Université de Montréal

Some Phenomenological Investigations in Deep Learning

par

Aristide Baratin

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Décembre, 2021

Université de Montréal
Faculté des arts et des sciences

Cette thèse intitulée:

Some Phenomenological Investigations in Deep Learning

présentée par:

Aristide Baratin

a été évaluée par un jury composé des personnes suivantes:

Ioannis Mitliagkas,	président-rapporteur
Simon Lacoste-Julien,	directeur de recherche
Aaron Courville,	membre du jury
Joan Bruna,	examinateur externe

Thèse acceptée le: .....

Là, tout n'est qu'ordre et beauté,
Luxe, calme et volupté.

Charles Baudelaire, « L'Invitation au voyage ».

Don't panic.

Douglas Adams, *The Hitchhiker's Guide to the Galaxy.*

Résumé

Les remarquables performances des réseaux de neurones profonds dans de nombreux domaines de l'apprentissage automatique au cours de la dernière décennie soulèvent un certain nombre de questions théoriques. Par exemple, quels mecanismes permettent à ces reseaux, qui ont largement la capacité de mémoriser entièrement les exemples d'entrainement, de généraliser correctement à de nouvelles données, même en l'absence de régularisation explicite ? De telles questions ont fait l'objet d'intenses efforts de recherche ces dernières années, combinant analyses de systèmes simplifiés et études empiriques de propriétés qui semblent être corrélées à la performance de généralisation. Les deux premiers articles présentés dans cette thèse contribuent à cette ligne de recherche. Leur but est de mettre en évidence et d'etudier des mécanismes de biais implicites permettant à de larges modèles de prioriser l'apprentissage de fonctions "simples" et d'adapter leur capacité à la complexité du problème.

Le troisième article aborde le problème de l'estimation de information mutuelle en haute, en mettant à profit l'expressivité et la scalabilité des reseaux de neurones profonds. Il introduit et étudie une nouvelle classe d'estimateurs, dont il présente plusieurs applications en apprentissage non supervisé, notamment à l'amélioration des modèles neuronaux génératifs.

Mots-Clés: Apprentissage statistique, réseaux de neurones profonds, théorie de l'apprentissage, information mutuelle, modèles génératifs.

Abstract

The striking empirical success of deep neural networks in machine learning raises a number of theoretical puzzles. For example, why can they generalize to unseen data despite their capacity to fully memorize the training examples? Such puzzles have been the subject of intense research efforts in the past few years, which combine rigorous analysis of simplified systems with empirical studies of phenomenological properties shown to correlate with generalization. The first two articles presented in these thesis contribute to this line of work. They highlight and discuss mechanisms that allow large models to prioritize learning 'simple' functions during training and to adapt their capacity to the complexity of the problem.

The third article of this thesis addresses the long standing problem of estimating mutual information in high dimension, by leveraging the scalability of neural networks. It introduces and studies a new class of estimators and present several applications in unsupervised learning, especially on enhancing generative models.

Keywords: machine learning, deep neural networks, statistical learning theory, mutual information, generative models.

Contents

List of tables

List of figures

Remerciements

Je tiens d'abord à exprimer ma gratitude à mon directeur de recherche, Simon Lacoste-Julien, pour sa disponibilité, son efficacité, et l'attention constante dont il a fait preuve à mon égard tout au long de mon parcours au Mila. Mon travail doit beaucoup aux multiples échanges et discussions avec mes collaborateurs et amis, parmi lesquels Ishmaël Belghazi, Akram Erraqabi, Thomas George, Devon Hjelm, Guillaume Lajoie, César Laurent, Nasim Rahaman and Alessandro Sordoni, ainsi qu'avec mes nombreux compagnons d'infortune du Mila.

Je remercie les équipes de cet extraordinaire laboratoire qu'est le Mila pour leur efficacité et leur devouement. Je remercie également chaleureusement les membres de mon jury de thèse, Ioannis Mitliagkas et Aaron Courville – et tout particulièrement Joan Bruna, pour le temps qu'il aura consacré à l'examen de cette thèse.

Ma famille, mes amis – qu'ils soient parisiens, ontariens, berlinois ou montréalais – savent combien je leur dois. Je remercie tout particulierement ma compagne et super-partenaire Sarah, dont la présence chaleureuse et confiante m'a si bien accompagné dans cette aventure.

Chapter 1

Introduction

1. Motivating Challenges

Deep learning methods (Goodfellow et al., 2016a) have become ubiquitous in data science and machine learning in the last decade. They played a pivotal role in recent breakthroughs in various complex learning problems such as image classification (Krizhevsky et al., 2012), language translation (Sutskever et al., 2014), playing Go (Silver et al., 2017) and Starcraft (Vinyals et al., 2019), or protein structure prediction (Senior et al., 2020).

The enormous practical success of these methods, however, contrasts with our limited theoretical understanding of their performance (e.g., Goldblum et al., 2020). This large gap between theory and practice brings immediate challenges for these models treated as black-box, such as the lack of reliable guarantees of accuracy, robustness, and fairness.[1] Moreover one can hope that a better theoretical grasp of deep learning will (*i*) yield mathematically grounded principles for model and algorithm selection (*ii*) trigger progress towards overcoming notorious limitations of current models, such as their brittleness under adversarial perturbations (Xu et al., 2020) or distribution shift (Taori et al., 2020).

