plot the confusion matrix of predictions on the train and test sets. With $\lambda = 0$ regularization, the kernel interpolates the noise in the train set exactly, and reproduces this noise on the test set as expected. With higher regularization, the kernel no longer interpolates the train set, but the test and train confusion matrices remain close. That is, regularization prevents the kernel from fitting the noise on both the train and test sets in a similar way. Remarkably, higher regularization yields a classifier closer to Bayes-optimal. Figure 6.10 shows an analogous experiment for neural networks on CIFAR-10, with early-stopping in place of regularization: early in training, neural networks do not fit their train set, but their test and train confusion matrices remain close throughout training. These experiments suggest that Distributional Generalization is a meaningful notion even for non-interpolating classifiers. Formalizing and investigating this further is an interesting area for future study.

6.7 Conclusion and Discussion

In this work, we presented a new set of empirical behaviors of standard interpolating classifiers. We unified these under the framework of Distributional Generalization, which states that outputs of trained classifiers on the test set are "close" in distribution to their outputs on the train set. For interpolating classifiers, we stated several formal conjectures (Conjectures 1 and 2) to characterize the form of distributional closeness that can be expected.

Beyond Test Error. Our work proposes studying the *entire distribution* of classifier outputs on test samples, beyond just its test error. We show that this distribution is often highly structured, and we take steps towards characterizing it. Surprisingly, modern interpolating classifiers appear to satisfy certain forms of distributional generalization "automatically," despite being trained to simply minimize train error. This even holds in cases when satisfying distributional generalization is in conflict with satisfying classical generalization— that is, when a distributionally-generalizing classifier must necessarily have high test error (e.g. Experiment 1). We thus hope that studying distributional generalization will be useful to better understand modern classifiers, and to understand generalization more broadly.

Interpolating vs. Non-interpolating Methods. Our work also suggests that interpolating classifiers should be viewed as conceptually different objects from non-interpolating ones, even if both have the same test error. In particular, an interpolating classifier will match certain aspects of the original distribution, which a non-interpolating classifier will not. This also suggests, informally, that interpolating methods should not be seen as methods which simply "memorize" their training data in a naive way (as in a look up table) – rather this "memorization" strongly influences the classifier's decision boundary (as in 1-Nearest-Neighbors).

6.7.1 Open Questions

Our work raises a number of open questions and connections to other areas. We briefly collect some of them here.

1. As described in the Limitations (Section 6.5), we do not precisely understand the set of distributions and interpolating classifiers for which our conjectures hold. We empirically tested a number of "realistic" settings, but it is open to state formal assumptions defining these settings.
2. It is open to theoretically prove versions of Distributional Generalization for models beyond 1-Nearest-Neighbors. This is most interesting in cases where Distributional Generalization is at odds with classical generalization (e.g. Figure 6.3c).
3. It is open to understand the mechanisms behind the Agreement Property (Section 6.4), theoretically or empirically.

4. In some of our experiments (e.g. Section 6.3.4), ensembling over independent random-initializations had a similar effect to ensembling over independent train sets. This is related to works on deep ensembles (Lakshminarayanan et al., 2017; Fort, Hu, and Lakshminarayanan, 2019a) as well as random forests for conditional density estimation (Meinshausen, 2006; Pospisil and Lee, 2018; Athey, Tibshirani, Wager, et al., 2019). Investigating this further is an interesting area of future work.
5. There are a number of works suggesting "local" behavior of neural networks, and these are somewhat consistent with our locality intuitions in this work. However, it is open to formally understand whether these intuitions are justified in our setting.
6. We give two families of tests $\mathcal{T}$ for which our Interpolating Indistinguishability conjecture (Equation 6.4) empirically holds. This may not be exhaustive – there may be other ways in which the source distribution $\mathcal{D}$ and test distribution $\mathcal{D}_{\text{te}}$ are close. Indeed, we give preliminary experiments for another family of tests, based on student-teacher training, in Appendix D.1. It is open to explore more ways in which Distributional Generalization holds, beyond the tests presented here.

7
Conclusion

In this thesis, we have developed several empirical laws about deep learning "in the wild." Our investigations shed light on existing questions in learning theory (generalization, overparameterization, interpolation), and also revealed entirely new behaviors which required new frameworks to state (Distributional Generalization).

Deep Double Descent taught us that even modern networks may have pathological behavior in a "critical regime", but are well-behaved outside of it. The Deep Bootstrap demonstrated that although we train models on finite train sets in practice, they behave as if we trained them on an infinite stream of samples. And Distributional Generalization implies that although we think we are training *classifiers*, the objects we get are better thought of as *samplers*.

We hope these results will be helpful as conceptual tools: they abstract away the complexity of "deep learning" down to its essential objects, and demonstrate robust behaviors that occur even at this high level of abstraction. Abstracting away all irrelevant details makes our theory both simpler and plausibly more universal: we hope our results hold not only for the neural networks of today, but for whatever "deep learning" means 10+ years from now.

In fact, we hope our results will eventually weave into a general theory of *learning*, beyond just deep learning. This has already happened to some extent: double descent was previously observed to hold in many non-deep learning systems as well (Belkin et al., 2019a). And in Distributional Generalization, the conjectures we derived to explain behaviors of neural networks turned out to also apply to other methods (decision trees and kernels) – revealing new behaviors of these methods. We hope that the Deep Bootstrap Framework will be similarly universal: instead of applying only to deep networks trained via SGD or variants, it may apply generically anytime a "reasonable" online optimizer is used in an offline setting.

It is remarkable that it is even possible to state claims that hold for deep networks just as well as for decision trees— two a priori very different methods, used on very different kinds of tasks. This gives us hope that there exists a unified theory of learning, which captures important behaviors of many modern models – deep or otherwise.

What can we learn, deeply or otherwise?

7.1 Open Questions

Here we give a partial list of open questions that we believe may be relevant to the future of deep learning theory.

1. What distinguishes deep learning from other methods? Why was deep learning so successful in settings where prior methods were not? Are they information-theoretic factors (e.g. sample complexity), computational factors (e.g. optimization time and space), or other factors

(e.g. ability to integrate auxiliary data, representation learning, etc).

2. Is there a "minimal set of assumptions" for deep learning theory? Is there some set of conjectures/laws/assumptions which, if we assume, would explain all other phenomena in deep learning? (Can we do for deep learning theory what one-way-functions did for cryptography theory?)
3. What learning methods would we use if we relax certain constraints: If we had infinite computation time, or infinite space, or infinite samples? Would we still use deep learning in its current form?
4. Is there an "axiomatic definition" of deep learning? E.g. can deep learning be defined as "the unique system which satisfies properties X, Y, Z" for certain values of X, Y, Z? (These properties may be related to online learning, robustness, ability to incorporate auxiliary data, etc.)
5. Can we formally define broader notions of learning or generalization, which capture the observed behavior of large models such as GPT-3 and CLIP? These behaviors go beyond the classical notions of generalization, and even beyond our notion of Distributional Generalization.
6. In deep learning, we often get "more than we asked for." That is, we only minimize an on-distribution error/loss, but we get networks with many other interesting properties (from good internal representations, to unexpected kinds of off-distribution generalization, etc). How should we think of these auxiliary behaviors, and how can we predict them?
7. What is *representation learning* formally? Why do trained deep networks have structured internal representations, and what is this structure?
8. Why can deep learning systems often be effectively *composed* with other learning systems (deep and non-deep)?
9. The "definitional question": Can we formally define "deep learning" in a way that captures all current and future evolutions of the term? (But not too broad a definition to be trivial).

10. The "natural distributions question": Can we formally understand the set of tasks for which deep learning systems "work"?

A
Additional Works

During the course of the PhD, the author completed several other works in deep learning which were not included in this thesis. We briefly list them below, organized by topic.

A.1 On Adversarial Examples

- Adversarial robustness may be at odds with simplicity. Nakkiran (2019c).
- Computational limitations in robust classification and win-win results. Degwekar, Nakkiran, and Vaikuntanathan (2019).
- Adversarial examples are just bugs, too. Nakkiran (2019a).

A.2 On Dynamics of SGD and Representation Learning

- SGD on neural networks learns functions of increasing complexity. Nakkiran et al. (2019).
- Revisiting model stitching to compare neural representations. Bansal, Nakkiran, and Barak (2021).

A.3 On Theoretical Models

- More data can hurt for linear regression: Sample-wise double descent. Nakkiran (2019b).
- Optimal regularization can mitigate double descent. Nakkiran, Venkat, Kakade, and Ma (2021b).
- Learning rate annealing can provably help generalization, even for convex problems. Nakkiran (2020).

B

Deep Double Descent

B.1 Experimental Details

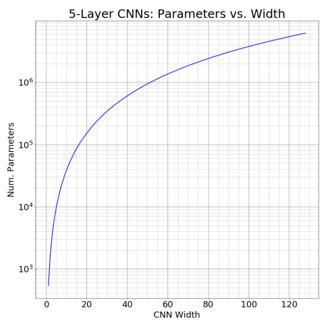
B.1.1 Models

We use the following families of architectures. The PyTorch Paszke, Gross, Chintala, Chanan, Yang, DeVito, Lin, Desmaison, Antiga, and Lerer (2017) specification of our ResNets and CNNs are available at `https://gitlab.com/harvard-machine-learning/double-descent/tree/master`.

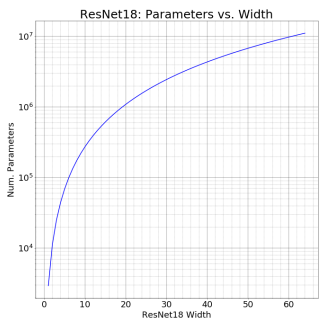
ResNets. We define a family of ResNet18s of increasing size as follows. We follow the Preactivation ResNet18 architecture of He et al. (2016b), using 4 ResNet blocks, each consisting of two BatchNorm-ReLU-Convolution layers. The layer widths for the 4 blocks are $[k, 2k, 4k, 8k]$ for varying $k \in \mathbb{N}$ and the strides are [1, 2, 2, 2]. The standard ResNet18 corresponds to $k = 64$ convolutional channels in the first layer. The scaling of model size with k is shown in Figure B.1b. Our implementation is adapted from `https://github.com/kuangliu/pytorch-cifar`.

Standard CNNs. We consider a simple family of 5-layer CNNs, with four Conv-BatchNorm-ReLU-MaxPool layers and a fully-connected output layer. We scale the four convolutional layer widths as $[k, 2k, 4k, 8k]$. The MaxPool is [1, 2, 2, 8]. For all the convolution layers, the kernel size = 3, stride = 1 and padding=1. This architecture is based on the "backbone" architecture from Page (2018a). For $k = 64$, this CNN has 1558026 parameters and can reach $> 90\%$ test accuracy on CIFAR-10 (Krizhevsky (2009a)) with data-augmentation. The scaling of model size with k is shown in Figure B.1a.

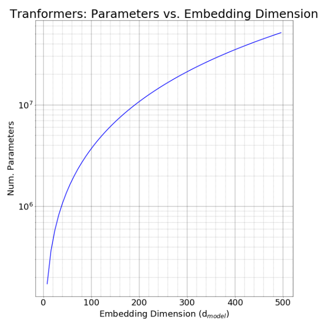
Transformers. We consider the encoder-decoder Transformer model from Vaswani et al. (2017a) with 6 layers and 8 attention heads per layer, as implemented by fairseq Ott et al. (2019). We scale the size of the network by modifying the embedding dimension (d_{model}), and scale the width of the fully-connected layers proportionally ($d_{\text{ff}} = 4d_{\text{model}}$). We train with 10% label smoothing and no drop-out, for 80 gradient steps.



(a) 5-layer CNNs



(b) ResNet18s



(c) Transformers

Figure B.1: Scaling of model size with our parameterization of width & embedding dimension.

B.1.2 Image Classification: Experimental Setup

We describe the details of training for CNNs and ResNets below.

Loss function: Unless stated otherwise, we use the cross-entropy loss for all the experiments.

Data-augmentation: In experiments where data-augmentation was used, we apply `RandomCrop(32, padding=4)` and `RandomHorizontalFlip`. In experiments with added label noise, the label for all augmentations of a given training sample are given the same label.

Regularization: No explicit regularization like weight decay or dropout was applied unless explicitly stated.

Initialization: We use the default initialization provided by PyTorch for all the layers.

Optimization:

- **Adam:** Unless specified otherwise, learning rate was set at constant to $1e{-4}$ and all other parameters were set to their default PyTorch values.
- **SGD:** Unless specified otherwise, learning rate schedule inverse-square root (defined below) was used with initial learning rate $\gamma_0 = 0.1$ and updates every $L = 512$ gradient steps. No momentum was used.

We found our results are robust to various other natural choices of optimizers and learning rate schedule. We used the above settings because (1) they optimize well, and (2) they do not require experiment-specific hyperparameter tuning, and allow us to use the same optimization across many experiments.

Batch size: All experiments use a batchsize of 128.

Learning rate schedule descriptions:

- **Inverse-square root** (γ_0, L): At gradient step t, the learning rate is set to $\gamma(t) := \frac{\gamma_0}{\sqrt{1+\lfloor t/512 \rfloor}}$. We set learning-rate with respect to number of gradient steps, and not epochs, in order to allow comparison between experiments with varying train-set sizes.

- **Dynamic drop (γ_0, drop, patience)**: Starts with an initial learning rate of γ_0 and drops by a factor of 'drop' if the training loss has remained constant or become worse for 'patience' number of gradient steps.

B.1.3 Neural Machine Translation: Experimental Setup

Here we describe the experimental setup for the neural machine translation experiments.

Training procedure.

In this setting, the distribution $\mathcal{D}$ consists of triples

$$(x, y, i) \ : \ x \in V_{src}^*, \ y \in V_{tgt}^*, \ i \in \{0, \ldots, |y|\}$$

where V_{src} and V_{tgt} are the source and target vocabularies, the string x is a sentence in the source language, y is its translation in the target language, and i is the index of the token to be predicted by the model. We assume that $i|x, y$ is distributed uniformly on $\{0, \ldots, |y|\}$.

A standard probabilistic model defines an autoregressive factorization of the likelihood:

$$p_M(y|x) = \prod_{i=1}^{|y|} p_M(y_i|y_{<i}, x).$$

Given a set of training samples S, we define

$$\mathrm{Error}_S(M) = \frac{1}{|S|} \sum_{(x,y,i)\in S} - \log p_M(y_i|y_{<i}, x).$$

In practice, S is *not* constructed from independent samples from D, but rather by first sampling (x, y) and then including all $(x, y, 0), \ldots, (x, y, |y|)$ in S.

For training transformers, we replicate the optimization procedure specified in Vaswani et al.

(2017a) section 5.3, where the learning rate schedule consists of a "warmup" phase with linearly increasing learning rate followed by a phase with inverse square-root decay. We preprocess the data using byte pair encoding (BPE) as described in Sennrich, Haddow, and Birch (2016). We use the implementation provided by fairseq (`https://github.com/pytorch/fairseq`).

Datasets. The IWSLT '14 German to English dataset contains TED Talks as described in Cettolo, Girardi, and Federico (2012). The WMT '14 English to French dataset is taken from `http://www.statmt.org/wmt14/translation-task.html`.

B.1.4 Per-section Experimental Details

Here we provide full details for experiments in the body, when not otherwise provided.

Introduction: Experimental Details Figure 4.1: All models were trained using Adam with learning-rate 0.0001 for 4K epochs. Plotting means and standard deviations for 5 trials, with random network initialization.

Model-wise Double Descent: Experimental Details Figure 4.7: Plotting means and standard deviations for 5 trials, with random network initialization.

Sample-wise Nonmonotonicity: Experimental Details Figure 4.11a: All models are trained with SGD for 500K epochs, and data-augmentation. Bottom: Means and standard deviations from 5 trials with random initialization, and random subsampling of the train set.

B.2 Extended discussion of related work

Belkin et al. (2019a): This paper proposed, in very general terms, that the apparent contradiction between traditional notions of the bias-variance trade-off and empirically successful practices in deep learning can be reconciled under a double-descent curve—as model complexity increases, the test error follows the traditional "U-shaped curve", but beyond the point of interpolation, the error starts to *decrease*. This work provides empirical evidence for the double-descent curve with fully connected networks trained on subsets of MNIST, CIFAR10, SVHN and TIMIT datasets. They use the l_2 loss for their experiments. They demonstrate that neural networks are not an aberration in this regard—double-descent is a general phenomenon observed also in linear regression with random features and random forests.

Theoretical works on linear least squares regression: A variety of papers have attempted to theoretically analyze this behavior in restricted settings, particularly the case of least squares regression under various assumptions on the training data, feature spaces and regularization method.

1. Advani and Saxe (2017); Hastie et al. (2019) both consider the linear regression problem stated above and analyze the generalization behavior in the asymptotic limit $N, D \to \infty$ using random matrix theory. Hastie et al. (2019) highlight that when the model is misspecified, the minimum of training error can occur for over-parameterized models
2. Belkin et al. (2019b) Linear least squares regression for two data models, where the input data is sampled from a Gaussian and a Fourier series model for functions on a circle. They provide a finite-sample analysis for these two cases
3. Bartlett et al. (2020) provides generalization bounds for the minimum l_2-norm interpolant for Gaussian features

4. Muthukumar et al. (2019) characterize the fundamental limit of of any interpolating solution in the presence of noise and provide some interesting Fourier-theoretic interpretations.
5. Mei and Montanari (2019): This work provides asymptotic analysis for ridge regression over random features

Similar double descent behavior was investigated in Opper (1995, 2001)

Geiger et al. (2019b) showed that deep fully connected networks trained on the MNIST dataset with hinge loss exhibit a "jamming transition" when the number of parameters exceeds a threshold that allows training to near-zero train loss. Geiger et al. (2019a) provide further experiments on CIFAR-10 with a convolutional network. They also highlight interesting behavior with ensembling around the critical regime, which is consistent with our informal intuitions in Section 4.5.

Advani and Saxe (2017); Geiger et al. (2019b,a) also point out that double-descent is not observed when optimal early-stopping is used.

B.3 Random Features: A Case Study

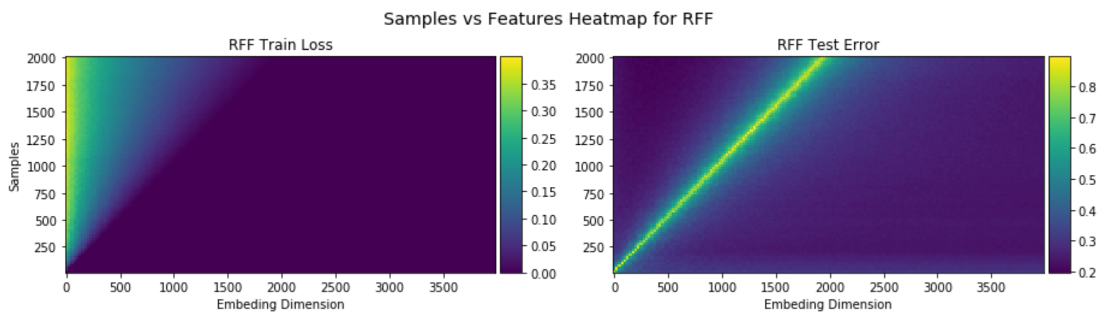


Figure B.2: Random Fourier Features on the Fashion MNIST dataset. The setting is equivalent to two-layer neural network with e^{-ix} activation, with randomly-initialized first layer that is fixed throughout training. The second layer is trained using gradient flow.

In this section, for completeness sake, we show that both the model- and sample-wise double descent phenomena are not unique to deep neural networks—they exist even in the setting of Random Fourier Features of Rahimi and Recht (2007). This setting is equivalent to a two-layer neural network with e^{-ix} activation. The first layer is initialized with a $\mathcal{N}(0, \frac{1}{d})$ Gaussian distribution and then fixed throughout training. The width (or embedding dimension) d of the first layer parameterizes the model size. The second layer is initialized with 0s and trained with MSE loss.

Figure B.2 shows the grid of Test Error as a function of both number of samples n and model size d. Note that in this setting EMC $= d$ (the embedding dimension). As a result, as demonstrated in the figure, the peak follows the path of $n = d$. Both model-wise and sample-wise (see figure B.3) double descent phenomena are captured, by horizontally and vertically crossing the grid, respectively.

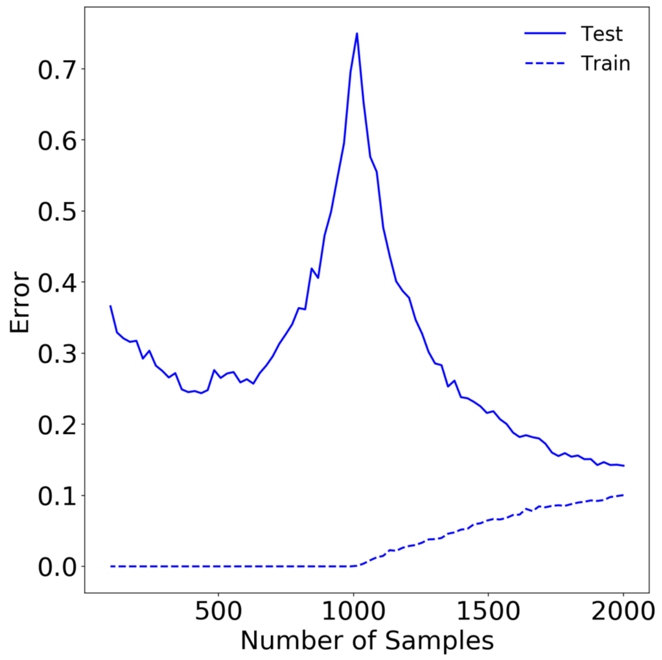
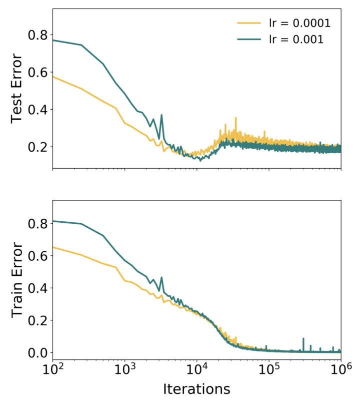


Figure B.3: Sample-wise double-descent slice for Random Fourier Features on the Fashion MNIST dataset. In this figure the embedding dimension (number of random features) is 1000.

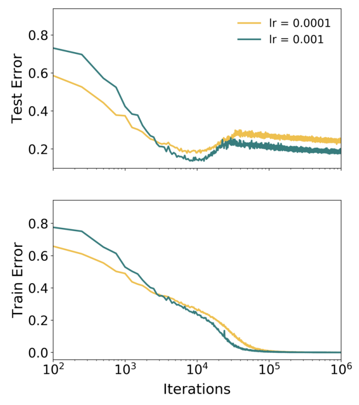
B.4 Appendix: Additional Experiments

B.4.1 Epoch-wise Double Descent: Additional results

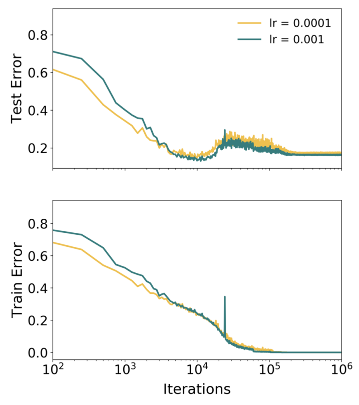
Here, we provide a rigorous evaluation of epoch-wise double descent for a variety of optimizers and learning rate schedules. We train ResNet18 on CIFAR-10 with data-augmentation and 20% label noise with three different optimizers—Adam, SGD, SGD + Momentum (momentum set to 0.9) and three different learning rate schedules—constant, inverse-square root, dynamic drop for differnet values of initial learning rate. We observe that double-descent occurs reliably for all optimizers and learning rate schedules and the peak of the double descent curve shifts with the interpolation point.



(a) Constant learning rate



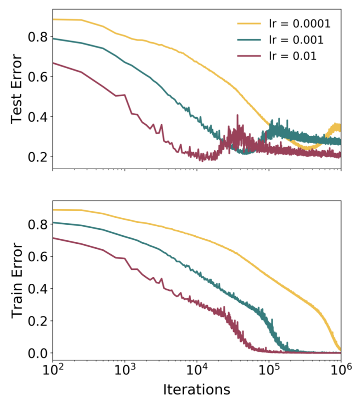
(b) Inverse-square root learning rate



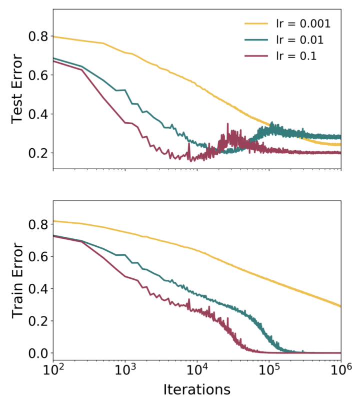
(c) Dynamic learning rate

Figure B.4: Epoch-wise double descent for ResNet18 trained with Adam and multiple learning rate schedules

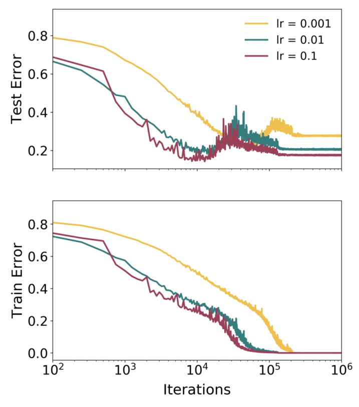
A practical recommendation resulting from epoch-wise double descent is that stopping the training when the test error starts to increase may not always be the best strategy. In some cases, the test error may decrease again after reaching a maximum, and the final value may be lower than the minimum earlier in training.



(a) Constant learning rate

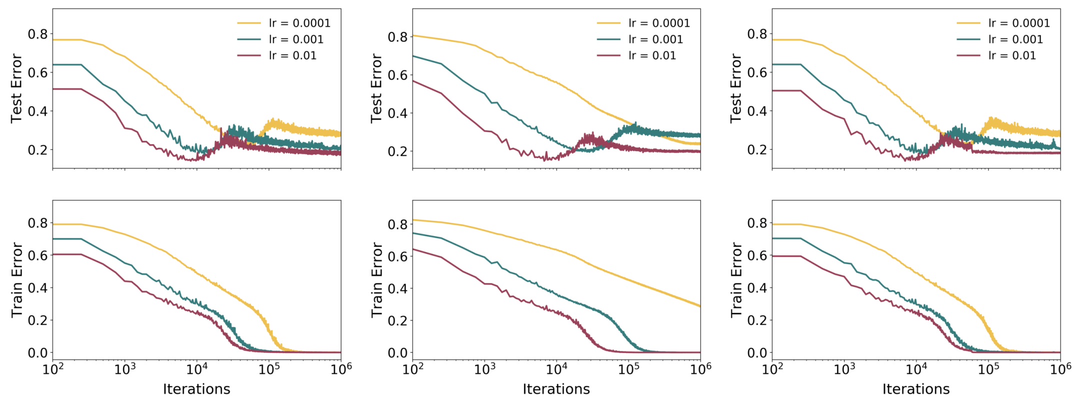


(b) Inverse-square root learning rate



(c) Dynamic learning rate

Figure B.5: Epoch-wise double descent for ResNet18 trained with SGD and multiple learning rate schedules



(a) Constant learning rate **(b)** Inverse-square root learning rate **(c)** Dynamic learning rate

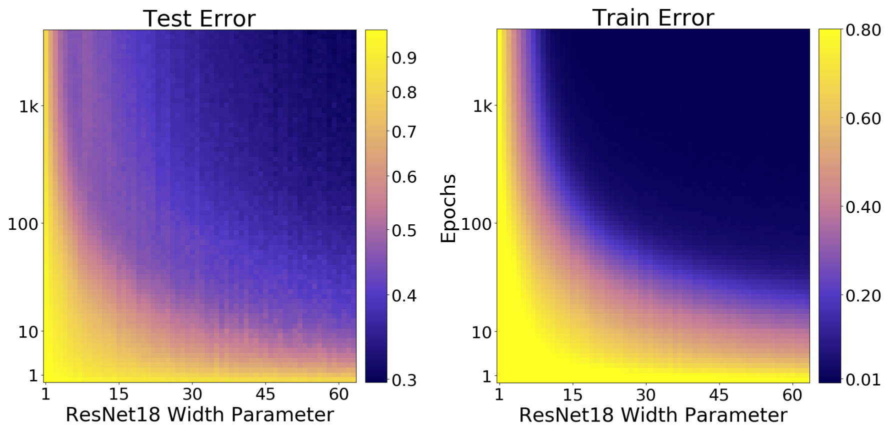
Figure B.6: Epoch-wise double descent for ResNet18 trained with SGD+Momentum and multiple learning rate schedules

B.4.2 Model-Wise Double Descent: Additional Results

Clean Settings With Model-wise Double Descent

CIFAR100, ResNet18

CIFAR100, Standard CNN



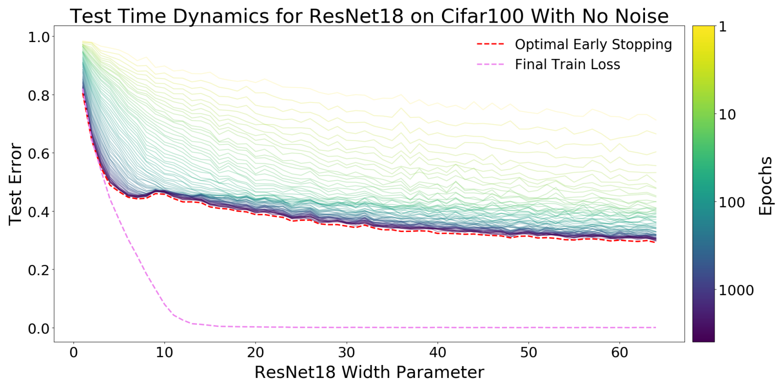
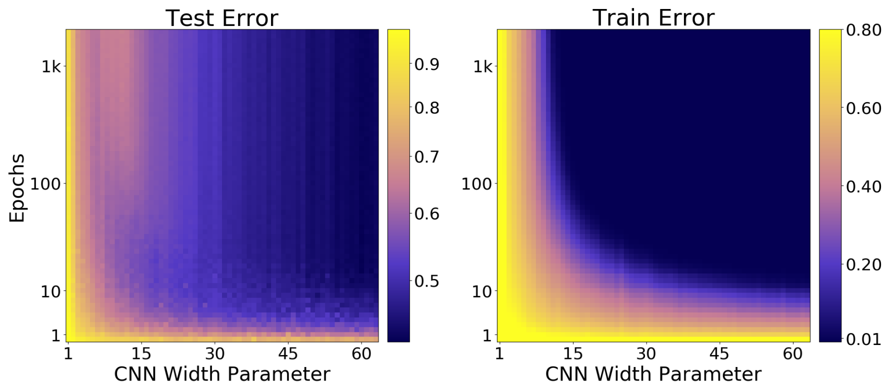


Figure B.7: Top: Train and test performance as a function of both model size and train epochs. Bottom: Test error dynamics of the same model (ResNet18, on CIFAR-100 with no label noise, data-augmentation and Adam optimizer trained for 4k epochs with learning rate 0.0001). Note that even with optimal early stopping this setting exhibits double descent.

Weight Decay

Here, we now study the effect of varying the level of regularization on test error. We train CIFAR10 with data-augmentation and 20% label noise on ResNet18 for weight decay co-efficients λ ranging from 0 to 0.1. We train the networks using SGD + inverse-square root learning rate. Figure below shows a picture qualitatively very similar to that observed for model-wise double descent wherein ”model complexity” is now controlled by the regularization parameter. This confirms our generalized double descent hypothesis along yet another axis of Effective Model Complexity.



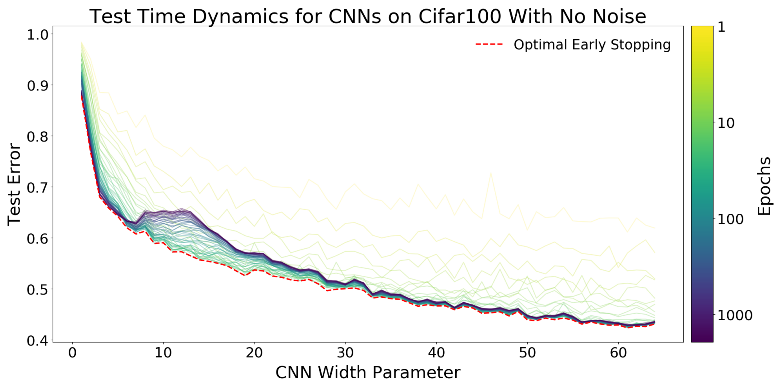
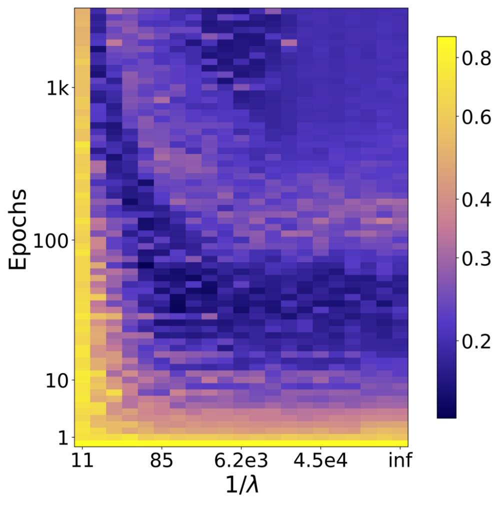
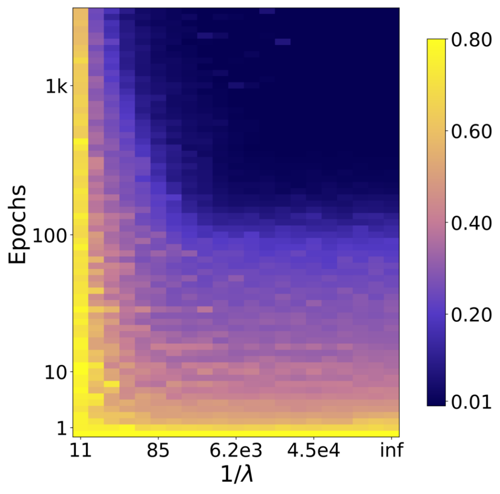


Figure B.8: Top: Train and test performance as a function of both model size and train epochs. Bottom: Test error dynamics of the same models. 5-Layer CNNs, CIFAR-100 with no label noise, no data-augmentation Trained with SGD for 1e6 steps. Same experiment as Figure 4.7.



(a) Test Error



(b) Train Loss

Figure B.10: Generalized double descent for weight decay. We found that using the same initial learning rate for all weight decay values led to training instabilities. This resulted in some noise in the Test Error (Weight Decay $\times$ Epochs) plot shown above.

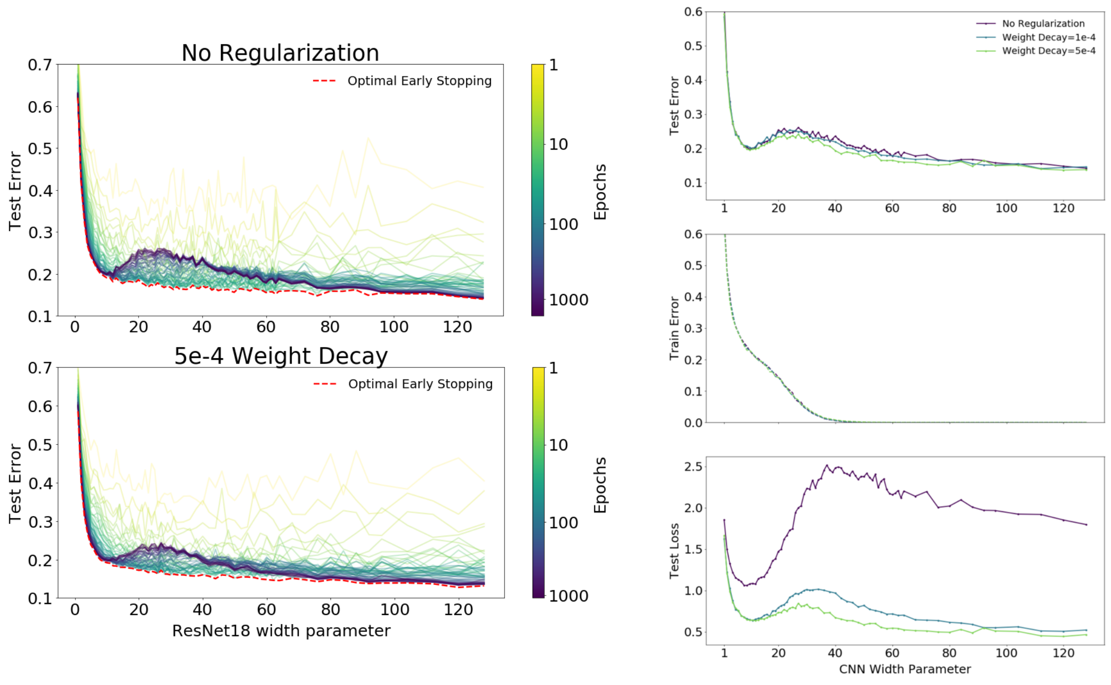


Figure B.9: Left: Test error dynamics with weight decay of 5e-4 (bottom left) and without weight decay (top left). Right: Test and train error and *test loss* for models with varying amounts of weight decay. All models are 5-Layer CNNs on CIFAR-10 with 10% label noise, trained with data-augmentation and SGD for 500K steps.

Early Stopping does not exhibit double descent

Language models

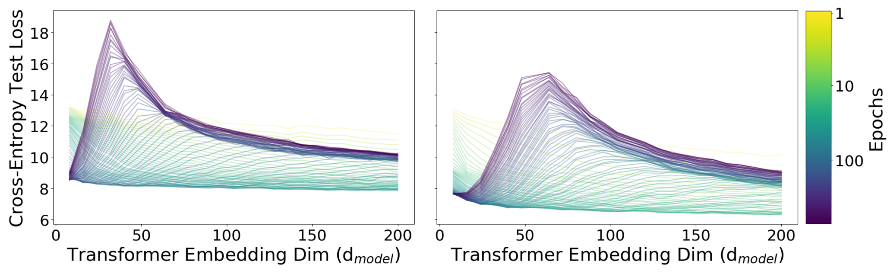


Figure B.11: Model-wise test error dynamics for a subsampled IWSLT'14 dataset. Left: 4k samples, Right: 18k samples. Note that with optimal early-stopping, more samples is always better.

CIFAR10, 10% noise, SGD

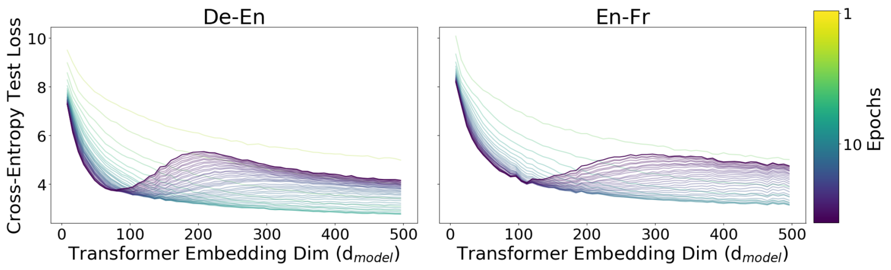


Figure B.12: Model-wise test error dynamics for a IWSLT‘14 de-en and subsampled WMT‘14 en-fr datasets. Left: IWSLT‘14, Right: subsampled (200k samples) WMT‘14. Note that with optimal early-stopping, the test error is much lower for this task.