There are two related questions that theory should address:

(*i*) *Understanding generalization: when does deep learning work and why?*

Using large and flexible models has a long history in machine learning, but it usually comes along with some form of capacity control to avoid overfitting, such as a regularizer term added to the loss that penalizes complex hypotheses. Somewhat surprisingly (Geman et al., 1992), high-capacity neural networks can often perform well on real data even *without* explicit regularization (Neyshabur et al., 2015; Zhang et al., 2017a) or early stopping (Hoffer et al., 2017), and even while (over)fitting some added noise in the training data. Empirical observations like double descent curves (Advani & Saxe, 2017; Belkin et al., 2019a; Geiger et al., 2019; Nakkiran et al., 2020) even suggest that generalization performance improves with

[1] See the NeuriPS 2019 Workshop on 'Machine Learning with Guarantees' for an overview of these issues.

capacity (e.g, model size).[2] While choices related to the architecture and the optimization procedure introduce biases that can effectively reduce capacity, such biases do not affect the ability of these models to memorize even random training data (Zhang et al., 2017a). Explaining such behavior is a challenge for statistical learning theory, especially for complexity-based and uniform convergence approaches to generalization (Nagarajan & Kolter, 2019a).

(*ii*) *Understanding failures modes: when does deep learning fail and why?*

Deep learning models have also shown important weaknesses such as extreme sensitivity to distribution shifts. For example, in image classification, translating an image by a few pixels (Azulay & Weiss, 2019) or modifying the background (Beery et al., 2018) can drastically change the model's output. This suggests that predictions tend to be based on superficial heuristics and correlations rather than robustly informative features (Arjovsky et al., 2019; Geirhos et al., 2020). Similarly, a high average test accuracy can hide disproportionately high errors on atypical or under-represented groups of the data (Buolamwini & Gebru, 2018); this can occur when the model exploits spurious input-output correlations that hold for the majority of the training examples – and uses its excess capacity to memorize the other examples (Sagawa et al., 2020a,b). This is part of a more general struggle of these systems to learn long-tailed distributions, which are ubiquitous in the natural world (Newman, 2005).

A major difficulty to address both questions theoretically is due to *underspecification* (D'Amour et al., 2020). In large function classes such as neural networks, there are many functions that fit perfectly the training data, yet behave very differently on new samples from the same distribution; there are even many functions that have a similar strong performance on new samples drawn from the training distribution, but behave very differently when tested outside the training domain. The challenge is to understand what type of solutions is favoured among all possible ones, depending both on the various inductive biases encoded in the choice of architecture and the training algorithm and on the specific properties of natural data.

The first two contributions presented in this thesis were motivated by the question (*i*) above, and specifically by the problem of characterizing biases introduced by the training dynamics and acting as implicit regularizers (Neyshabur et al., 2015). Given the mathematical challenges posed by the highly non-linear training dynamics of deep networks, research efforts in this direction have focused either on (*i*) theoretical analysis of simpler systems or idealized limits of deep learning; (*ii*) empirical studies of phenomenological properties of deep learning and their correlation with generalization. The works presented in Chapter 3 and 4 belong to

[2]As it was recognized already in the 90s (e.g., Seung et al., 1992; Opper, 1995; Bös, 1998), some of these observations (such as double descent) are not specific to deep learning but correspond to universal characteristics of high dimensional statistical inference. These early results, derived in teacher-student linear settings using heuristics from statistical physics such as the replica method, have been put on rigorous grounds and considerably extended in a recent line of works (e.g., Hastie et al., 2019; Mei & Montanari, 2019; Loureiro et al., 2021). See Bartlett et al. (2021a) for a review of these.

the latter category. One of our goals was to formulate, formalize and test the hypothesis that neural networks training is biased towards learning *simple* functions, i.e., typically, functions supported on a small number of highly predictive features. Question (*ii*) is not directly investigated in this thesis, but some of our results naturally suggest the hypothesis that such a simplicity bias, while it may drive the generalization ability of deep learning models in a wide range of supervised tasks, also underpins their failure to generalize out-of-distribution.

One specificity of deep learning is to produce efficient representations of the data. Whereas learning in high dimension is a notoriously cursed problem (Donoho, 2000), an intuitive idea is that the relevant information for most tasks of interest is encoded in some low-dimensional structure in data space. Most modern deep learning architectures can be thought of as encoding some prior knowledge on such structure, e.g., by exploiting symmetries of the input data (Bronstein et al., 2021). In the context of supervised learning, our experiments in Chapter 4 illustrate how the internal layer representations improve their alignment to such a low-dimensional structure during training. In the more ambitious context of unsupervised representation learning, the recent self-supervised methods (Misra & Maaten, 2020; Chen et al., 2020; He et al., 2020; Caron et al., 2020) aim at learning features invariant under some irrelevant information, such as the one added by data augmentation.