Training Procedure

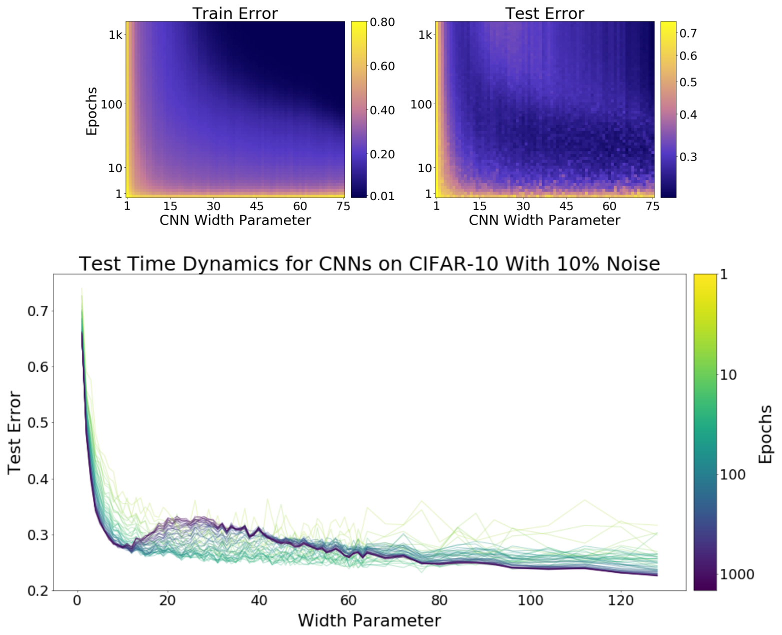


Figure B.13: Top: Train and test performance as a function of both model size and train epochs. Bottom: Test error dynamics of the same model (CNN, on CIFAR-10 with 10% label noise, data-augmentation and SGD optimizer with learning rate $\propto 1/\sqrt{T}$).

B.4.3 Ensembling

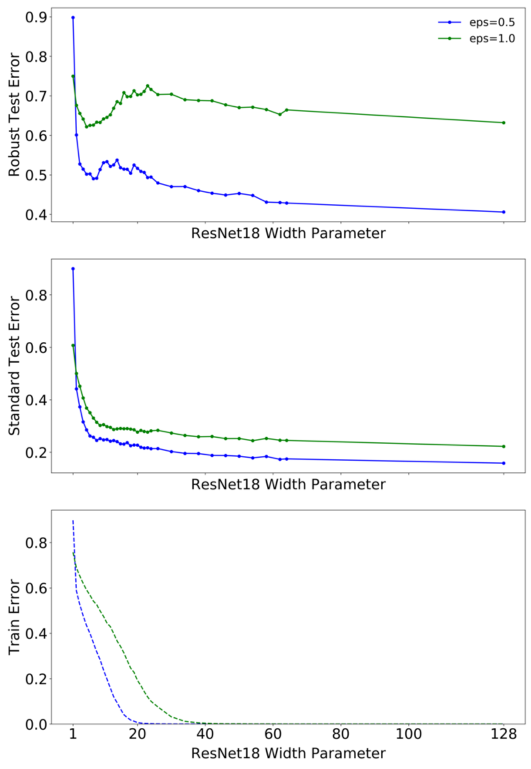


Figure B.14: Model-wise double descent for adversarial training ResNet18s on CIFAR-10 (subsampled to 25k train samples) with no label noise. We train for L2 robustness of radius $\varepsilon = 0.5$ and $\varepsilon = 1.0$, using 10-step PGD (Goodfellow, Shlens, and Szegedy (2015); Madry, Makelov, Schmidt, Tsipras, and Vladu (2018)). Trained using SGD (batch size 128) with learning rate 0.1 for 400 epochs, then 0.01 for 400 epochs.

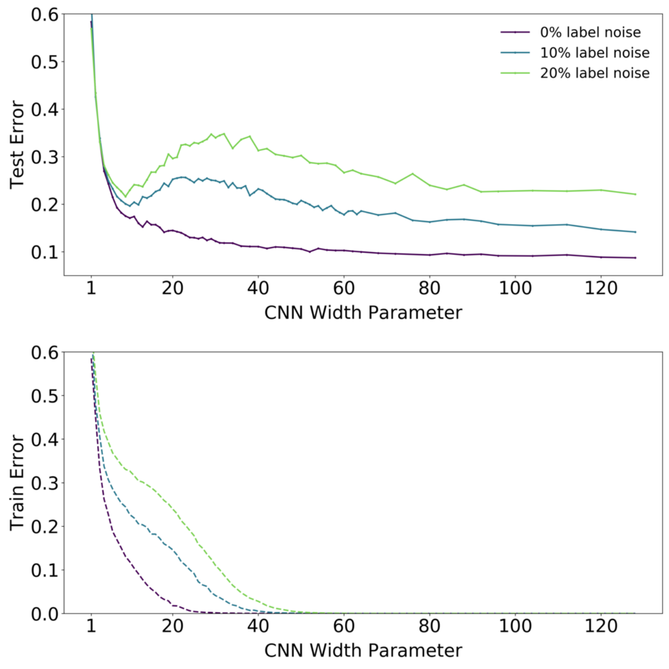


Figure B.15

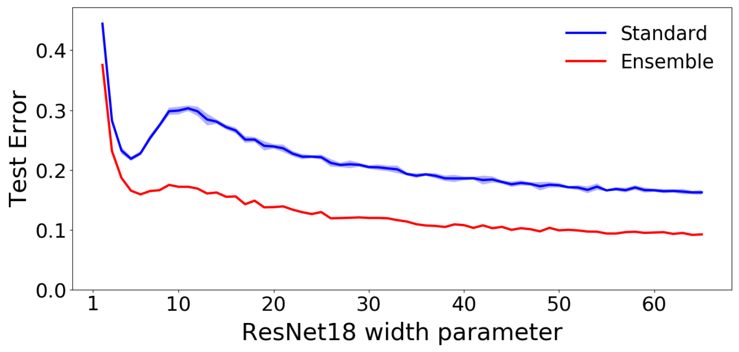


Figure B.16: Effect of Ensembling (ResNets, 15% label noise). Test error of an ensemble of 5 models, compared to the base models. The ensembled classifier is determined by plurality vote over the 5 base models. Note that emsembling helps most around the critical regime. All models are ResNet18s trained on CIFAR-10 with 15% label noise, using Adam for 4K epochs (same setting as Figure 4.1). Test error is measured against the original (not noisy) test set, and each model in the ensemble is trained using a train set with independently-sampled 15% label noise.

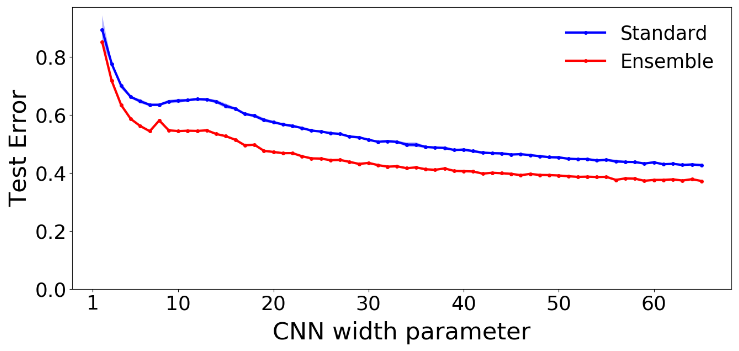


Figure B.17: Effect of Ensembling (CNNs, no label noise). Test error of an ensemble of 5 models, compared to the base models. All models are 5-layer CNNs trained on CIFAR-10 with no label noise, using SGD and no data augmentation. (same setting as Figure 4.7).

C

The Deep Bootstrap Framework

C.1 Toy Example

Here we present a theoretically-inspired toy example, giving a simple setting where the bootstrap gap is small, but the generalization gap is large. We also give an analogous example where the bootstrap error is large. The purpose of these examples is (1) to present a simple setting where the bootstrap framework can be more useful than studying the generalization gap. And (2) to illustrate that the bootstrap gap is not always small, and can be large in certain standard settings.

We consider the following setup. Let us pass to a regression setting, where we have a distribution over $(x, y) \in \mathbb{R}^d \times \mathbb{R}$, and we care about mean-square-error instead of classification error. That is, for a model f, we have $\text{TestMSE}(f) := \mathbb{E}_{x,y}[(f(x) - y)^2]$. Both our examples are from the following class of distributions in dimension $d = 1000$.

$$x \sim \mathcal{N}(0, V)$$
$$y := \sigma(\langle \beta^*, x \rangle)$$

where $\beta^* \in \mathbb{R}^d$ is the ground-truth, and σ is a pointwise activation function. The model family is linear,

$$f_\beta(x) := \langle \beta, x \rangle$$

We draw n samples from the distribution, and train the model f_β using full-batch gradient descent on the empirical loss:

$$\text{TrainMSE}(f_\beta) := \frac{1}{n} \sum_i (f(x_i) - y_i)^2 = \frac{1}{n} ||X\beta - y||^2$$

We chose $\beta^* = e_1$, and covariance V to be diagonal with 10 eigenvalues of 1 and the remaining eigenvalues of 0.1. That is, x is essentially 10-dimensional, with the remaining coordinates "noise."

The two distributions are instances of the above setting for different choices of parameters.

- **Setting A.** Linear activation $\sigma(x) = x$. With $n = 20$ train samples.
- **Setting B.** Sign activation $\sigma(x) = \text{sgn}(x)$. With $n = 100$ train samples.

Setting A is a standard well-specified linear regression setting. Setting B is a misspecified regression setting. Figure C.1 shows the Real and Ideal worlds in these settings, for gradient-descent on the empirical loss (with step-size $\eta = 0.1$). Observe that in the well-specified Setting A, the Ideal World performs much better than the Real World, and the bootstrap framework is not as useful. However, in the misspecified Setting B, the bootstrap gap remains small even as the generalization gap grows.

This toy example is contrived to help isolate factors important in more realistic settings. We have

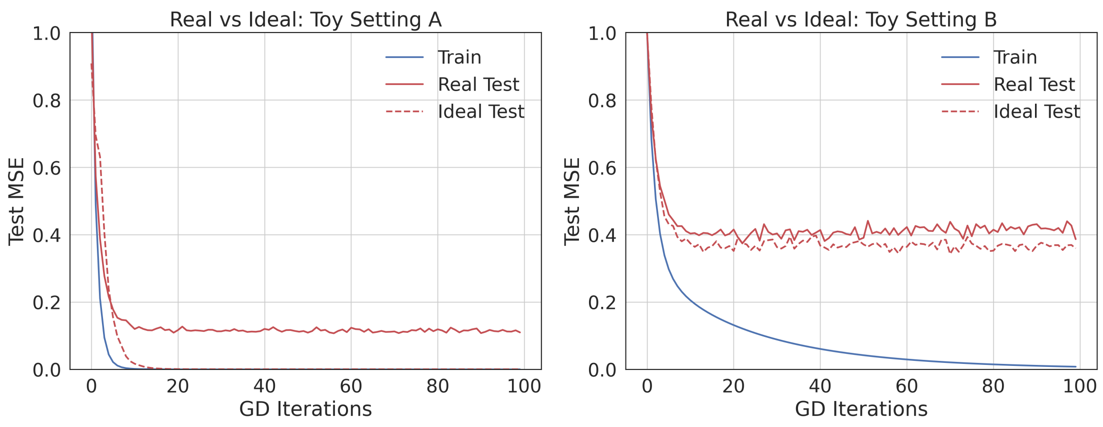


Figure C.1: Toy Example. Examples of settings with large and small bootstrap error.

observed behavior similar to Setting B in other simple settings with real data, such as regression on MNIST/Fashion-MNIST, as well as in the more complex settings in the body of this paper.

C.2 Additional Figures

C.2.1 Introduction Experiment

Figure C.2 shows the same experiment as Figure 6.1 in the Introduction, including the train error in the Real World. Notice that the bootstrap error remains small, even as the generalization gap (between train and test) grows.

C.2.2 DARTS Architectures

Figure C.3 shows the Real vs Ideal world for trained random DARTS architectures.

C.2.3 Effect of Data Augmentation

Figure C.4 shows the effect of data-augmentation in the Ideal World, for several selected architectures on CIFAR-5m. Recall that data augmentation in the Ideal World corresponds to randomly

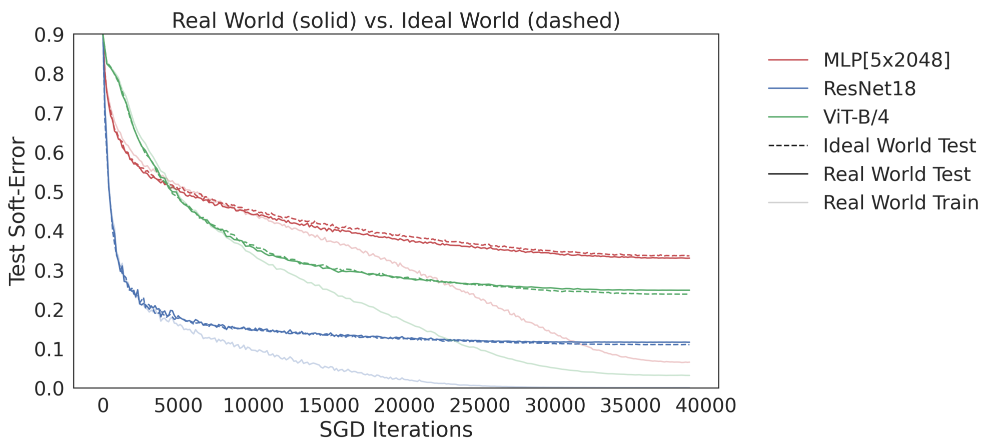


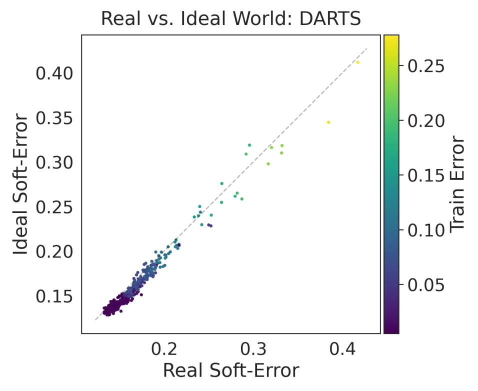
Figure C.2: The corresponding train soft-errors for Figure 6.1.

augmenting each fresh sample once, as opposed to augmenting the same sample multiple times. We train with SGD using the same hyperparameters as the main experiments (described in Appendix C.4.2). We use standard CIFAR-10 data augmentation: random crop and random horizontal flip.

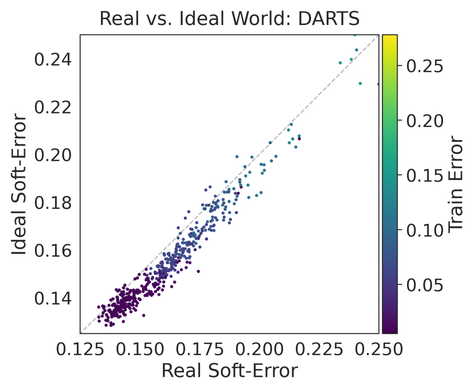
The test performance without augmentation is shown as solid lines, and with augmentation as dashed lines. Note that VGG and ResNet do not behave differently with augmentation, but augmentation significantly hurts AlexNet and the MLP. This may be because VGG and ResNet have global spatial pooling, which makes them (partially) shift-invariant, and thus more amenable to the random cropping. In contrast, augmentation hurts the architectures without global pooling, perhaps because for these architectures, augmented samples appear more out-of-distribution.

Figure C.5a shows the same architectures and setting as Figure 5.2 but trained without data augmentation. That is, we train on 50K samples from CIFAR-5m, using SGD with cosine decay and initial learning rate $\{0.1, 0.01, 0.001\}$.

Figure C.5b shows learning curves with and without data augmentation of a ResNet-18 on $n = 10k$ samples. This is the analogous setting of Figure 5.5a in the body, which is for $n = 50k$ samples.



(a) All architectures.



(b) High accuracy architectures.

Figure C.3: Random DARTS Architectures. Panel (b) shows zoomed view of panel (a).

C.2.4 Adam

Figure C.6 shows several experiments with the Adam optimizer (Kingma and Ba, 2015) in place of SGD. We train all architectures on 50K samples from CIFAR-5m, with data-augmentation, batch-size 128, using Adam with default parameters (lr=$0.001, \beta_1 = 0.9, \beta_2 = 0.999$).

C.2.5 Pretraining

Pretrained MLP

Figure C.8 shows the effect of pretraining for an MLP (3x2048) on CIFAR-5m, by comparing training from scratch (random initialization) to training from an ImageNet-pretrained initialization. The pretrained MLP generalizes better in the Real World, and also optimizes faster in the Ideal World. We fine tune on 50K samples from CIFAR-5m, with no data-augmentation.

For ImageNet-pretraining, we train the MLP[3x2048] on full ImageNet (224px, 1000 classes), using Adam with default settings, and batchsize 1024. We use standard ImageNet data augmentation (random resized crop + horizontal flip) and train for 500 epochs. This MLP achieves test accuracy 21% and train accuracy 30% on ImageNet. For fine-tuning, we adapt the network to 32px

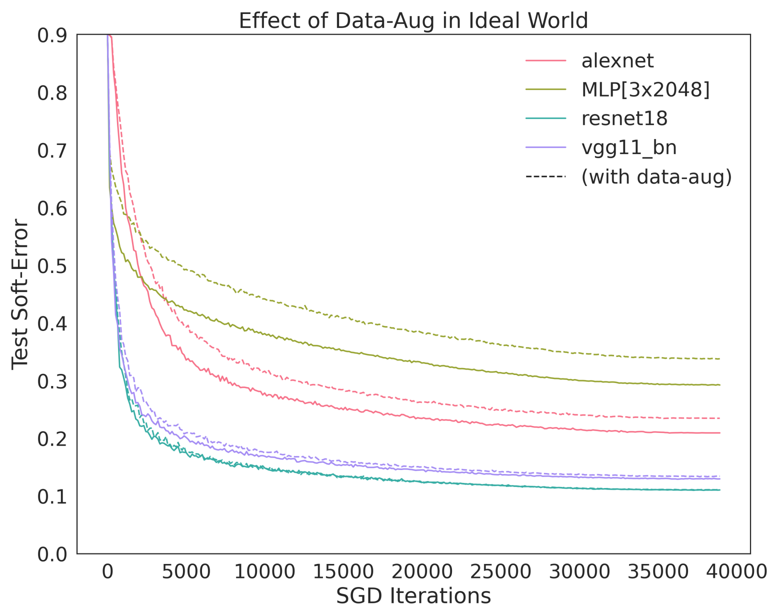


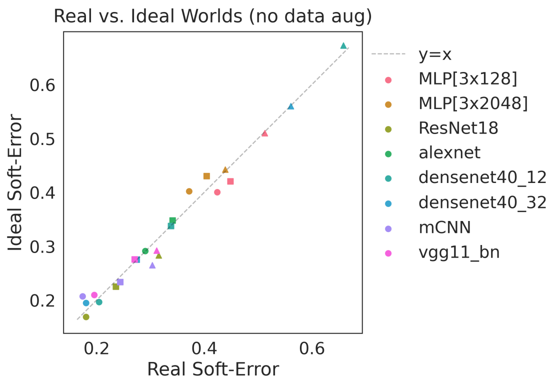
Figure C.4: Effect of Data Augmentation in the Ideal World.

input size by resizing the first layer filters from 224x224 to 32x32 via bilinear interpolation. We then replace the classification layer, and fine-tune the entire network on CIFAR-5m.