This leads to a third challenge for a theory of deep learning:

(*iii*) *Understanding the structure of natural data: how to characterize good representations?*

Although this question is not investigated in this thesis, Chapter 5 indirectly contributes to it by designing a method to efficiently compute information measures for high dimensional random variables. Specifically, we design new estimators of mutual information (Shannon, 1948) that leverage the scalability of neural networks. Mutual information plays a central role in approaches to representation learning such as information bottleneck (Tishby et al., 2000) or information maximization (Linsker, 1988; Hjelm et al., 2019). We also use it as an attempt to enhance deep generative models.

2. Summary of Contributions

The first two contributions presented in this report are driven by two mains ideas:

(1) to investigate implicit regularization mechanisms directly in function space (as opposed to parameter space);

(2) to go beyond the usual analysis at stationary points (e.g., via parameter, Jacobian or Hessian norms) and study the whole trajectories of optimization.

Some of our main insights can be summarized as follows:

Insight 1: We formulate the hypothesis that neural networks prioritize learning 'simple' functions during training (Chapter 3). We examine this hypothesis through the lens of Fourier

analysis of the network function, i.e. using Fourier frequency as measure of complexity. We find empirical evidence of a 'spectral bias' towards low frequency functions.

Next, we seek a notion of complexity that is more intrinsic to the network function and the data distribution (as opposed to the somewhat ad-hoc Fourier basis). We note that the dynamics of gradient descent in (kernelized) linear regression provides a mathematically precise instantiation of the 'spectral bias', using the kernel eigenbasis: the leading spectral components are learned faster, see Lemma B.5 for a proof of this (well-known) fact. Now, while the network training dynamics is not linear, it can be locally linearized and expressed in terms of neural tangent kernels (Jacot et al., 2018), whose eigendecomposition yields a natural (though local) notion of function complexity. Neural tangent kernels have attracted a lot of interest recently, due to a specific training regime where neural networks can provably be well approximated by their linearization around initialization (Jacot et al., 2018; Du et al., 2019b,a; Allen-Zhu et al., 2019; Chizat et al., 2019) during optimization. In such a regime, deep learning inherits all (optimization and generalization) properties of kernel learning.

Insight 2: We propose to analyze the spectral bias of deep networks through the lens of neural tangent kernels (Chapter 4, section 2).

Fig. B.2 in Appendix B.1.5 makes the link with the synthetic experiments of Chapter 3: for a randomly initialized MLP on 1D uniform data, the Fourier decomposition of the tangent kernel's eigenfunctions show a remarkable alignment to sinusoids of increasing frequencies. During the course of the project, several works (e.g., Yang & Salman, 2019) came out which explored this idea, by providing detailed spectral analysis of the neural tangent kernel in the linear regime, for tractable data distributions allowing for explicit computations. As a result, we refocused our work on the impact of feature learning: our main goal in Chapter 4 is to study how the (time dependent) tangent kernel and its spectrum adapt to the task during training for standard training setups.

Insight 3: We observe a dynamical alignment of the neural tangent features along a small number of task-relevant directions during training (Chapter 4, sections 2 and 3). In the language of the spectral bias, this evolution makes these 'task-relevant' features (which depend on the labels via the loss gradients) 'easy-to-fit' (w.r.t to the learned tangent kernel). We interpret this as a combined mechanism of feature selection and geometrical compression, and argue that it helps the network adapt to the intrinsic complexity of the problem.

The next goal is to leverage these insights to tentatively define a quantitative notion of effective capacity for neural networks trained by gradient descent. This part of the work follows a line of research on complexity measures (see, Jiang et al., 2020, and references therein), some of which are theoretically motivated by generalization bounds (e.g., Neyshabur et al., 2019; Bartlett et al., 2017); others (e.g., Keskar et al., 2016; Fort et al., 2019; Chatterjee, 2020) are justified by experimentation and observation. The main purpose of such measures

is to identify factors inherent to the model, the algorithm and properties of the data, which would explain, or at least correlate with, generalization.

Insight 4: By extrapolating an analysis of Rademacher complexity bounds for linear models, we design a complexity measure for deep networks in terms of sequences of tangent kernel classes along optimization paths. We run basic ablation experiments which suggest a correlation with generalization (Chapter 4, section 4). We illustrate in a toy setting how this measure suggests a modification of the learning algorithm where the geometry in which to perform gradient steps is optimized at each iteration.

Finally, Chapter 5 contributes to the deep learning toolbox by proposing and studying new estimators of mutual information, which leverage the scalability of neural networks. We introduce MINE (Mutual Information Neural Estimation) and provide some theoretical analysis. We present a handful of applications on which MINE can be used to optimize mutual information. We apply MINE to improve adversarially trained generative models (Goodfellow et al., 2014a) ; we also use it to implement Information Bottleneck (Tishby et al., 2000), applying it to supervised classification.

3. Excluded Research

Part of my research contributions and publications during my Ph.D does not appear in this manuscript. These include work on:

- adversarial robustness through domain adaptation techniques (Erraqabi et al., 2017)
- revisiting the bias-variance tradeoff for neural networks (Neal et al., 2018)
- studying regularity properties of self-attention (Vuckovic et al., 2020)

Chapter 2

Background

This chapter introduces some context, background and notations for the work presented in the rest of the thesis.

1. Learning with Neural Networks

1.1. Supervised learning

The goal of supervised learning[1] is to infer-output mappings from examples. A typical task is to predict an output $y \in \mathcal{Y}$ from an input $x \in \mathcal{X}$, where the pairs (x, y) are drawn from some unknown joint distribution $p(x, y)$. The learner is given a finite set of training points $(x_1, y_1), \cdots, (x_n, y_n)$, generally assumed to be independently and identically distributed (i.i.d) according to ρ. The goal of the learner is to devise a map $f : \mathcal{X} \to \mathcal{Y}$ with low expected error,

$$L(f) = \mathbb{E}_{(x,y)\sim p} \left[\ell(f(x), y)\right] \tag{1}$$

for some bounded loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. In classification problems, for instance, $\mathcal{Y}$ is a finite set of class labels. A standard choice of loss function is the 0-1 loss, $\ell(y, y') = \mathbf{1}_{y \neq y'}$. In this case, the expected loss gives the probability $\Pr\{f(x) \neq y\}$ to make a wrong prediction.

Since the distribution ρ is unknown, in practice, the *empirical loss*

$$\hat{L}(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \tag{2}$$

is used as a proxy for the expected loss. However having a small empirical loss does not guarantee a small expected loss (1). This is the problem of *generalization*: two learners that achieve a similar performance on the training data might perform very differently on unseen data. The difference $L(f) - \hat{L}(f)$ is often called *generalization gap*. Note that, for any predictor f chosen independently of the training set, $\hat{L}(f)$ is an average of independent random

[1] Standard textbook references include Bousquet et al. (2003); Mohri et al. (2012); Shalev-Shwartz & Ben-David (2014).

variables and $L(f)$ is the average of the training loss over the training data, $L(f) = \mathbb{E}[\hat{L}(f)]$; so the generalization gap can be controlled by standard concentration inequalities such as Hoeffding's inequality (e.g., Boucheron et al., 2013, §2.6). The story is more complicated for training set-dependent predictors, such as minimizers of (2) or any predictor picked by the learning algorithm. A standard strategy to obtain generalization guarantees in this case is to extend concentration inequalities to *uniform* bounds over some class $\mathcal{F}$ of predictors considered by the learner. The tightness of these bounds depends on specific properties of the model class. We discuss this further in the next section.

Neural networks. Supervised learning problems are commonly addressed by choosing a *parametric model*, i.e a class of predictors $f_{\mathbf{w}}$ labelled by some $\mathbf{w} \in \mathcal{W}$, where $\mathcal{W} \subset \mathbb{R}^P$ is a space of parameters. For example, one can choose $f_{\mathbf{w}}(x) = \langle \mathbf{w}, \Phi(x) \rangle$ to depend linearly on finite-dimensional feature vectors $\Phi(x) \in \mathbb{R}^P$, where $\Phi \colon \mathcal{X} \to \mathbb{R}^P$ is a fixed representation map that can be designed to encode some prior knowledge of the data.

Neural networks are examples of *non-linear* parametric models. The most basic architecture, a.k.a the multilayer perceptron (MLP) or fully connected network, characterized by L representation layers with given widths $d_1, \cdots d_L$, defines functions that take inputs $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{d_0}$ and apply successive transformations $T^{(k)} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ to produce an output,

$$f(\mathbf{x}) = T^{(L+1)}\left(T^{(L)}(\cdots T^{(1)}(\mathbf{x}))\right). \tag{3}$$

The basic transformations $T^{(k)}$ usually take the form $T^{(k)}(\boldsymbol{u}) = \sigma(W^{(k)}\boldsymbol{u} + \mathbf{b}^{(k)})$ for some learnable weight matrix $W^{(k)}$ and learnable vector $\mathbf{b}^{(k)}$, where σ is a fixed non-linear activation function acting component-wise on vectors. A common choice of activation function is the Rectified Linear Unit (ReLU) (Jarrett et al., 2009; Glorot et al., 2011), $\sigma(u) = \max(0, u)$. The idea with the compositional structure of (3) is to produce, through the learning process, a hierarchy of representations of the input data that are relevant for the task.

Most modern architectures result from this idea of stacking together many layers of neural components with trainable parameters. These components are often designed to produce representations that capture specific structures and symmetries of the input data (Bronstein et al., 2021). For example, many architectures used for computer vision applications include convolutional layers, which capture the action of local spatial translations on the input image. The transformer architecture (Vaswani et al., 2017) processes sequential inputs or set-structured data, in a way that model correlations between inputs elements and can capture permutation invariance (Lee et al., 2018).

Neural networks provide flexible model classes with high approximation power. Classical results (Hornik et al., 1989; Cybenko, 1989; Leshno et al., 1993) show that two-layer networks with a variety of activation functions can approximate any continuous function with arbitrarily high precision. More recent work analyzed the role of depth (number of layers) for efficient

approximation (Lu et al., 2017; Poole et al., 2016; Telgarsky, 2016; Eldan & Shamir, 2016; Safran & Shamir, 2017; Yarotsky, 2017). This high flexibility, combined with the strong inductive biases induced by the architectural choices, make these systems powerful candidate models to capture the complexities and regularities of real-world prediction tasks.