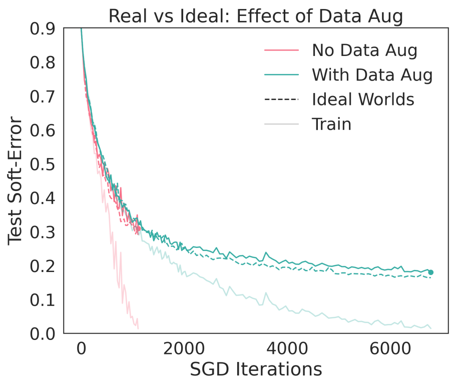
Pretrained Vision Transformer

Figure C.7 shows the effect of pretraining for Vision Transformer (ViT-B/4). We compare ViT-B/4 trained from scratch to training from an ImageNet-pretrained initialization. We fine tune on 50K samples from CIFAR-5m, with standard CIFAR-10 data augmentation. Notice that pretrained ViT generalizes better in the Real World, and also optimizes correspondingly faster in the Ideal World.

Both ViT models are fine-tuned using SGD identical to Figure 6.1 in the Introduction, as described in Section C.4.1. For ImageNet pretraining, we train on ImageNet resized to 32×32, after standard data augmentation. We pretrain for 30 epochs using Adam with batchsize 2048 and constant learning rate 1e-4. We then replace and zero-initialize the final layer in the MLP head, and

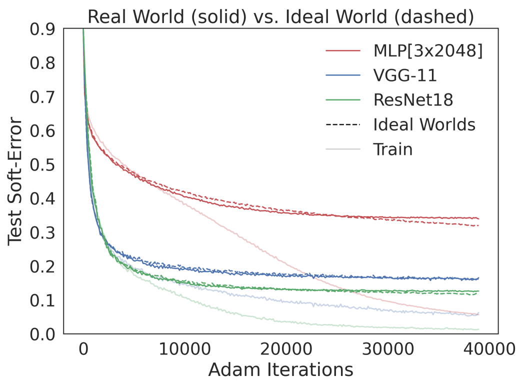


(a) No Data-augmentation. Compare to Figure 5.2.

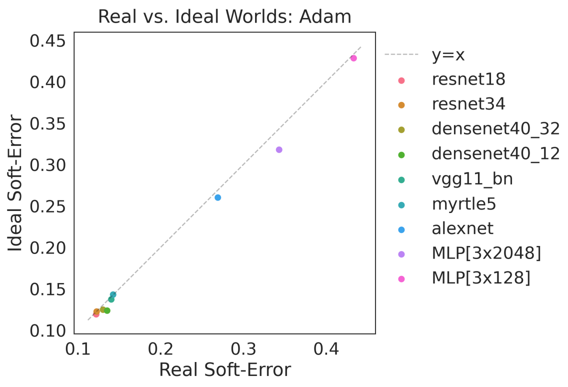


(b) ResNet-18 on 10K samples.

Figure C.5: Effect of Data Augmentation.



(a) Real vs. Ideal learning curves.



(b) Real vs. Ideal world at the end of training.

Figure C.6: Adam Experiments. For various architectures on 50K samples from CIFAR-5m.

fine-tune the full model for classification on CIFAR-5m. This pretraining process it not as extensive as in Dosovitskiy et al. (2021); we use it to demonstrate that our framework captures the effect of pretraining in various settings.

C.2.6 Learning Rate

Figure 5.2a shows that the Real and Ideal world remain close across varying initial learning rates. All of the figures in the body use a cosine decay learning rate schedule, but this is only for simplicity;

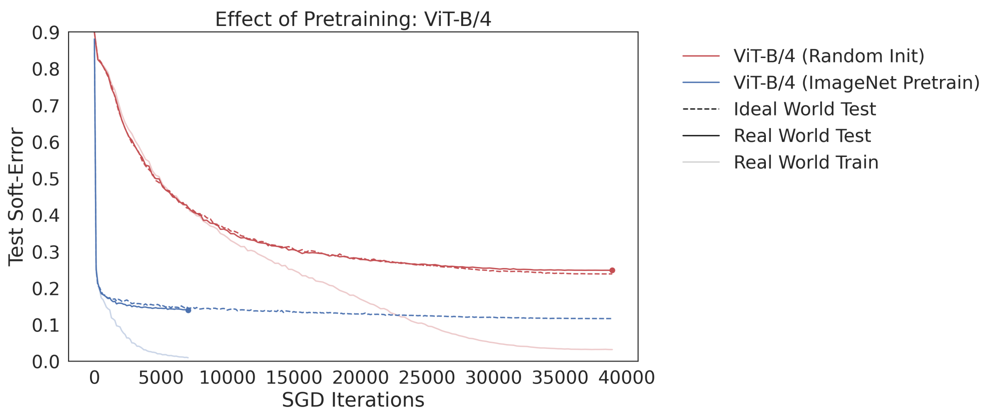


Figure C.7: Real vs. Ideal Worlds for Vision Transformer on CIFAR-5m, with and w/o pretraining.

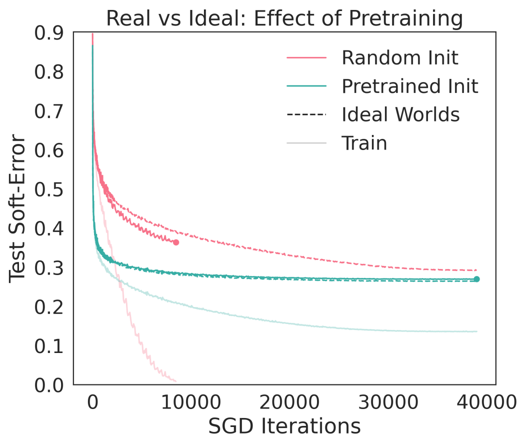


Figure C.8: ImageNet-Pretraining: MLP[3x2048].

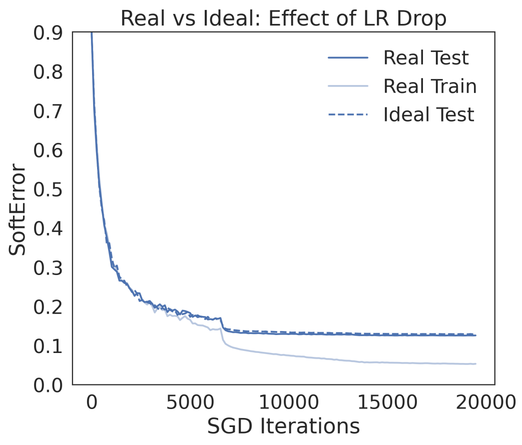


Figure C.9: Effect of Learning Rate Drop.

we observed that the effect of various learning rate schedules are mirrored in Real and Ideal worlds. For example, Figure C.9 shows a ResNet18 in the Real World trained with SGD for 50 epochs on CIFAR-5m, with a step-wise decay schedule (initial LR 0.1, dropping by factor 10 at 1/3 and 2/3 through training). Notice that the Ideal World error drops correspondingly with the Real World, suggesting that the LR drop has a similar effect on the population optimization as it does on the empirical optimization.

C.2.7 Difference Between Worlds

Figure C.10 shows test soft-error, test error, and test loss for the MLP from Figure 6.1.

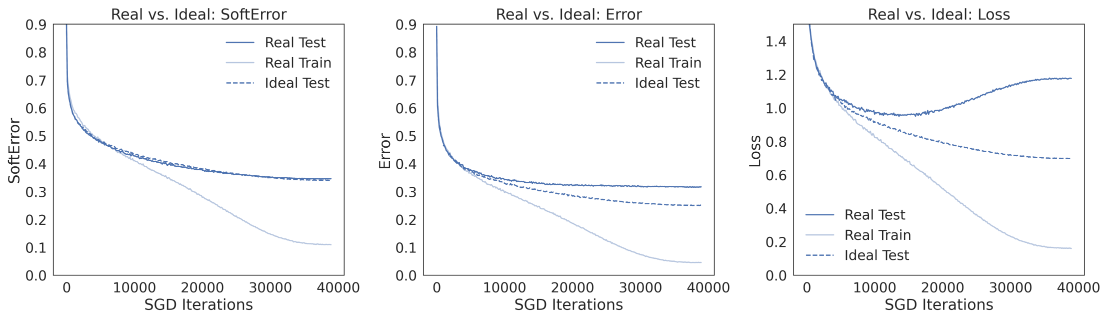


Figure C.10: SoftError vs. Error vs. Loss: MLP[5x2048].

C.2.8 Error vs. SoftError

Here we show the results of several of our experiments if we measure the bootstrap gap with respect to Test Error instead of SoftError. The bootstrap gap is often reasonably small even with respect to Error, though it is not as well behaved as SoftError.

Figure C.11 shows the same setting as Figure 5.2a in the body, but measuring Error instaed of SoftError.

Training with MSE

We can measure Test Error even for networks which do not naturally output a probability distribution. Here, we train various architectures on CIFAR-5m using the squared-loss (MSE) directly on logits, with no softmax layer. This follows the methodology in Hui and Belkin (2020). We train all Real-World models using SGD, batchsize 128, momentum 0.9, initial learning rate 0.002 with cosine decay for 100 epochs.

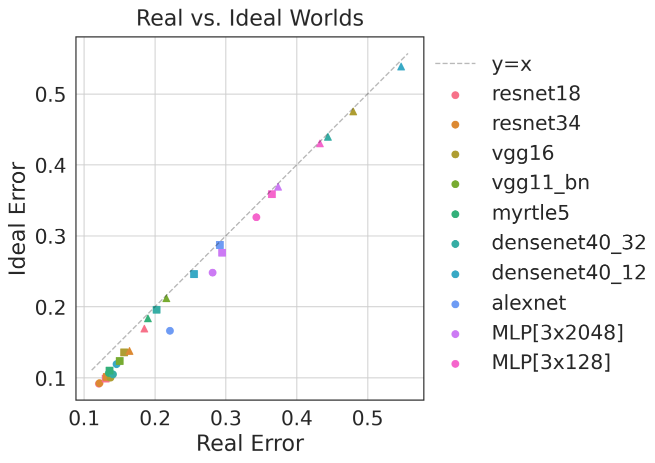
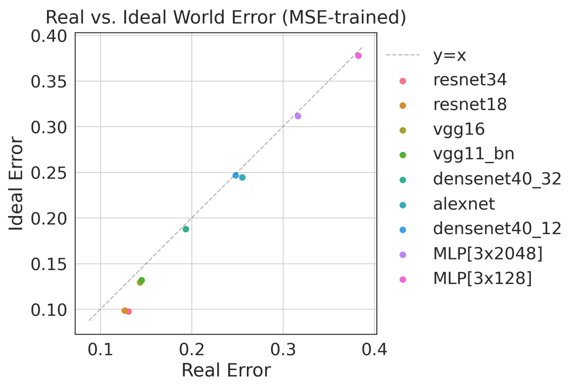
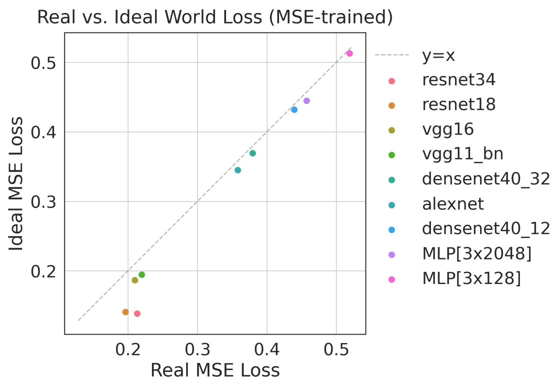


Figure C.11: Measuring Test Error instead of SoftError. Compare to Figure 5.2a

Figure C.12 shows the Test Error and Test Loss in the Real and Ideal Worlds. The bootstrap gap, with respect to test error, for MSE-trained networks is reasonably small – though there are deviations in the low error regime. Compare this to Figure 5.2a, which measures the SoftError for networks trained with cross-entropy.



(a) Test Error.



(b) Test Loss.

Figure C.12: Real vs. Ideal: Training with Squared Loss.

C.3 Bootstrap Connection

Here we briefly describe the connection between our Deep Bootstrap framework and the nonparametric bootstrap of Efron (1979).

For an online learning procedure $\mathcal{F}$ and a sequence of labeled samples $\{x_i\}$, let $\text{Train}^{\mathcal{F}}(x_1, x_2, \dots)$ denote the function which optimizes on the samples $x_1, x_2, \dots$ in sequence, and outputs the resulting model. (For example, the function which initializes a network of a certain architecture, and takes successive gradient steps on the sequence of samples, and outputs the resulting model).

For a given $(n, \mathcal{D}, \mathcal{F}, t)$, define the function $G : \mathcal{X}^t \to \mathbb{R}$ as follows. G takes as input t labeled samples $\{x_i\}$, and outputs the Test Soft-Error (w.r.t $\mathcal{D}$) of training on the sequence $\{x_i\}$. That is,

$$G(x_1, x_2, \dots x_t) := \text{TestSoftError}_{\mathcal{D}}(\text{Train}^{\mathcal{F}}(x_1, x_2, \dots, x_t))$$

Now, the Ideal World test error is simply G evaluated on iid samples $x_i \sim \mathcal{D}$:

$$\text{Ideal World:} \quad \text{TestSoftError}_{\mathcal{D}}(f_t^{\text{iid}}) = G(\{x_i\}) \quad \text{where } x_i \sim \mathcal{D}$$

The Real World, using a train set of size $n < t$, is equivalent* to evaluating G on t examples sampled with replacement from a train set of size n. This corresponds to training on the same sample multiple times, for t total train steps.

$$\text{Real World:} \quad \text{TestSoftError}_{\mathcal{D}}(f_t) = G(\{\widetilde{x}_i\}) \quad \text{where } S \sim \mathcal{D}^n; \widetilde{x}_i \sim S$$

Here, the samples $\widetilde{x}_i$ are drawn with replacement from the train set S. Thus, the Deep Bootstrap

*Technically we do not sample-with-replacement in the experiments, we simply reuse each sample a fixed number of times (once in each epoch). We describe it as sampling-with-replacement here to more clearly relate it to the nonparametric bootstrap.

error $\varepsilon = G(\{\widetilde{x_i}\}) - G(\{x_i\})$ measures the deviation of a certain function when it is evaluated on iid samples v.s. on samples-with-replacement, which is exactly the form of bootstrap error in applications of the nonparametric bootstrap (Efron, 1979; Efron and Tibshirani, 1986, 1994).

C.4 Appendix: Experimental Details

Technologies. All experiments run on NVIDIA V100 GPUs. We used PyTorch (Paszke, Gross, Massa, Lerer, Bradbury, Chanan, Killeen, Lin, Gimelshein, Antiga, Desmaison, Köpf, Yang, DeVito, Raison, Tejani, Chilamkurthy, Steiner, Fang, Bai, and Chintala, 2019), NumPy (Harris, Millman, van der Walt, Gommers, Virtanen, Cournapeau, Wieser, Taylor, Berg, Smith, Kern, Picus, Hoyer, van Kerkwijk, Brett, Haldane, del R'10, Wiebe, Peterson, G'erard-Marchant, Sheppard, Reddy, Weckesser, Abbasi, Gohlke, and Oliphant, 2020), Hugging Face transformers (Wolf, Debut, Sanh, Chaumond, Delangue, Moi, Cistac, Rault, Louf, Funtowicz, et al., 2019), pandas (McKinney et al., 2010), W&B (Biewald, 2020), Matplotlib (Hunter, 2007), and Plotly (Inc., 2015).

C.4.1 Introduction Experiment

All architectures in the Real World are trained with $n = 50K$ samples from CIFAR-5m, using SGD on the cross-entropy loss, with cosine learning rate decay, for 100 epochs. We use standard CIFAR-10 data augmentation of random crop+horizontal flip. All models use batch size 128, so they see the same number of samples at each point in training.

The ResNet is a preactivation ResNet18 (He et al., 2016b), the MLP has 5 hidden layers of width 2048, with pre-activation batch norm. The Vision Transformer uses the ViT-Base configuration from Dosovitskiy et al. (2021), with a patch size of 4×4 (adapted for the smaller CIFAR-10 image size of 32×32). We use the implementation from `https://github.com/lucidrains/vit-pytorch`. We train all architectures including ViT from scratch, with no pretraining. ResNets and MLP use initial learning rate 0.1 and momentum 0.9. ViT uses initial LR 0.01, momentum 0.9, and weight decay 1e-4. We did not optimize ViT hyperparameters as extensively as in Dosovitskiy et al. (2021); this experiment is only to demonstrate that our framework is meaningful for diverse architectures.

Figure 6.1 plots the Test Soft-Error over the course of training, and the Train Soft-Error at the end of training. We plot median over 10 trials (with random sampling of the train set, random initialization, and random SGD order in each trial).

C.4.2 Main Experiments

For CIFAR-5m we use the following architectures: AlexNet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2015), Preactivation ResNets (He et al., 2016b), DenseNet (Huang, Liu, van der Maaten, and Weinberger, 2017). The Myrtle5 architecture is a 5-layer CNN introduced by (Page, 2018b).

In the Real World, we train these architectures on $n = 50K$ samples from CIFAR-5m using cross-entropy loss. All models are trained with SGD with batchsize 128, initial learning rate $\{0.1, 0.01, 0.001\}$, cosine learning rate decay, for 100 total epochs, with data augmentation: random horizontal flip and `RandomCrop(32, padding=4)`. We plot median over 10 trials (with random sampling of the train set, random initialization, and random SGD order in each trial).

DARTS Architectures. We sample architectures from the DARTS search space (Liu et al., 2019), as implemented in the codebase of Dong and Yang (2020). We follow the parameters used for CIFAR-10 in Dong and Yang (2020), while also varying width and depth for added diversity. Specifically, we use 4 nodes, number of cells $\in \{1, 5\}$, and width $\in \{16, 64\}$. We train all DARTS architectures with SGD, batchsize 128, initial learning rate 0.1, cosine learning rate decay, for 100 total epochs, with standard augmentation (random crop+flip).

ImageNet: DogBird All architectures for ImageNet-DogBird are trained with SGD, batchsize

128, learning rate 0.01, momentum 0.9, for 120 epochs, with standard ImageNet data augmentation (random resized crop to 224px, horizontal flip). We report medians over 10 trials for each architecture.

We additional include the ImageNet architectures: BagNet (Brendel and Bethge, 2019), MobileNet (Sandler, Howard, Zhu, Zhmoginov, and Chen, 2018), and ResNeXt (Xie, Girshick, Dollár, Tu, and He, 2017). The architectures SCONV9 and SCONV33 refer to the S-CONV architectures defined by Neyshabur (2020), instantiated for ImageNet with base-width 48, image size 224, and kernel size $\{9, 33\}$ respectively.

C.4.3 Implicit Bias

We use the D-CONV architecture from (Neyshabur, 2020), with base width 32, and the corresponding D-FC architecture. We train both architectures with SGD, batchsize 128, initial learning rate 0.1, cosine learning rate decay, for 100 total epochs, with random crop + horizontal flip data-augmentation. We plot median errors over 10 trials.