Gradient-based training. Given a parametric model, we may consider the problem

$$\arg\min_{\mathbf{w}\in\mathcal{W}} \hat{L}(f_{\mathbf{w}}) := \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\mathbf{w}}(x_i), y_i). \tag{4}$$

Training neural networks to minimize the empirical error is, in principle, a hard problem. In fact it is generally NP-hard, even with very small networks (Judd, 1988; Blum & Rivest, 1989), even in the realizable case where we know the labels can be predicted correctly by such a small network (Goel et al., 2021), and even with convex losses (Vu, 1998; Bartlett & Ben-David, 1999). Heuristics used in deep learning are all based on variants of gradient descent, for some surrogate differentiable loss function. Gradient descent iteratively updates the model parameters in the negative direction of the loss gradient,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \hat{L}(f_{\mathbf{w}_t}), \tag{5}$$

where $\eta_t > 0$ is an adjustable hyperparameter, called learning rate, which controls the size of the gradient step. Most modern approaches use stochastic versions of gradient descent, where the gradients are estimated using mini-batch samples; they also add momentum, adaptive learning rates, or combinations of the two (Kingma & Ba, 2015).[2] Gradients across layers, as defined by the chain rule, can be efficiently calculated by backpropagation (Rumelhart et al., 1986), using distributed implementations (Dean et al., 2012) based on GPUs (Chellapilla et al., 2006), even for networks with hundreds of billions (Brown et al., 2020) and even trillions (Fedus et al., 2021; Lin et al., 2021) of parameters.

1.2. Generative modeling

Generative modeling aims at producing samples from a distribution that approximates the true distribution of the data. To make the learning problem tractable in high-dimension, the distribution is often modeled using low dimensional latent variables. In this context, the generative process can be thought of as inverting an inference process. For example, variational autoencoders (Kingma & Welling, 2014; Rezende et al., 2014) model both processes stochastically, through probabilistic encoder and a decoder networks $q_\phi(z|x), p_\theta(x|z)$ jointly trained to maximize a lower bound of the log likelihood (Blei et al., 2017).

The generative adversarial framework proposed by Goodfellow et al. (2014a) considers distributions $p_\theta(x)$ induced by deterministic generator functions $g_\theta \colon \mathcal{Z} \to \mathcal{X}$ between latent

[2] See e.g., Ruder (2016) for a didactic overview of gradient-based optimizers most commonly used in deep learning.

and observed variables, as the push-forward of some simple (e.g., gaussian) distribution $q(z)$:

$$x \sim p_\theta \qquad \Longleftrightarrow \qquad x = g_\theta(z), \quad z \sim q. \tag{6}$$

The generator is trained jointly with a second model $d_\varphi : \mathcal{X} \to [0, 1]$ called discriminator, so as to optimize the following min max objective

$$\min_\theta \max_\varphi V(d_\varphi, p_\theta) := \mathbb{E}_{x\sim p} \left[d_\varphi(x)\right] + \mathbb{E}_{x\sim p_\theta} \left[\log(1 - d_\varphi(x))\right] \tag{7}$$

where p denotes the true data distribution. In words, the discriminator is trained to distinguish true versus modeled samples, whereas the goal of the generator is to fool the discriminator.

Empirically, generative adversarial networks (GANs) have been shown to perform remarkably well, reaching unprecedented performance on high resolution image synthesis (e.g., Karras et al., 2018; Brock et al., 2019). However they are not exempt of weaknesses. For example, mode collapse is a common pathology where the generator fails to produces samples with sufficient diversity (i.e., they poorly represent some modes). We will investigate in Chapter 5 an approach that regularizes the generator's loss with the neg-entropy of the samples, so as to encourage their diversity. As the sample entropy is intractable, we propose to use an estimator of mutual information as a proxy.

2. Complexity and Generalization

Understanding generalization with neural networks is a long-standing problem (e.g., Watkin et al., 1993). In light of the empirical success of deep learning, this problem has been revived in the last few years, stimulated in part by thought-provoking experiments (Neyshabur et al., 2015; Zhang et al., 2017a; Nakkiran et al., 2020) with large networks trained on data with various levels of noise.

A general idea in classical statistical theory is that the learning ability of a model results from a balance between *goodness-of-fit* and *complexity*. While a model should be flexible enough to capture regularities in the data, overly complex models tend to be too sensitive to random fluctuations in the training data and perform poorly on unseen data. This, for example, yields Occam's razor as a guiding principle for model selection and algorithm design: among models that fit well the training data, it favours the simplest ones. In practice, many methods using large models also involve explicit regularization schemes, which effectively reduce their complexity and prevent overfitting.

In some respects, the empirical performance of deep learning seems at odds with this idea. Neural networks are complex models with high expressive power; they often have vastly more free parameters than training points and can be trained to fit perfectly even unstructured random data (Zhang et al., 2017a). Yet they often show high generalization performance on natural data even without explicit regularization (Neyshabur et al., 2015; Hoffer et al., 2017).

On the other hand, the type of solutions and their generalization ability depend implicitly on various aspects of the training procedure, such as the choice of architecture and optimizer. The challenge is to understand the bias introduced by such choices and their role as implicit regularizers (Neyshabur et al., 2015).

Note that the notion of 'complexity' is equivocal. They are many ways to measure complexity, for instance by the number of free parameters, an upper bound on some parameter norm, or the description length of the model in some programming language. Because of the no-free-lunch theorem, we should not expect *a priori* that one is better (in the sense that it drives generalization more) than the other. The notion of complexity that best describes real-data regularities ultimately depends on the problem of interest.

2.1. Generalization bounds

Here we give a few examples of classical generalization guarantees that can be obtained in the statistical learning framework, based on uniform convergence arguments. They are all different ways to show that with enough training samples, models with bounded complexity can be guaranteed to give probably approximately correct (PAC) (Valiant, 1984) predictions on unseen data.

Consider the setting of Section 1.1. We assume for simplicity that the loss function takes values in $[0, 1]$. The basic tool is Hoeffding's inequality: given any predictor f in the model class $\mathcal{F}$, it gives bounds on the generalization gap with high probability on the choice of training samples:

$$\Pr\{L(f) - \hat{L}(f) > \epsilon\} \leq e^{-2n\epsilon^2} \tag{8}$$

Equivalently (by setting $\delta := e^{-2n\epsilon^2}$ above), this says that for arbitrarily small $\delta > 0$, with probability at least $1 - \delta$,

$$L(f) \leq \hat{L}(f) + \sqrt{\frac{\log(1/\delta)}{2n}} \tag{9}$$

The probability is with respect to the sampling of the training data, so this does not apply to data-dependent functions such as the one picked by the learning algorithm. To extend this result to these, we need bounds that hold *uniformly* over some class of functions considered a priori by the learner.

Uniform bounds. In the case of a finite class $\mathcal{F}$, uniform bounds can be easily obtained by applying union bound (Boole's inequality) over all functions in $\mathcal{F}$. This brings a cardinality factor $|\mathcal{F}|$ on the right hand side of (8). We obtain the following bound that holds for all $f \in \mathcal{F}$, with probability at least $1 - \delta$ over the sampling of the training data,

$$L(f) \leq \hat{L}(f) + \sqrt{\frac{\log|\mathcal{F}| + \log(1/\delta)}{2n}} \tag{10}$$

where the term $\ln|\mathcal{F}|$ can be interpreted as a measure of complexity for the model class.

Occam bounds give a new interpretation of (10) – and allow for an extension to countably infinite model classes. Consider a nonnegative function π on $\mathcal{F}$ such that $\sum_{f\in\mathcal{F}} \pi(f) \leq 1$, e.g., some probability distribution on $\mathcal{F}$. For $\delta > 0$, the trick is to apply union bound to (8) with an f-dependent ϵ_f such that $\delta = \pi(f)e^{-2n\epsilon_f^2}$. We obtain the following uniform bound with probability $1-\delta$,

$$L(f) \leq \hat{L}(f) + \sqrt{\frac{\log(1/\pi(f)) + \log(1/\delta)}{2n}} \tag{11}$$

where the description length (Pu, 2006, Chap 2) $\log(1/\pi(f))$ shows up as a measure of complexity for the predictor f.

Vapnik-Chervonenkis (VC) theory allows for further extensions of the bound (10) to uncountable model classes $\mathcal{F}$, in terms of generalized notions of cardinality such as the VC-dimension[3] (Vapnik, 1995). VC dimension yields a purely combinatorial notion of complexity: it is defined as the size of the largest subset $S \subset \mathcal{X}$ such that any labelling of S can be predicted correctly by some predictor in $\mathcal{F}$. Classical results (Maass, 1994; Bartlett et al., 1999, 2019a) however show that the VC-dimension of neural networks scales with the number of parameters. This typically leads to spurious generalization bounds for the large networks used in practice, whose number of parameters exceeds by far the number of training samples. It also fails to explain several empirical learning phenomena such as the double descent of the test error as the network size increases (Neyshabur et al., 2015; Neal et al., 2018; Belkin et al., 2019a; Nakkiran et al., 2020). Even if we could restrict the analysis to some identified subclass of 'typical' networks obtained from the standard training procedures, the fact that such a subclass is able to fit all possible labels (Zhang et al., 2017a) suggests that VC-theory is not sufficient to explain generalization for such systems.

Margin-based bounds. Methods using *scale-sensitive* versions of the VC-dimension (Bartlett, 1996), such as the fat-shattering dimension or Rademacher complexity (Bartlett & Mendelson, 2002) can in principle overcome these limitations. These methods exploit the fact that, in many practical settings, prediction problems are addressed by considering classes of real or vector-valued *score* functions. Score functions and score-based losses depend not only on prediction accuracies, but also on prediction margins. Since margins depend on the magnitude of the function output, such methods can bypass the cardinality-based analysis of VC theory by controlling complexity in terms of the size of the parameters (Bartlett, 1996).