C.4.4 Image-GPT Finetuning

We fine-tune iGPT-S, using the publicly available pretrained model checkpoints from Chen et al. (2020). The "Early" checkpoint in Figure 5.5b refers to checkpoint 131000, and the "Final" checkpoint is 1000000. Following Chen et al. (2020), we use Adam with ($lr = 0.003, \beta_1 = 0.9, \beta_2 = 0.95$), and batchsize 128. We do not use data augmentation. For simplicity, we differ slightly from Chen et al. (2020) in that we simply attach the classification head to the [average-pooled] last transformer layer, and we fine-tune using only classification loss and not the joint generative+classification loss used in Chen et al. (2020). Note that we fine-tune the entire model, not just the classification head.

C.5 Appendix: Datasets

C.5.1 CIFAR-5m

CIFAR-5m is a dataset of 6 million synthetic CIFAR-10-like images. We release this dataset publicly on Google Cloud Storage, as described in `https://github.com/preetum/cifar5m`.

The images are RGB 32 × 32px. We generate samples from the Denoising Diffusion generative model of Ho et al. (2020) trained on the CIFAR-10 train set (Krizhevsky, 2009b). We use the publicly available trained model and sampling code provided by the authors at `https://github.com/hojonathanho/diffusion`. We then label these unconditional samples by a 98.5% accurate Big-Transfer model (Kolesnikov et al., 2019). Specifically, we use the pretrained BiT-M-R152x2 model, fine-tuned on CIFAR-10 using the author-provided code at `https://github.com/google-research/big_transfer`. We use 5 million images for training, and reserve the remaining images for the test set.

The distribution of CIFAR-5m is of course not identical to CIFAR-10, but is close for research purposes. For example, we show baselines of training a network on 50K samples of either dataset (CIFAR-5m, CIFAR-10), and testing on both datasets. Table C.1 shows a ResNet18 trained with standard data-augmentation, and Table C.2 shows a WideResNet28-10 (Zagoruyko and Komodakis, 2016) trained with cutout augmentation (DeVries and Taylor, 2017). Mean of 5 trials for all results. In particular, the WRN-28-10 trained on CIFAR-5m achieves 91.2% test accuracy on the original CIFAR-10 test set. We hope that as simulated 3D environments become more mature (e.g. Gan, Schwartz, Alter, Schrimpf, Traer, De Freitas, Kubilius, Bhandwaldar, Haber, Sano, et al. (2020)), they will provide a source of realistic infinite datasets to use in such research.

Random samples from CIFAR-5m are shown in Figure C.13. For comparison, we show random samples from CIFAR-10 in Figure C.14.

Figure C.13: CIFAR-5m Samples. Random samples from each class (by row).

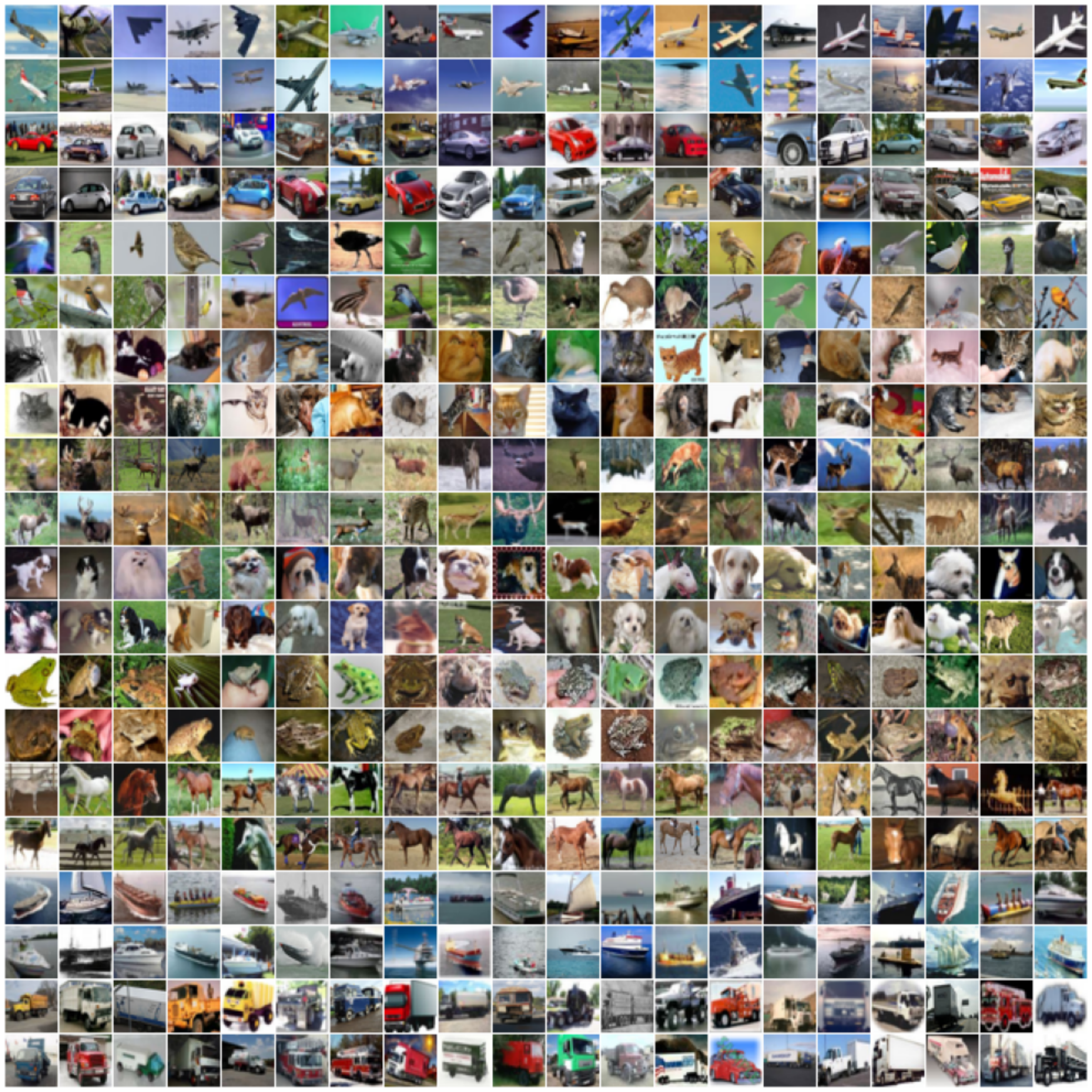


Figure C.14: CIFAR-10 Samples. Random samples from each class (by row).

Trained On	Test Error On	
	CIFAR-10	CIFAR-5m
CIFAR-10	0.050	0.096
CIFAR-5m	0.110	0.106

Table C.1: ResNet18 on CIFAR-10/5m

Trained On	Test Error On	
	CIFAR-10	CIFAR-5m
CIFAR-10	0.032	0.091
CIFAR-5m	0.088	0.097

Table C.2: WRN28-10 + cutout on CIFAR-10/5m

C.5.2 Imagenet: DogBird

The ImageNet-DogBird task is constructed by collapsing classes from ImageNet. The task is to distinguish dogs from birds. The dogs are all ImageNet classes under the WordNet synset "hunting dog" (including 63 ImageNet classes) and birds are all classes under synset "bird" (including 59 ImageNet classes). This is a relatively easy task compared to full ImageNet: A ResNet-18 trained on 10K samples from ImageNet-DogBird, with standard ImageNet data augmentation, can achieve test accuracy 95%. The listing of the ImageNet wnids included in each class is provided below.

Hunting Dogs (n2087122): n02091831, n02097047, n02088364, n02094433, n02097658, n02089078, n02090622, n02095314, n02102040, n02097130, n02096051, n02098105, n02095889, n02100236, n02099267, n02102318, n02097474, n02090721, n02102973, n02095570, n02091635, n02099429, n02090379, n02094258, n02100583, n02092002, n02093428, n02098413, n02097298, n02093754, n02096177, n02091032, n02096437, n02087394, n02092339, n02099712, n02088632, n02093647, n02098286, n02096585, n02093991, n02100877, n02094114, n02101388, n02089973, n02088094, n02088466, n02093859, n02088238, n02102480, n02101556, n02089867, n02099601, n02102177, n02101006, n02091134, n02100735, n02099849, n02093256, n02097209, n02091467, n02091244, n02096294

Birds (n1503061): n01855672, n01560419, n02009229, n01614925, n01530575, n01798484, n02007558, n01860187, n01820546, n01817953, n01833805, n02058221, n01806567, n01558993, n02056570, n01797886, n02018207, n01828970, n02017213, n02006656, n01608432, n01818515, n02018795, n01622779, n01582220, n02013706, n01534433, n02027492, n02012849, n02051845, n01824575, n01616318, n02002556, n01819313, n01806143, n02033041, n01601694, n01843383, n02025239, n02002724, n01843065, n01514859, n01796340, n01855032, n01580077, n01807496, n01847000, n01532829, n01537544, n01531178, n02037110, n01514668, n02028035, n01795545, n01592084, n01518878, n01829413, n02009912, n02011460

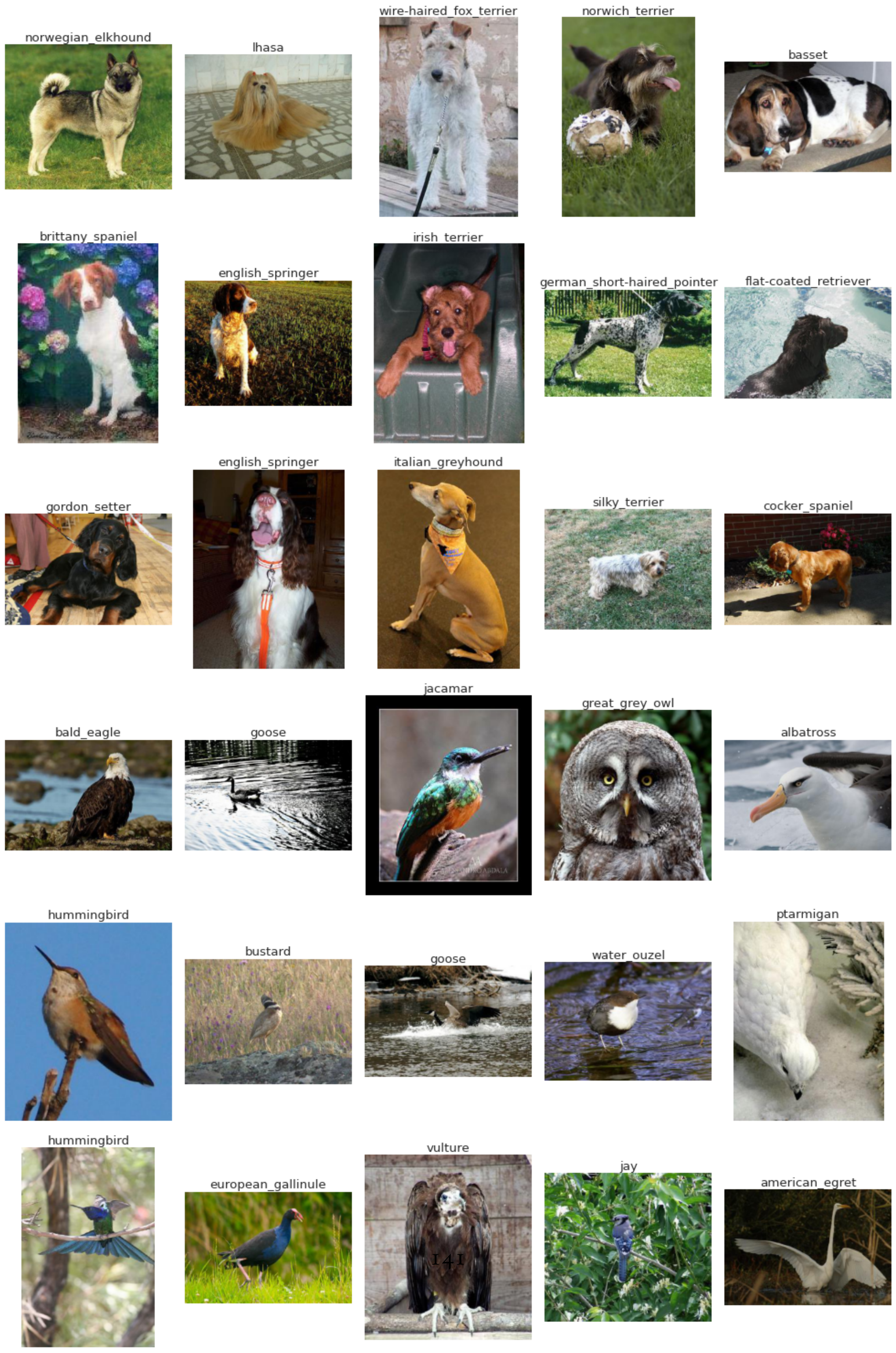


Figure C.15: ImageNet-DogBird Samples. Random samples from each class. Annotated by their original ImageNet class for reference.

D

Distributional Generalization

D.1 Student-Teacher Indistinguishability

Here we show another instance of the Indistinguishability Conjecture, by giving another way in which the distributions $\mathcal{D}_{\text{te}}$ and $\mathcal{D}$ are close – specifically, we claim they are roughly indistinguishable with respect to *training*. That is, training a student-network on samples from $(x, y) \sim \mathcal{D}$ yields a similar model as training on pseudo-labeled samples $(x, f(x)) \sim \mathcal{D}_{\text{te}}$, as long as the student is "weaker" than the teacher f.

We specifically consider a setup where a teacher network is trained on n samples, and a student network is trained on $k \ll n$ samples. In this setting, we claim that teacher-labeled samples are "as good as" real samples to the student – in that the student achieves similar test accuracy whether trained on real or pseudo-labeled samples. (In practice we find that $k \leq n/2$ is sufficient, though this limit does not appear to be fundamental.)

To see why this may be surprising, consider a setting where we use a teacher that is only only say 80% accurate. Now, we may expect that training a student on the pseudo-labeled distribution will always be worse than training on the true labels. After all, the pseudo-labels are only 80% correct. However, we find that when the student is trained on less than half the number samples than the teacher was trained on, the student does just as well as if it were trained on true labels.

Experiments. We use a ResNet18 for all student and teacher models. In Figure D.1, we use a fixed teacher network trained on $n \in \{5000, 10000\}$ samples from CIFAR-10. For each k, we compare the test error of the following two networks:

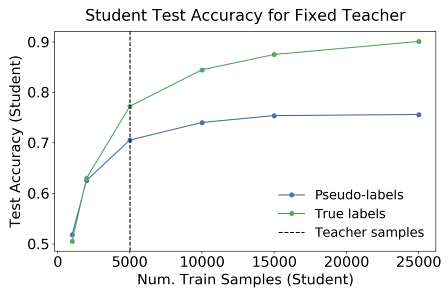
1. A student ResNet18 on trained on k pseudo-labeled samples (call this model $G_{n,k}$: a student trained on k pseudo-samples from a teacher trained on n samples).
2. A ResNet18 trained on k true samples.

When $k \leq n/2$, the test error of the student is close to that of a network trained on true samples. When $k \gg n/2$ however, the student can distinguish whether it is being trained on real or pseudo-labeled samples.

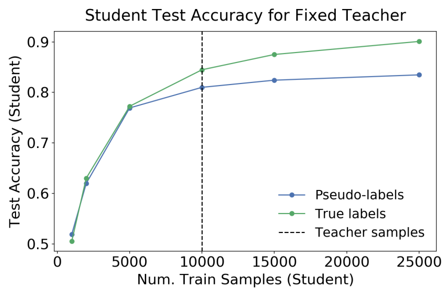
Figure D.2 shows test errors of $G_{n,k}$ for all $n, k \in \{1000, 2000, 5000, 10000, 15000, 25000\}$. The test error of $G_{n,k}$ appears to depend only on $\min(n, k)$ – intuitively, the test error is bottle-necked by the minimum power of student and teacher.

Discussion Previous sections considered tests which were asked to distinguish the distributions $\mathcal{D}$ and $\mathcal{D}_{\text{te}}$ based on a single sample from either distribution. Here, we consider a more powerful test, which is given access to k iid samples from either $\mathcal{D}$ or $\mathcal{D}_{\text{te}}$.[*] This student-teacher indistinguishability is also essentially equivalent to the following claim: We cannot learn a ResNet-

[*]This is not fundamentally different from a single sample test, via a hybrid argument.



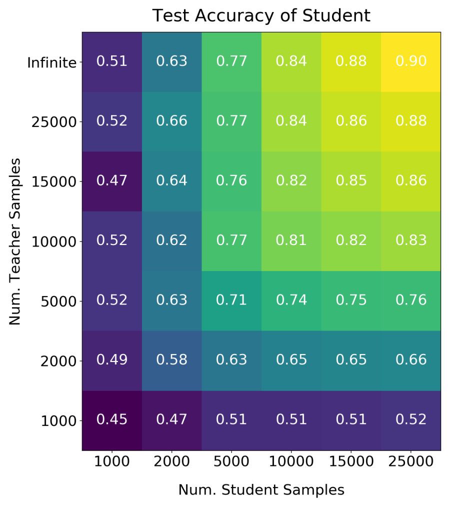
(a) Teacher trained on 5000 samples.



(b) Teacher trained on 10000 samples.

Figure D.1: Pseudo-labeling. Accuracy of student when trained on true labels vs. pseudo-labels.

distinguisher between distributions $\mathcal{D}$ and $\mathcal{D}_{\text{te}}$, given $k \leq n/2$ samples from each distribution.



Num. Teacher Samples	1000	2000	5000	10000	15000	25000
Infinite	0.51	0.63	0.77	0.84	0.88	0.90
25000	0.52	0.66	0.77	0.84	0.86	0.88
15000	0.47	0.64	0.76	0.82	0.85	0.86
10000	0.52	0.62	0.77	0.81	0.82	0.83
5000	0.52	0.63	0.71	0.74	0.75	0.76
2000	0.49	0.58	0.63	0.65	0.65	0.66
1000	0.45	0.47	0.51	0.51	0.51	0.52

Figure D.2: Test error of a student trained on k pseudo-labeled samples (x-axis) from a teacher trained on n samples (y-axis). The top row ($n = \infty$) shows a student trained on true samples from the distribution.

D.2 Experimental Details

Here we describe general background, and experimental details common to all sections. Then we provide section-specific details below.

D.2.1 Datasets

We consider the image datasets CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), MNIST (LeCun, Bottou, Bengio, and Haffner, 1998), Fashion-MNIST (Xiao, Rasul, and Vollgraf, 2017), CelebA (Liu et al., 2015), and ImageNet (Russakovsky et al., 2015).

We also consider tabular datasets from the UCI repository Dua and Graff (2017). For UCI data, we consider the 121 classification tasks as standardized in Fernández-Delgado, Cernadas, Barro, and Amorim (2014). Some of these tasks have very few examples, so we restrict to the 92 classification tasks from Fernández-Delgado et al. (2014) which have at least 200 total examples.

D.2.2 Models

We consider neural-networks, kernel methods, and decision trees.

Decision Trees

We train interpolating decision trees using a growth rule from Random Forests (Breiman, 2001; Ho, 1995): selecting a split based on a random $\sqrt{d}$ subset of d features, splitting based on Gini impurity, and growing trees until all leafs have a single sample. This is as implemented by Scikit-learn Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay (2011) defaults with `RandomForestClassifier(n_estimators=1, bootstrap=False)`.