Let us give a simple instance of margin bounds based on Rademacher complexity, which we use in Chapter 4. For simplicity we consider a binary classification task $\mathcal{Y} = \{\pm 1\}$ and a class $\mathcal{F}$ of real-valued score functions. Given any scale parameter $\gamma > 0$, we define the expected margin loss as

$$L_\gamma(f) = \mathbb{E}_{(x,y)\sim\rho}\left[\mathbf{1}_{yf(x)\leq\gamma}\right] \tag{12}$$

[3]and its extensions to non-binary classes (Natarajan, 1989; Daniely et al., 2011).

and denote by $\widehat{L}_\gamma(f)$ its empirical estimate. Note that setting $\gamma = 0$ in this formula reproduces the expected classification error, $L_0(f)$. The following bound on the expected classification error holds (Mohri et al., 2012, Theorems 4.4) for all $f \in \mathcal{F}$ with probability $1 - \delta$:

$$L_0(f) \leq \widehat{L}_\gamma(f) + \frac{2}{\gamma}\mathcal{R}(\mathcal{F}) + 3\sqrt{\frac{\log(2/\delta)}{2n}} \tag{13}$$

where $\mathcal{R}(\mathcal{F})$ is the empirical Rademacher complexity defined as,

$$\mathcal{R}(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\sigma}\in\{\pm 1\}^n}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(x_i)\right]. \tag{14}$$

Rademacher complexity measures how well on average $\mathcal{F}$ correlates with a random labelling of the inputs. Unlike VC-dimension, it is a data-dependent measure of complexity, which can yield much tighter bounds on some problems. Such a framework yields generalization bounds for neural networks subject to norm constraints on the parameters (such as an upper bound on the spectral norm of the weights) and independent of the network size (Bartlett & Mendelson, 2002; Bartlett et al., 2017; Golowich et al., 2017; Neyshabur et al., 2018).

PAC-Bayes bounds. The PAC-Bayes approach (McAllester, 1999) yields bounds similar to (11) but with a twist: it bounds the expected performance of randomized predictors,

$$\mathbb{E}_{f\sim Q}[L(f)] \leq \mathbb{E}_{f\sim Q}[\hat{L}(f)] + \sqrt{\frac{\mathrm{KL}(Q||Q_0) + \log(1/\delta)}{2n}}, \tag{15}$$

uniformly for all distributions Q on $\mathcal{F}$, given some fixed prior distribution Q_0. Here the complexity of a random predictor is quantified by its distance (as measured by KL divergence) to the fixed prior. Several works including Langford & Caruana (2002); Nagarajan & Kolter (2019b) proposed a PAC-Bayes analysis of neural networks. The notion of stability under parameter perturbation around solutions can also be formalized in a natural way in this framework, which leads to useful insights for generalization (Dziugaite & Roy, 2017; Neyshabur et al., 2017a, 2018).

2.2. Empirical complexity measures

Despite such progress, most of these bounds are numerically vacuous (e.g., a bound ≥ 1 for the classification error) when evaluated empirically on realistic dataset sizes. Although non-vacuous bounds have been obtained in recent analysis, these hold only on modified networks obtained after a suitable compression (Arora et al., 2018b) or optimization (Dziugaite & Roy, 2017) procedure. Worse, the analysis of Nagarajan & Kolter (2019a) illustrates with an example a scenario where *any* uniform bound over a class that is likely to contain the algorithm's possible outputs is provably nearly vacuous.

Even without providing tight generalization guarantees, these approaches can nevertheless give useful insights for generalization. They theoretically motivate distribution-dependent measures of complexity that may be expected to correlate well with generalization for natural data. Other complexity measures have been proposed in the recent literature, such as sharpness (Keskar et al., 2016) or gradient coherence (Fort et al., 2019; Chatterjee, 2020), which do not arise from bounds but are justified by experimentation and observation. One such empirical measure will be motivated and studied in Chapter 4. The goal with these measures is to identify factors inherent to the model, the algorithm and properties of the data, which are correlated, and ideally causally related to, generalization. Large scale correlation studies for a large family of existing measures have been recently undertaken by Jiang et al. (2020); Dziugaite et al. (2020).

2.3. Lessons from linear models

Motivated by the empirical success of deep learning, there has recently been a wealth of work on linear or random feature models. Many of the puzzling properties observed empirically for large neural networks and kernel methods (Belkin et al., 2018), such as benign overfitting – i.e. good generalization performance despite a near-prefect fit to noisy training data –, the benefit of overparametrization, and the potentially hurtful role of regularization, have been rigorously analyzed in various linear setups (e.g., Bartlett et al., 2020; Muthukumar et al., 2019, 2020; Ghorbani et al., 2020; Bartlett et al., 2021b). Rigorous derivations of double descent curves for the test error have been obtained in this context (Hastie et al., 2019; Mei & Montanari, 2019; Belkin et al., 2019b).

While such theoretical analysis do not directly extend to deep learning, they can be appreciated in light of a recent line of work on a certain linear regime of deep learning. This is a training regime where the network can be well approximated by its first-order Taylor expansion around initialization, which yields a linear dynamics characterized by a specific, architecture-dependent kernel (Jacot et al., 2018; Du et al., 2019b,a; Allen-Zhu et al., 2019; Yang, 2020). We discuss this below.