Kernels

Throughout this work we consider classification via kernel regression and kernel SVM. For M-class classification via kernel regression, we follow the methodology in e.g. Rahimi and Recht (2007); Belkin et al. (2018b); Shankar et al. (2020). We solve the following convex problem for training:

$$\alpha^* := \operatorname*{argmin}_{\alpha \in \mathbb{R}^{N\times M}} ||K\alpha - y||_2^2 + \lambda \alpha^T K \alpha$$

where $K_{ij} = k(x_i, x_j)$ is the kernel matrix of the training points for a kernel function k, $y \in \mathbb{R}^{N\times M}$ is the one-hot encoding of the train labels, and $\lambda \geq 0$ is the regularization parameter. The solution can be written

$$\alpha^* = (K + \lambda I)^{-1} y$$

which we solve numerically using SciPy `linalg.solve` (Virtanen, Gommers, Oliphant, Haberland, Reddy, Cournapeau, Burovski, Peterson, Weckesser, Bright, van der Walt, Brett, Wilson, Jarrod Millman, Mayorov, Nelson, Jones, Kern, Larson, Carey, Polat, Feng, Moore, Vand erPlas, Laxalde, Perktold, Cimrman, Henriksen, Quintero, Harris, Archibald, Ribeiro, Pedregosa, van Mulbregt, and Contributors, 2020). We use the explicit form of all kernels involved. That is, we do not use random-feature approximations (Rahimi and Recht, 2007), though we expect they would behave similarly.

The kernel predictions on test points are then given by

$$g_\alpha(x) := \sum_{i \in [N]} \alpha_i k(x_i, x) \tag{D.1}$$

$$f_\alpha(x) := \operatorname*{argmax}_{j \in [M]} g_\alpha(x)_j \tag{D.2}$$

where $g(x) \in \mathbb{R}^M$ are the kernel regressor outputs, and $g(x) \in [M]$ is the thresholded classification decision. This is equivalent to training M separate binary regressors (one for each label), and taking the argmax for classification. We usually consider *unregularized* regression ($\lambda = 0$), except in Section 6.6.

For kernel SVM, we use the implementation provided by Scikit-learn (Pedregosa et al., 2011) `sklearn.svm.SVC` with a precomputed kernel, for inverse-regularization parameter $C \geq 0$ (larger C corresponds to smaller regularization).

Types of Kernels. We use the following kernel functions $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$.

- Gaussian Kernel (RBF): $k(x_i, x_j) = \exp(-\frac{||x_i - x_j||_2^2}{2\tilde{\sigma}^2})$.
- Laplace Kernel: $k(x_i, x_j) = \exp(-\frac{||x_i - x_j||_2}{\tilde{\sigma}})$.
- Myrtle10 Kernel: This is the compositional kernel introduced by Shankar et al. (2020). We use their exact kernel for CIFAR-10.

For the Gaussian and Laplace kernels, we parameterize bandwidth by $\sigma := \tilde{\sigma}/\sqrt{d}$. We use the following bandwidths, found by cross-validation to maximize the unregularized test accuracy:

- MNIST: $\sigma = 0.15$ for RBF kernel.
- Fashion-MNIST: $\sigma = 0.1$ for RBF kernel. $\sigma = 1.0$ for Laplace kernel.
- CIFAR-10: Myrtle10 Kernel from Shankar et al. (2020), and $\sigma = 0.1$ for RBF kernel.

Neural Networks

We use 4 different neural networks in our experiments. We use a multi-layer perceptron, and three different Residual networks.

MLP: We use a Multi-layer perceptron or a fully connected network with 3 hidden layers with 512 neurons in each layer. A hidden layer is followed by a BatchNormalization layer and ReLU activation function.

WideResNet: We use the standard WideResNet-28-10 described in Zagoruyko and Komodakis (2016). Our code is based on this repository.

	MLP	ResNet18	WideResNet-28-10	ResNet50
Batchsize	128	128	128	32
Epochs	820	200	200	50
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)	SGD + Momentum (0.9)	SGD + Momentum (0.9)	SGD
Learning rate (LR) schedule	Constant LR $= 0.001$	Inital LR$= 0.05$ scale by 0.1 at epochs $(80, 120)$	Inital LR$= 0.1$ scale by 0.2 at epochs $(80, 120, 160)$	Initial LR $= 0.001$, scale by 0.1 if training loss stagnant for 2000 gradient steps
Data Augmentation	Random flips + RandomCrop(32, padding=4)			
CIFAR-10 Error	$\sim 40\%$	$\sim 8\%$	$\sim 4\%$	N/A

Table D.1: Hyperparameters used to train the neural networks and their errors on the unmodified CIFAR-10 dataset

ResNet50: We use a standard ResNet-50 from the PyTorch library (Paszke et al., 2017).

ResNet18: We use a modification of ResNet18 He et al. (2016a) adapted to CIFAR-10 image sizes. Our code is based on this repository.

For Experiment 1 and 2 and Section 6.3, the hyperparameters used to train the above networks are given in Table D.1.

D.3 Feature Calibration: Appendix

D.3.1 A guide to reading the plots

All the experiments in support of Conjecture 1 (experiments in Section 6.3 and the Introduction) involve various quantities which we enumaerate here

1. Inputs x: Each experiment involves inputs from a standard dataset like CIFAR-10 or MNIST. We use the standard train/test splits for every dataset.
2. Distinguishable feature $L(x)$: This feature depends only on input x. We consider various features like the original classes itself, a superset of classes (as in coarse partition) or some secondary attributes (like the binary attributes provided with CelebA)
3. Output labels y: The output label may be some modification of the original labels. For instance, by adding some type of label noise, or a constructed binary task as in Experiment 1
4. Classifier family F: We consider various types of classifiers like neural networks trained with gradient based methods, kernel and decision trees.

In each experiment, we are interested in two joint densities $(y, L(x))$, which depends on our dataset and task and is common across train and test, and $(f(x), L(x))$ which depends on the interpolating classifiers outputs on the *test* set. Since $y, L(x)$ and $f(x)$ are discrete, we will look at their discrete joint distributions. We sometimes refer to $(y, L(x))$ as the train joint density, as at interpolation $(y, L(x)) = (f(x), L(x))$ for all training inputs x. We also refer to $(f(x), L(x))$ as the test density, as we measure this only on the test set.

D.3.2 Experiment 1

Experimental details: We now provide further details for Experiment 1. We first construct a dataset from CIFAR-10 that obeys the joint density $(y, L(x))$ shown in Figure 6.1 left panel. We then

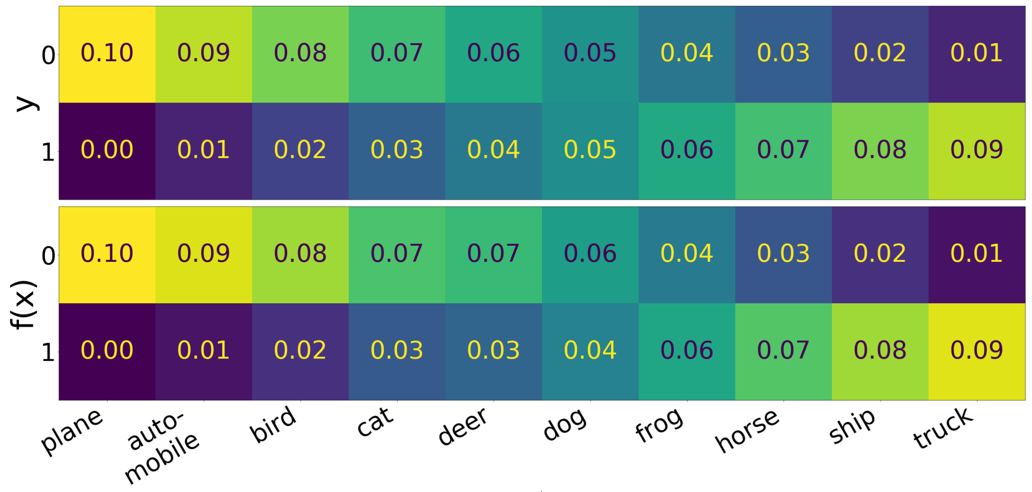


Figure D.3: **Distributional Generalization in Experiment 2.** Joint densities of the distributions involved in Experiment 2. The top panel shows the joint density of labels on the train set: $(\texttt{CIFAR\_Class(x)}, y)$. The bottom panels shows the joint density of classifier predictions on the test set: $(\texttt{CIFAR\_Class(x)}, f(x))$. Distributional Generalization claims that these two joint densities are close.

train a WideResNet-28-10 (WRN-28-10) on this modified dataset to zero training error. The network is trained with the hyperparameters described in Table D.1. We then observe the joint density $(f(x), L(x))$ on the test images and find that the two joint densities are close as shown in Figure 6.1.

We now consider a modification of this experiment as follows:

Experiment 2. *Consider the following distribution over images x and binary labels y. Sample x as a uniformly random CIFAR-10 image, and sample the label as* $p(y|x) = \mathit{Bernoulli}(\texttt{CIFAR\_Class(x)}/10)$. *That is, if the CIFAR-10 class of x is* $k \in \{0, 1, \ldots 9\}$*, then the label is* 1 *with probability* $(k/10)$ *and* 0 *otherwise. Figure D.3 shows this joint distribution of* (x, y)*. As before, train a WideResNet to 0 training error on this distribution.*

In this experiment too, we observe that the train and test joint densities are close as shown in Figure D.3.

Now, we repeat the same experiment, but with an MLP instead of WRN-28-10. The training procedure is described in Table D.1. This MLP has an error on 37% on the original CIFAR-10 dataset.

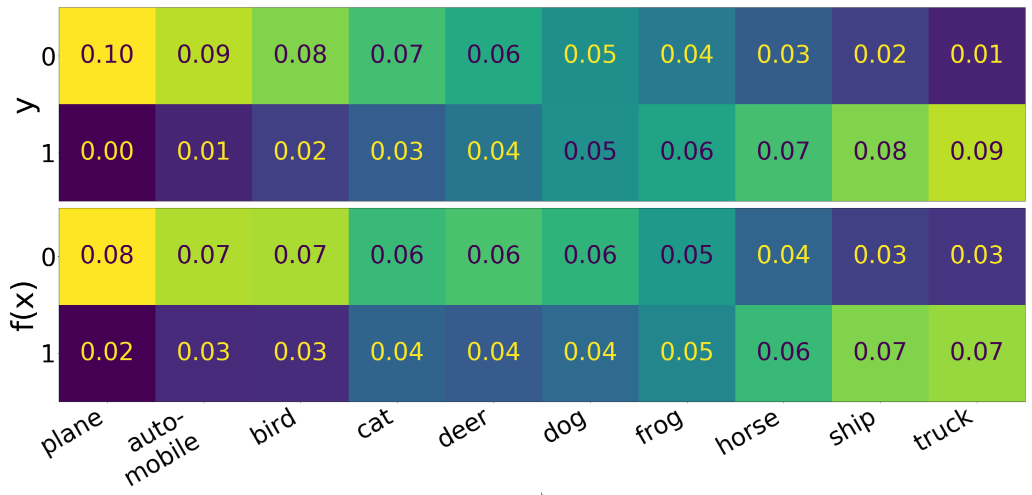


Figure D.4: Joint density of $(y, \text{Class}(x))$, top, and $(f(x), \text{Class}(x))$, bottom, for test samples (x, y) from Experiment 2 for an MLP.

Since this MLP has poor accuracy on the original CIFAR-10 classification task, it does not form a distinguishable partition for it. As a result, the train and test joint densities (Figure D.4) do not match as well as they did for WRN-28-10.

D.3.3 Constant Partition

We now describe the experiment for a constant partition $L(x) = 0$. For this experiment, we first construct a dataset based on CIFAR-10 that has class-imbalance. For class $k \in \{0 \dots 9\}$, sample $(k+1) \times 500$ images from that class. This will give us a dataset where classes will have marginal distribution $p(y = \ell) \propto \ell + 1$ for classes $\ell \in [10]$, as shown in Figure 6.2. We do this both for the training set and the test set, to keep the distribution $\mathcal{D}$ fixed.

Now, we train the MLP, ResNet-18 and RBF Kernel on this dataset. We plot the resulting $p(f(x))$ for each of these models below. That is, we plot the fraction of test images for which the network outputs $\{0, 1 \dots 9\}$ respectively. As predicted, the networks closely track the train set.

D.3.4 Class Partition

Neural Networks and CIFAR-10

We now provide experiments in support of Conjecture 1 when the class itself is a distinguishable partition. We know that WRN-28-10 achieves an error of 4% on this dataset. Hence, the original labels in CIFAR-10 form a distinguishable partition for this dataset. To demonstrate that Conjecture 1 holds, we consider different structured label noise on the CIFAR-10 dataset. To do so, we apply a variety of confusion matrices to the data. That is, for a confusion matrix $C : 10 \times 10$ matrix, the element c_{ij} gives the joint density that a randomly sampled image had original label j, but is flipped to class i. For no noise, this would be an identity matrix.

We begin by a simple confusion matrix where we flip only one class $0 \to 1$ with varying probability p. Figure 6.3 shows one such confusion matrix for $p = 0.4$. We then train a WideResNet-28-10 to zero train error on this dataset. We use the hyperparameters described in D.2.2 We find that the classifier outputs on the test set closely track the confusion matrix that was applied to the distribution. Figure 6.3 shows that this is independent of the value of p and continues to hold for $p \in [0, 1]$.

To show that this is not dependent on the particular class used, we also show that the same holds for a random confusion matrix. We generate a sparse confusion matrix as follows. We set the diagonal to 0.5. Then, for every class j, we pick any two random classes for and set them to 0.2 and 0.3. We train a WRN-28-10 on it and report the test confusion matrix. The resulting train and test densities are shown in Figure 6.4 and also below for clarity.

Decision Trees

Similar results hold for decision trees; here we show experiments on two UCI tasks: `wine` and `mushroom`.

The `wine` task is a 3-way classification problem: to identify the cultivar of a given wine (out of 3 cultivars), given 13 physical attributes describing the wine. Figure D.5 shows an analogous experiment with label noise taking class 1 to class 2.

The `mushroom` task is a 2-way classification problem: to classify the type of edibility of a mushroom (edible vs poisonous) given 22 physical attributes (e.g. stalk color, odor, etc). Figure D.6 shows an analogous experiment with label noise flipping class 0 to class 1.

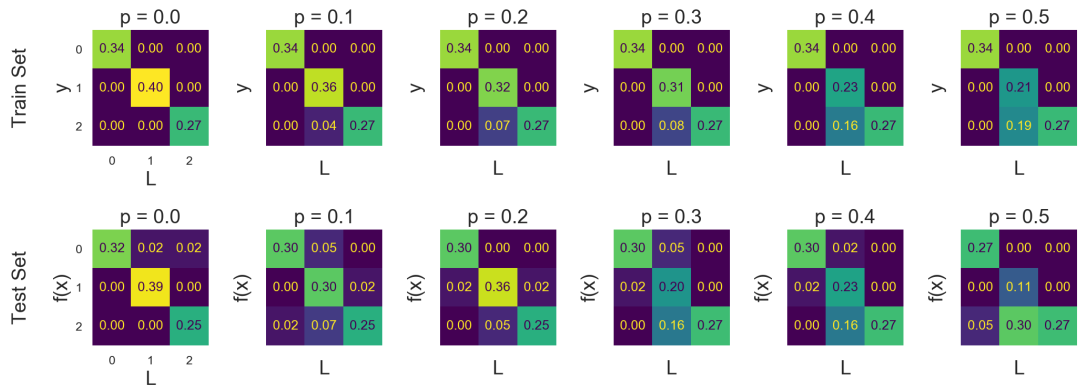


Figure D.5: Decision trees on UCI (wine). We add label noise that takes class 1 to class 2 with probability $p \in [0, 0.5]$. Each column shows the test and train confusion matrices for a given p. Note that this decision trees achieve high accuracy on this task with no label noise (leftmost column). We plot the empirical joint density of the train set, and not the population joint density of the train distribution, and thus the top row exhibits some statistical error due to small-sample effects.

D.3.5 Multiple Features

For the CelebA experiments, we train a ResNet-50 to predict the attribute {Attractive, Not Attractive}. We choose this attribute because a ResNet-50 performs poorly on this task (test error $\sim 20\%$) and has good class balance. We choose an attribute with poor generalization because the conjecture would hold trivially for if the network generalizes well. We initialize the network with a pretrained ResNet-50 from the PyTorch library Paszke et al. (2017) and use the hyperparameters described in Section D.2.2 to train on this attribute. We then check the train/test joint density with various other attributes like Male, Wearing Lipstick etc. Note that the network is not given any label information for these additional attributes, but is calibrated with respect to them. That is, the network

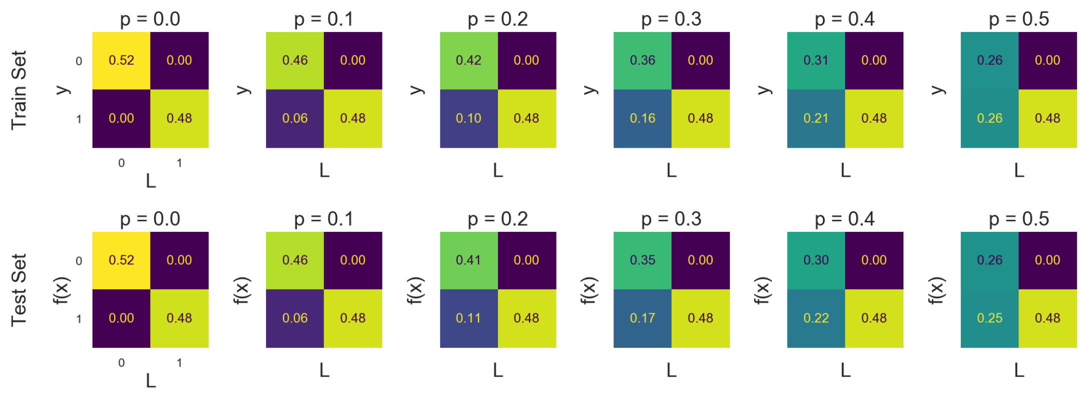


Figure D.6: Decision trees on UCI (mushroom). We add label noise that takes class 1 to class 2 with probability $p \in [0, 0.5]$. Each column shows the test and train confusion matrices for a given p. Note that this decision trees achieve high accuracy on this task with no label noise (leftmost column).

says $\sim 30\%$ of images that have 'Heavy Makeup' will be classified as 'Attractive', even if the network makes mistakes on which particular inputs it chooses to do so. Loosely, this can be viewed as the network performing 1-Nearest-Neighbor classification in a metric space that is well separated for each of these distinguishable features.

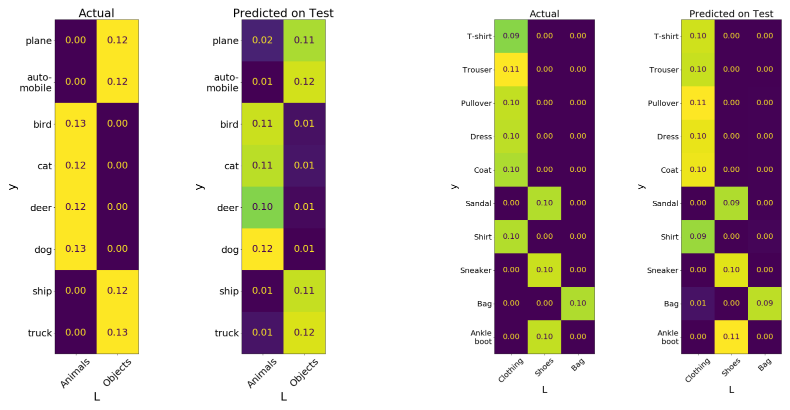
D.3.6 Coarse Partition

We now consider cases where the original classes do not form a distinguishable partition for the classifier in consideration. That is, the classifier is not powerful enough to obtain low error on the original dataset, but can perform well on a coarser division of the classes.

To verify this, we consider a division of the CIFAR-10 classes into Objects {airplane, automobile, ship, truck} vs Animals {cat, deer, dog, frog}. An MLP trained on this problem has low error ($\sim 8\%$), but the same network performs poorly on the full dataset ($\sim 37\%$ error). Hence, Object vs Animals forms a distinguishable partition with MLPs. In Figure D.7a, we show the results of training an MLP on the original CIFAR-10 classes. We see that the network mostly classifies objects as objects and animals as animals, even when it might mislabel a dog for a cat.

Model	AlexNet	ResNet18	ResNet50	BagNet8	BagNet32
ImageNet Accuracy	0.565	0.698	0.761	0.464	0.667
Accuracy on dogs	0.588	0.729	0.793	0.462	0.701
Accuracy on terriers	0.572	0.704	0.775	0.421	0.659
Accuracy for binary {dog/not-dog}	0.984	0.993	0.996	0.972	0.992
Accuracy on {terrier/not-terrier} among dogs	0.913	0.955	0.969	0.876	0.944
Fraction of real-terriers among dogs	0.224	0.224	0.224	0.224	0.224
Fraction of predicted-terriers among dogs	0.209	0.222	0.229	0.192	0.215

Table D.2: ImageNet classifiers are calibrated with respect to dogs: All classifiers predict terrier for roughly $\sim 22\%$ of all dogs (last row), though they may mistake which specific dogs are terriers.



(a) CIFAR10 + MLP **(b)** Fashion-MNIST + RBF

Figure D.7: Coarse partitions as distinguishable features: We consider a setting where the original classes are not distinguishable, but the a superset of the classes are.

We perform a similar experiment for the RBF kernel on Fashion-MNIST, with partition {clothing, shoe, bag}, in Figure D.7b.

ImageNet experiment. In Table D.2 we provide results of the terrier experiment in the body, for various ImageNet classifiers. We use publicly available pretrained ImageNet models from this repository, and use their evaluations on the ImageNet test set.

D.3.7 Pointwise Density Estimation

Here we describe an informal version of the conditional-density estimation property, and give preliminary experimental evidence to support it. We do not yet understand this property deeply enough to state it formally, but we believe this informal version captures the essential behavior.

Conjecture 4 (Conditional Density Estimation, Informal). *For all distributions $\mathcal{D} \equiv p(x, y)$, number of samples n, family of models $\mathcal{F}$, and $\varepsilon > 0$, let L_{fine} be a "finest distinguishable partition" — that is, informally an $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$-distinguishable partition that cannot be refined further. Then:*

$$\textit{With high probability over } x \sim \mathcal{D}: \quad \{f(x)\}_{f \leftarrow \mathrm{Train}_{\mathcal{F}}(\mathcal{D}^n)} \approx_{\varepsilon} p(y|L_{fine}(x))$$

Conjecture 4 is a *pointwise* version of Conjecture 1: it says that for most inputs x, we can sample from the conditional density $p(y|L_{\text{fine}}(x))$ by training a fresh classifier f on iid samples from $\mathcal{D}$, and outputting $f(x)$. We think of $L_{\text{fine}}(x)$ as the "local neighborhood of x", or the finest class of x as distinguishable by neural networks. We would ideally like to sample from $p(y|x)$, but the best we can hope for is to sample from $p(y|L_{\text{fine}}(x))$.

Note that this pointwise conjecture must neccesarily invoke a notion of "finest distinguishable partition", which we do not formally define, while Conjecture 1 applied to all distinguishable partitions.

Experiment: Train-set Ensemble

We now give preliminary evidence for this pointwise density estimation. Consider the following simple distribution: MNIST with label noise that flips class 0 to class 1 with probability 40%. To check the pointwise conjecture, we would like to train an ensemble of classifiers $f_i \leftarrow \mathrm{Train}(\mathcal{D}^n)$ on fresh samples from this distribution. We do not have sufficient samples to use truly independent samples, so we approximate this as follows:

1. Sample 5k random examples from the MNIST-train set.
2. Add independent label noise (flipping 0 to 1 w.p. 40%).
3. Train a Gaussian kernel interpolating classifier on these noisy samples (via the hyperparameters in Appendix D.2).

We train 100 such classifiers, and let $\{f_i\}$ be the ensemble. Then, we claim that if x_0 is a digit 0 in the test set, the empirical distribution $\{f_i(x_0)\}$ over the ensemble is roughly 60% label-0 and 40% label-1. More generally, for all test points x, the distribution $\{f_i(x)\}$ should be close in total variation distance to $p(y|x)$. In Figure D.8 we plot the histogram of this TV distance for all x in the test set

$$H(x) := TV(\ \{f_i(x)\} \ , \ p(y|x) \ )$$

and we see that $H(x)$ is concentrated at 0, with $\mathbb{E}_x[H(x)] \approx 0.036$.

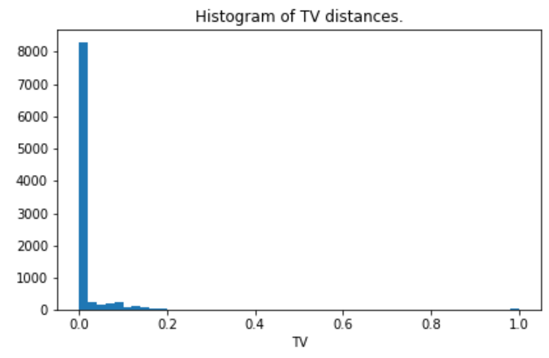


Figure D.8: MNIST Ensemble.

Discussion. Here we obtained a conditional density estimate of $p(y|x)$ by training an ensemble of kernel classifiers on (approximately-) independent train sets. In fact, we observed that for some classifiers we can *re-use* the same train set, and get the same behavior – that is, training an ensemble with fresh random initialization alone. In these cases, randomness in the training procedure is somehow able to substitute for randomness in the sampling procedure. This cannot hold for deterministic training procedures, such as kernel regression, but we have observed it for neural-networks and decision trees. The corresponding statement about decision trees is implicit in works on the conditional-density-estimation properties of random forests (e.g. Olson and Wyner (2018)).

D.4 Agreement Property: Appendix

D.4.1 Experimental Details

For ResNets on CIFAR-10 and CIFAR-100, we use the following training procedure. For $n \leq 25000$, we sample two disjoint train sets S_1, S_2 of size n from the 50K total train samples. Then we train two ResNet18s f_1, f_2 on S_1, S_2 respectively. We optimize using SGD on the cross-entropy loss, with batch size 128, using learning rate schedule 0.1 for $40\lfloor \frac{50000}{n} \rfloor$ epochs, then 0.01 for $20\lfloor \frac{50000}{n} \rfloor$ epochs. That is, we scale up the number of epoches for smaller train sizes, to keep the number of gradient steps constant. We also early-stop optimization when the train loss reaches < 0.0001, to save computational time. For experiments with data-augmentation, we use horizontal flips and `RandomCrop(32, padding=4)`. We estimate test accuracy and agreement probability on the CIFAR-10/100 test sets.

For the kernel experiments on Fashion-MNIST, we repeat the same procedure: we sample two disjoint train sets from all the train samples, train kernel regressors, and evaluate their agreement on the test set. Each point on the figures correspond to one trial.

For UCI, some UCI tasks have very few examples, and so here we consider only the 92 classification tasks from Fernández-Delgado et al. (2014) which have at least 200 total examples. For each task, we randomly partition all the examples into a 40%-40%-20% split for 2 disjoint train sets, and 1 test set (20%). We then train two interpolating decision trees, and compare their performance on the test set. Decision trees are trained using a growth rule from Random Forests (Breiman, 2001; Ho, 1995): selecting a split based on a random $\sqrt{d}$ subset of d features, splitting based on Gini impurity, and growing trees until all leafs have a single sample. This is as implemented by Scikit-learn Pedregosa et al. (2011) defaults with

`RandomForestClassifier(n_estimators=1, bootstrap=False)`.

In Figure 6.8c of the body, each point corresponds to one UCI task, and we plot the means of agreement probability and test accuracy when averaged over 100 random partitions for each task. Figure D.9 shows the corresponding plot for a single trial.

D.4.2 Additional Plots

Figure D.10 shows the Laplace Kernel on Fashion-MNIST.

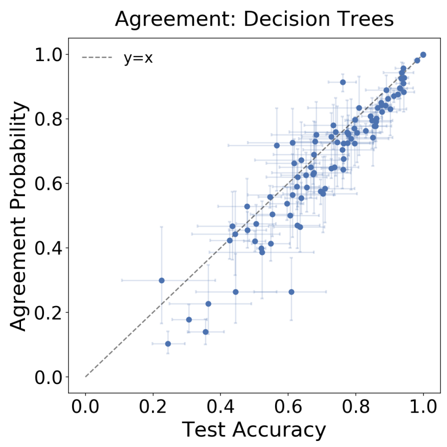


Figure D.9: Agreement Probability for a single trial of UCI classification tasks. Analogous to Figure 6.8c in the body, for a single trial.

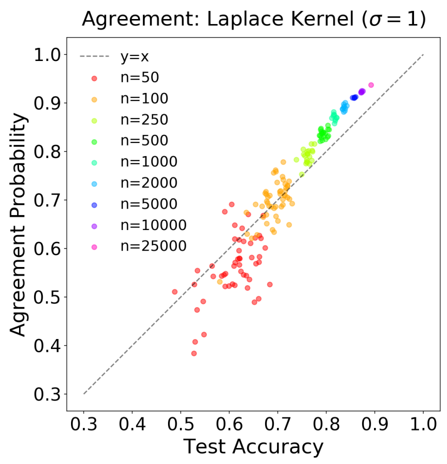


Figure D.10: Laplace Kernel on Fashion-MNIST

D.4.3 Alternate Mechanisms

We would like to understand the distribution of $f \leftarrow \text{Train}(\mathcal{D}^n)$, in order to evaluate the proposed mechanisms in Sections 6.4.2 and 6.4.2. Technically, sampling from this distribution requires training a classifier on a *fresh* train set. Since we do not have infinite samples for CIFAR-10, we construct empirical estimates by training an ensemble of classifiers on random subsets of CIFAR-10. Then, to approximate a sample $f \leftarrow \text{Train}(\mathcal{D}^n)$, we simply sample from our ensemble $f \leftarrow \{f_i\}_i$.

Bimodal Samples

Figure D.11 shows a histogram of

$$h(x) := \Pr_{f \leftarrow \text{Train}(\mathcal{D}^n)}[f(x) = y]$$

for test samples x in CIFAR-10, where f is a ResNet18 trained on 5000 samples[†]. This quantity can be interpreted as the "easiness" of a given test sample (x, y) to a certain classifier family.

If the EASY/HARD bimodal model were true, we would expect the distribution of $h(x)$ to be concentrated on $h(x) = 1$ (easy samples) and $h(x) = 0.1$ (hard samples). But this is not the case in Figure D.11.

[†]We estimate this probability over the empirical ensemble $f \leftarrow \{f_i\}_i$, where each f_i is a classifier trained on a random $5k$-subset of CIFAR-10. We train 100 classifiers in this ensemble.

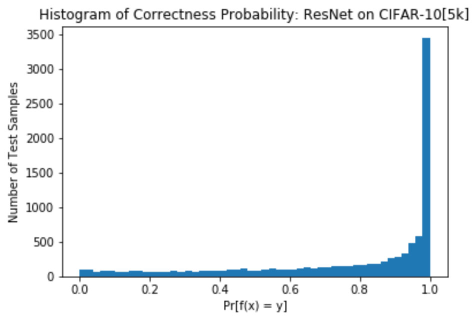


Figure D.11: Histogram of sample-hardnesses.

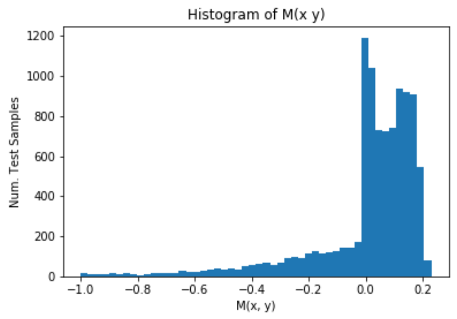


Figure D.12: Pointwise agreement histogram.

Pointwise Agreement

We could more generally posit that Conjecture 2 is true because the Agreement Property holds *pointwise* for most test samples x:

$$\text{w.h.p. for } (x,y) \sim \mathcal{D} : \quad \Pr_{f_1 \leftarrow \mathrm{Train}(\mathcal{D}^n)} [f_1(x) = y] \approx \Pr_{\substack{f_1 \leftarrow \mathrm{Train}(\mathcal{D}^n) \\ f_2 \leftarrow \mathrm{Train}(\mathcal{D}^n)}} [f_1(x) = f_2(x)] \tag{D.3}$$

However, we find (perhaps surprisingly) that this is not the case. To see why this is surprising, observe that Conjecture 2 implies that the agreement probability is close to test accuracy, *in expectation* over the test sample and the classifiers $f_1, f_2 \leftarrow \mathrm{Train}(\mathcal{D}^n)$:

$$\Pr_{\substack{f_1 \\ (x,y)\sim\mathcal{D}}} [f_1(x) = y] \approx \Pr_{\substack{f_1,f_2 \\ (x,y)\sim\mathcal{D}}} [f_1(x) = f_2(x)] \tag{Conjecture 2}$$

$$\iff \mathop{\mathbb{E}}_{f_1} \mathop{\mathbb{E}}_{x,y\sim\mathcal{D}} [1\{f_1(x) = y\}] \approx \mathop{\mathbb{E}}_{f_1,f_2} \mathop{\mathbb{E}}_{x,y\sim\mathcal{D}} [1\{f_1(x) = f_2(x)\}] \tag{D.4}$$

Swapping the order of expectation, this implies

$$\underbrace{\mathop{\mathbb{E}}_{x,y\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{f_1,f_2}[1\{f_1(x)=y\}-1\{f_1(x)=f_2(x)\}]\right]}_{M(x,y)}\approx 0 \tag{D.5}$$

Now, we may expect that this means $M(x,y) \approx 0$ pointwise, for most test samples (x,y). But this is not the case. It turns out that $M(x,y)$ takes on significantly positive and negative values, and these effects "cancel out" in expectation over the distribution, to yield Conjecture 2.

For example, we compute $M(x,y)$ for the Myrtle10 kernel on CIFAR-10 with 1000 train samples. [‡]

1. The agreement probability is within 0.8% of the test error (as in Figure 6.7c), and so

$$\mathop{\mathbb{E}}_{x,y\sim\mathcal{D}}[M(x,y)]\approx 0.008$$

2. However, $M(x,y)$ is not pointwise close to 0. E.g,

$$\mathop{\mathbb{E}}_{x,y\sim\mathcal{D}}[|M(x,y)|]\approx 0.133$$

Figure D.12 plots the distribution of $M(x,y)$. We see that some samples (x,y) have high agreement probability, and some low, and these happen to balance in expectation to yield the test accuracy.

[‡]We estimate the expectation in $M(x,y)$ by training an ensemble of 5000 pairs of classifiers (f_1, f_2), each pair on disjoint train samples.

D.5 Non-interpolating Classifiers: Appendix

Here we give an additional example of distributional generalization: in kernel SVM (as opposed to kernel regression, in the main text).

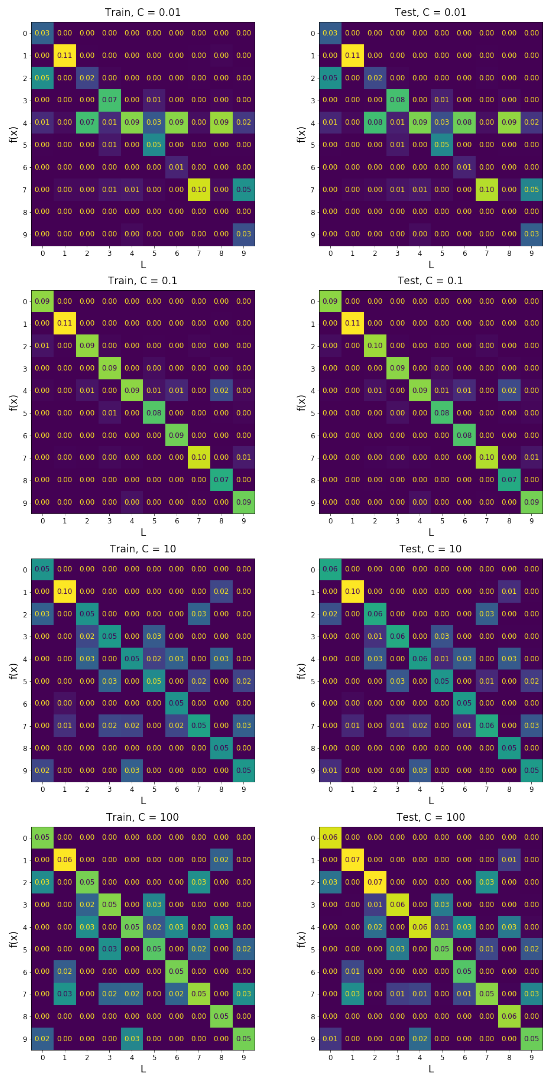


Figure D.13: Distributional Generalization. Train (left) and test (right) confusion matrices for kernel SVM on MNIST with random sparse label noise. Each row corrosponds to one value of inverse-regularization parameter C. All rows are trained on the same (noisy) train set.

D.6 Nearest-Neighbor Proofs

D.6.1 Agreement Property

Theorem 2 (Agreement Property). *For a given distribution $\mathcal{D}$ on (x, y), and given number of train samples $n \in \mathbb{N}$, suppose* NN *satisfies the following regularity condition: If we sample two independent train sets S_1, S_2, then the following two "couplings" are statistically close:*

$$\{(x_i, \mathrm{NN}_{S_2}(x_i))\}_{\substack{S_1 \sim \mathcal{D}^n \\ S_2 \sim \mathcal{D}^n \\ x_i \in_R S_1}} \quad \approx_\delta \quad \{(\mathrm{NN}_{S_1}(x), \mathrm{NN}_{S_2}(x))\}_{\substack{S_1 \sim \mathcal{D}^n \\ S_2 \sim \mathcal{D}^n \\ x \sim \mathcal{D}}} \tag{D.6}$$

The LHS is simply a random test point x_i, along with its nearest-neighbor in the train set. The RHS produces an (x_i, x_j) by sampling two independent train sets, sampling a test point $x \sim D$, and producing the nearest-neighbor of x in S_1 and S_2 respectively.

Then:

$$\Pr_{\substack{S \sim \mathcal{D}^n \\ (x,y) \sim \mathcal{D}}}[\mathrm{NN}_S^{(y)}(x) = y] \approx_\delta \Pr_{\substack{S_1 \sim \mathcal{D}^n \\ S_2 \sim \mathcal{D}^n \\ (x,y) \sim \mathcal{D}}}[\mathrm{NN}_{S_1}^{(y)}(x) = \mathrm{NN}_{S_2}^{(y)}(x)] \tag{D.7}$$

Proof. Let the LHS of Equation (D.6) be denoted as distribution P over $\mathcal{X} \times \mathcal{X}$. And let Q be the RHS of Equation (D.6). Let $\mathcal{D}_x$ denote the marginal distribution on x of $\mathcal{D}$, and let $p(y|x)$ denote the conditional distribution with respect to $\mathcal{D}$.

The proof follows by considering the sampling of train set S in the following order: first, sample all the x-marginals: sample test point $x \sim \mathcal{D}_x$ and train points $S_x \sim \mathcal{D}_x^n$. Then compute the nearest-neighbors $\widehat{x} \leftarrow \mathrm{NN}_{S_x}(x)$. And finally, sample the *values* y of all the points involved, according to the densities $p(y|x)$.

$$
\begin{aligned}
\Pr_{\substack{S\sim\mathcal{D}^n\\(x,y)\sim\mathcal{D}}}[\mathrm{NN}_S^{(y)}(x)=y] &= \mathop{\mathbb{E}}_{\substack{S\sim\mathcal{D}^n\\(x,y)\sim\mathcal{D}}}[1\{\mathrm{NN}_S^{(y)}(x)=y\}] && \text{(D.8)}\\
&= \mathop{\mathbb{E}}_{\substack{S_x\sim\mathcal{D}_x^n\\x\sim\mathcal{D}_x\\\widehat{x}\leftarrow\mathrm{NN}_{S_x}(x)}} \underbrace{\left[\mathop{\mathbb{E}}_{\substack{y\sim p(y|x)\\\widehat{y}\sim p(y|\widehat{x})}}[1\{\widehat{y}=y\}]\right]}_{T(x,\widehat{x})} && \text{(D.9)}\\
&= \mathop{\mathbb{E}}_{\substack{S_x\sim\mathcal{D}_x^n\\x\sim\mathcal{D}_x\\\widehat{x}\leftarrow\mathrm{NN}_{S_x}(x)}}[T(x,\widehat{x})] && \text{(D.10)}\\
&= \mathop{\mathbb{E}}_{(x_1,x_2)\sim P} T(x_1,x_2) && \text{(P: LHS of Equation (D.6))}\\
&\approx_\partial \mathop{\mathbb{E}}_{(x_1,x_2)\sim Q} T(x_1,x_2) && \text{(Q: RHS of Equation (D.6))}\\
&= \mathop{\mathbb{E}}_{\substack{S_1\sim\mathcal{D}_x^n\\S_2\sim\mathcal{D}_x^n\\x\sim\mathcal{D}_x\\\widehat{x}_1\leftarrow\mathrm{NN}_{S_1}(x)\\\widehat{x}_2\leftarrow\mathrm{NN}_{S_2}(x)}}[T(\widehat{x}_1,\widehat{x}_2)] && \text{(D.11)}\\
&= \mathop{\mathbb{E}}_{\substack{S_1\sim\mathcal{D}_x^n\\S_2\sim\mathcal{D}_x^n\\x\sim\mathcal{D}_x\\\widehat{x}_1\leftarrow\mathrm{NN}_{S_1}(x)\\\widehat{x}_2\leftarrow\mathrm{NN}_{S_2}(x)}}\left[\mathop{\mathbb{E}}_{\substack{\widehat{y}_1\sim p(y|\widehat{x}_1)\\\widehat{y}_2\sim p(y|\widehat{x}_2)}}[1\{\widehat{y}_1=\widehat{y}_2\}]\right] && \text{(D.12)}\\
&= \mathop{\mathbb{E}}_{\substack{S_1\sim\mathcal{D}^n\\S_2\sim\mathcal{D}^n\\(x,y)\sim\mathcal{D}}}[1\{\mathrm{NN}_{S_1}^{(y)}(x)=\mathrm{NN}_{S_2}^{(y)}(x)\}] && \text{(D.13)}\\
&= \Pr_{\substack{S_1\sim\mathcal{D}^n\\S_2\sim\mathcal{D}^n\\(x,y)\sim\mathcal{D}}}[\mathrm{NN}_{S_1}^{(y)}(x)=\mathrm{NN}_{S_2}^{(y)}(x)] && \text{(D.14)}
\end{aligned}
$$

as desired. □

References

E. Abbe and C. Sandon. On the universality of deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e7e8f8e5982b3298c8addedf6811d500-Abstract.html.

M. S. Advani and A. M. Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6155–6166, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/62dad6e273d32235ae02b7d321578ee8-Abstract.html.

M. Anthony and P. L. Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 2018. URL http://proceedings.mlr.press/v80/arora18b.html.

S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 322–332. PMLR, 2019. URL http://proceedings.mlr.press/v97/arora19a.html.

S. Athey, J. Tibshirani, S. Wager, et al. Generalized random forests. *The Annals of Statistics*, 47(2): 1148–1178, 2019.

F. Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

Y. Bansal, P. Nakkiran, and B. Barak. Revisiting model stitching to compare neural representations. *arXiv preprint arXiv:2106.07682*, 2021.

P. L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In *Advances in neural information processing systems*, pages 134–140, 1997.

P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. In D. P. Helmbold and R. C. Williamson, editors, *Computational Learning Theory, 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, July 16-19, 2001, Proceedings*, volume 2111 of *Lecture Notes in Computer Science*, pages 224–240. Springer, 2001. doi: 10.1007/3-540-44581-1_15. URL https://doi.org/10.1007/3-540-44581-1_15.

P. L. Bartlett, V. Maiorov, and R. Meir. Almost linear vc dimension bounds for piecewise polynomial networks. In *Advances in neural information processing systems*, pages 190–196, 1999.

P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6240–6249, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/b22b257ad0519d4500539da3c8bcf4dd-Abstract.html.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 2020.

P. L. Bartlett, A. Montanari, and A. Rakhlin. Deep learning: a statistical viewpoint. *arXiv preprint arXiv:2103.09177*, 2021.

M. Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *ArXiv*, abs/2105.14368, 2021.

M. Belkin, D. J. Hsu, and P. Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2306–2317, 2018a. URL https://proceedings.neurips.cc/paper/2018/hash/e22312179bf43e61576081a2f250f845-Abstract.html.

M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 540–548. PMLR, 2018b. URL http://proceedings.mlr.press/v80/belkin18a.html.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32): 15849–15854, 2019a.

M. Belkin, D. Hsu, and J. Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019b.

J. Berner, P. Grohs, G. Kutyniok, and P. Petersen. The modern mathematics of deep learning. *arXiv preprint arXiv:2105.04026*, 2021.

K. Bibas, Y. Fogel, and M. Feder. A new look at an old problem: A universal learning approach to linear regression. *arXiv preprint arXiv:1905.04708*, 2019.

L. Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

J. Bornschein, F. Visin, and S. Osindero. Small data, big decisions: Model selection in the small-data regime. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1035–1044. PMLR, 2020. URL http://proceedings.mlr.press/v119/bornschein20a.html.

L. Bottou and Y. LeCun. Large scale online learning. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 217–224. MIT Press, 2003. URL https://proceedings.neurips.cc/paper/2003/hash/9fb7b048c96d44a0337f049e0a61ff06-Abstract.html.

O. Bousquet and A. Elisseeff. Algorithmic stability and generalization performance. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 196–202. MIT Press, 2000. URL https://proceedings.neurips.cc/paper/2000/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

L. Breiman. Reflections after refereeing papers for nips. *The Mathematics of Generalization*, pages 11–15, 1995.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

W. Brendel and M. Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkfMWhAqYQ.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

S. Bubeck. Introduction to online optimization. *Lecture Notes*, 2, 2011.

E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inf. Theory*, 52(12):5406–5425, 2006. doi: 10.1109/TIT.2006.885507. URL https://doi.org/10.1109/TIT.2006.885507.

M. Cettolo, C. Girardi, and M. Federico. WIT3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Annual conference of the European Association for Machine Translation*, pages 261–268, Trento, Italy, 2012. European Association for Machine Translation. URL https://www.aclweb.org/anthology/2012.eamt-1.60.

N. S. Chatterji and P. M. Long. Finite-sample analysis of interpolating linear classifiers in the overparameterized regime. *arXiv preprint arXiv:2004.12019*, 2020.

M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining from pixels. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20s.html.

L. Chizat and F. R. Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In J. D. Abernethy and S. Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 1305–1338. PMLR, 2020. URL http://proceedings.mlr.press/v125/chizat20a.html.

A. Degwekar, P. Nakkiran, and V. Vaikuntanathan. Computational limitations in robust classification and win-win results. In A. Beygelzimer and D. Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 994–1028, Phoenix, USA, 25–28 Jun 2019. PMLR. URL http://proceedings.mlr.press/v99/degwekar19a.html.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

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

X. Dong and Y. Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJxyZkBKDr.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In G. Elidan, K. Kersting, and A. T. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. URL http://auai.org/uai2017/proceedings/papers/173.pdf.

G. K. Dziugaite, A. Drouin, B. Neal, N. Rajkumar, E. Caballero, L. Wang, I. Mitliagkas, and D. M. Roy. In search of robust measures of generalization. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/86d7c8a08b4aaa1bc7c599473f5dddda-Abstract.html.

B. Efron. Bootstrap methods: Another look at the jackknife. *Ann. Statist.*, 7(1):1–26, 01 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

B. Efron and T. Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pages 54–75, 1986.

B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014.

E. Fix and J. Hodges. *Discriminatory analysis: nonparametric discrimination, consistency properties*. USAF School of Aviation Medicine, 1951.

S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019a.

S. Fort, P. K. Nowak, S. Jastrzebski, and S. Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019b.

C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

X. Gao, X. Li, and S. Zhang. Online learning with non-convex losses and non-stationary regret. In A. J. Storkey and F. Pérez-Cruz, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 235–243. PMLR, 2018. URL http://proceedings.mlr.press/v84/gao18a.html.

M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d'Ascoli, G. Biroli, C. Hongler, and M. Wyart. Scaling description of generalization with number of parameters in deep learning. *arXiv preprint arXiv:1901.01608*, 2019a.

M. Geiger, S. Spigler, S. d'Ascoli, L. Sagun, M. Baity-Jesi, G. Biroli, and M. Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115, 2019b.

F. Gerace, B. Loureiro, F. Krzakala, M. Mézard, and L. Zdeborová. Generalisation error in learning with random features and the hidden manifold model. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3452–3462. PMLR, 2020. URL http://proceedings.mlr.press/v119/gerace20a.html.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. In S. Bubeck, V. Perchet, and P. Rigollet, editors, *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, pages 297–299. PMLR, 2018. URL http://proceedings.mlr.press/v75/golowich18a.html.

R. Gontijo-Lopes, S. J. Smullin, E. D. Cubuk, and E. Dyer. Affinity and diversity: Quantifying mechanisms of data augmentation. *arXiv preprint arXiv:2002.08973*, 2020.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. URL http://arxiv.org/abs/1406.2661.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

S. Gunasekar, J. D. Lee, D. Soudry, and N. Srebro. Characterizing implicit bias in terms of optimization geometry. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1827–1836. PMLR, 2018a. URL http://proceedings.mlr.press/v80/gunasekar18a.html.

S. Gunasekar, J. D. Lee, D. Soudry, and N. Srebro. Implicit bias of gradient descent on linear convolutional networks. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9482–9491, 2018b. URL https://proceedings.neurips.cc/paper/2018/hash/0e98aeeb54acf612b9eb4e48a269814c-Abstract.html.

M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1225–1234. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/hardt16.html.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'10, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension bounds for piecewise linear neural networks. In S. Kale and O. Shamir, editors, *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, volume 65 of *Proceedings of Machine Learning Research*, pages 1064–1068. PMLR, 2017. URL http://proceedings.mlr.press/v65/harvey17a.html.

T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

E. Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016a. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

J. Hestness, S. Narang, N. Ardalani, G. F. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou. Deep learning scaling is predictable, empirically. *CoRR*, abs/1712.00409, 2017. URL http://arxiv.org/abs/1712.00409.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.243. URL https://doi.org/10.1109/CVPR.2017.243.

Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. X. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 103–112, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/093f65e080a295f8076b1c5722a46aa2-Abstract.html.

L. Hui and M. Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*, 2020.

L. Hui and M. Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=hsFN92eQEla.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

P. T. Inc. Collaborative data science, 2015. URL https://plot.ly.

R. Jahandideh, A. T. Targhi, and M. Tahmasbi. Physical attribute prediction using deep residual neural networks. *arXiv preprint arXiv:1812.07857*, 2018.

P. Jain and P. Kar. Non-convex optimization for machine learning. *arXiv preprint arXiv:1712.07897*, 2017.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Z. Ji and M. Telgarsky. The implicit bias of gradient descent on nonseparable data. In A. Beygelzimer and D. Hsu, editors, *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pages 1772–1798. PMLR, 2019. URL http://proceedings.mlr.press/v99/ji19a.html.

Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic generalization measures and where to find them. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1724–1732. PMLR, 2017. URL http://proceedings.mlr.press/v70/jin17a.html.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, pages 1–11, 2021.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. Large scale learning of general visual representations for transfer. *arXiv preprint arXiv:1912.11370*, 2019.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009a.

A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Computer Science Department, University of Toronto, 2009b.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

A. Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6402–6413, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent only converges to minimizers. In V. Feldman, A. Rakhlin, and O. Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 1246–1257. JMLR.org, 2016. URL http://proceedings.mlr.press/v49/lee16.html.

T. Liang and A. Rakhlin. Just interpolate: Kernel" ridgeless" regression can generalize. *arXiv preprint arXiv:1808.00387*, 2018.

H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=S1eYHoC5FX`.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 3730–3738. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.425. URL `https://doi.org/10.1109/ICCV.2015.425`.

P. M. Long and H. Sedghi. Generalization bounds for deep convolutional neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=r1e_FpNFDr`.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

O.-A. Maillard and R. Munos. Online learning in adversarial lipschitz environments. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 305–320. Springer, 2010.

W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

S. Mei and A. Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun): 983–999, 2006.

P. P. Mitra. Understanding overfitting peaks in generalization error: Analytical risk curves for l2 and l1 penalized interpolation. *ArXiv*, abs/1906.03667, 2019.

A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

V. Muthukumar, K. Vodrahalli, and A. Sahai. Harmless interpolation of noisy data in regression. *arXiv preprint arXiv:1903.09139*, 2019.

V. Muthukumar, K. Vodrahalli, V. Subramanian, and A. Sahai. Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, 2020.

V. Nagarajan and J. Z. Kolter. Uniform convergence may be unable to explain generalization in deep learning. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11611–11622, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/05e97c207235d63ceb1db43c60db7bbb-Abstract.html`.

P. Nakkiran. A discussion of'adversarial examples are not bugs, they are features': Adversarial examples are just bugs, too. *Distill*, 4(8):e00019–5, 2019a.

P. Nakkiran. More data can hurt for linear regression: Sample-wise double descent. *arXiv preprint arXiv:1912.07242*, 2019b.

P. Nakkiran. Adversarial robustness may be at odds with simplicity. *CoRR*, abs/1901.00532, 2019c. URL `http://arxiv.org/abs/1901.00532`.

P. Nakkiran. Learning rate annealing can provably help generalization, even for convex problems. *OPT Workshop on Optimization for Machine Learning*, 2020. URL `https://arxiv.org/abs/2005.07360`.

P. Nakkiran and Y. Bansal. Distributional generalization: A new kind of generalization. *ArXiv*, abs/2009.08092, 2020.

P. Nakkiran, G. Kaplun, D. Kalimeris, T. Yang, H. Zhang, B. L. Edelman, and B. Barak. SGD on neural networks learns functions of increasing complexity. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3491–3501, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/b432f34c5a997c8e7c806a895ecc5e25-Abstract.html`.

P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=B1g5sA4twr`.

P. Nakkiran, B. Neyshabur, and H. Sedghi. The deep bootstrap framework: Good online learners are good offline generalizers. In *International Conference on Learning Representations*, 2021a. URL `https://openreview.net/forum?id=guetrIHLFGI`.

P. Nakkiran, P. Venkat, S. M. Kakade, and T. Ma. Optimal regularization can mitigate double descent. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=7R7fAoUygoa.

H. Narayanan and S. K. Mitter. Sample complexity of testing the manifold hypothesis. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 1786–1794. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/hash/8a1e808b55fde9455cb3d8857ed88389-Abstract.html.

N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari. Learning with noisy labels. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 1196–1204, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/3871bd64012152bfb53fdf04b401193f-Abstract.html.

B. Neyshabur. Towards learning convolutions from scratch. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/5c528e25e1fdeaf9d8160dc24dbf4d60-Abstract.html.

B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *ICLR (Workshop)*, 2015a.

B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In P. Grünwald, E. Hazan, and S. Kale, editors, *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, volume 40 of *JMLR Workshop and Conference Proceedings*, pages 1376–1401. JMLR.org, 2015b. URL http://proceedings.mlr.press/v40/Neyshabur15.html.

B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018a. URL https://openreview.net/forum?id=Skz_WfbCZ.

B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018b.

M. A. Olson and A. J. Wyner. Making sense of random forest probabilities: a kernel perspective. *arXiv preprint arXiv:1812.05792*, 2018.

M. Opper. Statistical mechanics of learning: Generalization. *The Handbook of Brain Theory and Neural Networks, 922-925.*, 1995.

M. Opper. Learning to generalize. *Frontiers of Life, 3(part 2), pp.763-775.*, 2001.

M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://www.aclweb.org/anthology/N19-4009.

D. Page. How to train your resnet. https://myrtle.ai/how-to-train-your-resnet-4-architecture/, 2018a.

D. Page. How to train your resnet, 2018b.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

L. Pillaud-Vivien, A. Rudi, and F. R. Bach. Statistical optimality of stochastic gradient descent on hard learning problems through multiple passes. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8125–8135, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/10ff0b5e85e5b85cc3095d431d8c08b4-Abstract.html.

T. Pospisil and A. B. Lee. Rfcde: Random forests for conditional density estimation. *arXiv preprint arXiv:1804.05753*, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1177–1184. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html.

D. Rolnick, A. Veit, S. Belongie, and N. Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

J. S. Rosenfeld, A. Rosenfeld, Y. Belinkov, and N. Shavit. A constructive prediction of the generalization error across scales. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=ryenvpEKDr.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00474. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html.

R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

S. Shalev-Shwartz, O. Shamir, and S. Shammah. Failures of gradient-based deep learning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3067–3075. PMLR, 2017. URL http://proceedings.mlr.press/v70/shalev-shwartz17a.html.

S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

V. Shankar, A. Fang, W. Guo, S. Fridovich-Keil, J. Ragan-Kelley, L. Schmidt, and B. Recht. Neural kernels without tangents. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8614–8623. PMLR, 2020. URL http://proceedings.mlr.press/v119/shankar20a.html.

U. Sharma and J. Kaplan. A neural scaling law from the dimension of the data manifold. *arXiv preprint arXiv:2004.10802*, 2020.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

D. Soudry, E. Hoffer, M. S. Nacson, and N. Srebro. The implicit bias of gradient descent on separable data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r1q7n9gAb.

C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 843–852. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.97. URL https://doi.org/10.1109/ICCV.2017.97.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298594. URL https://doi.org/10.1109/CVPR.2015.7298594.

S. Thulasidasan, T. Bhattacharya, J. A. Bilmes, G. Chennupati, and J. Mohd-Yusof. Combating label noise in deep learning using abstention. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6234–6243. PMLR, 2019. URL http://proceedings.mlr.press/v97/thulasidasan19a.html.

J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inf. Theory*, 53(12):4655–4666, 2007. doi: 10.1109/TIT.2007.909108. URL https://doi.org/10.1109/TIT.2007.909108.

A. B. Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008.

A. W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, XVI(2):264–280, 1971.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017a. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017b. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272, 2020. doi: https://doi.org/10.1038/s41592-019-0686-2.

L. Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

C. Wei and T. Ma. Improved sample complexities for deep neural networks and robust classification via an all-layer margin. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJe_yR4Fwr.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910, 2019.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5987–5995. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.634. URL https://doi.org/10.1109/CVPR.2017.634.

L. Yang, L. Deng, M. H. Hajiesmaili, C. Tan, and W. S. Wong. An optimal algorithm for online non-convex learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(2):1–25, 2018.

S. Zagoruyko and N. Komodakis. Wide residual networks. In R. C. Wilson, E. R. Hancock, and W. A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017a. URL https://openreview.net/forum?id=Sy8gdB9xx.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017b. URL https://openreview.net/forum?id=Sy8gdB9xx.

L. Ziyin, B. Chen, R. Wang, P. P. Liang, R. Salakhutdinov, L.-P. Morency, and M. Ueda. Learning not to learn in the presence of noisy labels. *arXiv preprint arXiv:2002.06541*, 2020.