3. Linear Regime

3.1. Linearized networks

Let $f_{\mathbf{w}}(x)$ represent a parametrized scalar function with parameter $\mathbf{w} \in \mathbb{R}^P$. For a neural network, the parameter can be thought of as a vector combining the entries of all trainable weights. Given any $\mathbf{w}_0 \in \mathbb{R}^P$, one can naively expand $f_{\mathbf{w}}(x)$ in $\mathbf{w}$ around $\mathbf{w}_0$,

$$f_{\mathbf{w}}(x) = f_{\mathbf{w}_0}(x) + \langle \mathbf{w} - \mathbf{w}_0, \Phi_{\mathbf{w}_0}(x) \rangle + O(\|\mathbf{w} - \mathbf{w}_0\|^2) \tag{16}$$

for any input x, where $\Phi_{\mathbf{w}}(x) := \nabla_{\mathbf{w}} f_{\mathbf{w}}(x) \in \mathbb{R}^P$ is the output gradient with respect to the parameter and $\langle , \rangle$ denotes the scalar product. The right-hand-side is a linear (affine) model

$$\bar{f}_{\mathbf{w}} = f_{\mathbf{w}_0} + \langle \mathbf{w} - \mathbf{w}_0, \Phi_{\mathbf{w}_0} \rangle \tag{17}$$

over some feature space of vectors $\Phi_{\mathbf{w}_0}(x)$. This is the linearization of the model at $\mathbf{w}_0$. The feature map $\Phi_{\mathbf{w}}$ defines the *tangent kernel* (Jacot et al., 2018),

$$k_{\mathbf{w}}(x, \tilde{x}) = \langle \Phi_{\mathbf{w}}(x), \Phi_{\mathbf{w}}(\tilde{x}) \rangle, \quad \text{for any inputs } x, \tilde{x}. \tag{18}$$

3.2. Lazy training

Suppose we train the network under gradient descent with the loss L and learning rate η, initialized at $\mathbf{w}_0$. Applying an expansion (16) around each iterate $\mathbf{w}_t$, we see that the function iterates $f_t := f_{\mathbf{w}_t}$, up to higher order terms (vanishing for gradient flow), follow the steepest gradient with respect to the time varying tangent kernel $k_t := k_{\mathbf{w}_t}$,

$$f_{t+1} = f_t - \eta_t K_t \cdot \nabla_{f_t} L + O(\eta^2) \tag{19}$$

where K_t denotes the operator defined as $(K_t \cdot g)(x) = \sum_{i=1}^n k_t(x, x_i) g(x_i)$. By contrast, training the linearized model at $\mathbf{w}_0$ corresponds to steepest descent with respect to a fixed kernel, the tangent kernel at initialization k_0,

$$\bar{f}_{t+1} = \bar{f}_t - \eta_t K_0 \cdot \nabla_{\bar{f}_t} L \tag{20}$$

The linearized dynamics (20) is a good approximation of the full dynamics as long as the parameter iterates $\mathbf{w}_t$ remain close to their initial value $\mathbf{w}_0$. Following the terminology of Chizat et al. (2019), *lazy training* refers to the situation where the approximation remains accurate until the training algorithm is stopped. In such a regime, the tangent kernel remains approximately constant during training, i.e., $k_t \approx k_0$ for all t and the model behaves like a kernel method.

Showing that the optimization of a deep network is in the lazy regime allows us to import well-known results from linear models, such as guarantees of convergence to a global optimum (Du et al., 2019b,a; Allen-Zhu et al., 2019). The inductive bias of learning can also be characterized by analyzing the tangent kernel and its reproducing kernel Hilbert space (RKHS) (Bietti & Mairal, 2019; Yang & Salman, 2019).

When does lazy training occur? Several works pointed out how the variance of the initial random weights controls the transition between the linearized and the full training dynamics (Chizat et al., 2019; Woodworth et al., 2020). The very notion of tangent kernel, however, arose from the study of infinite-width networks. Jacot et al. (2018) proved that, for an MLP with appropriate parametrization and scaling of the initial random weights, when the network widths grow to infinity, (i) the tangent kernel at initialization converges to a deterministic

limit; (*ii*) the tangent kernel remains constant during training by gradient descent. Yang (2020); Yang & Littwin (2021) extended this result to a large class of modern architectures and gave concrete algorithms to compute the tangent kernel in the limit. While the linearization is only exact in the infinite width limit, some empirical results (Lee et al., 2019) suggest that it can accurately describe the training of realistic finite-sized networks.

3.3. Deep learning versus kernel learning

On the other hand, it is also clear from other results that the kernel approximation does not fully capture the behavior of deep models – including, in fact, infinitely wide networks (Yang & Hu, 2021). For example, in the so-called mean field limit, training two-layer networks by gradient descent learns adaptive representations (Chizat & Bach, 2018; Mei et al., 2018) and it can be shown that the inductive bias cannot be characterized in terms of a RKHS norm (Savarese et al., 2019; Williams et al., 2019). Performance gaps between the two regimes are also often observed in practice (Chizat et al., 2019; Arora et al., 2019), although the architecture and structure of the data seem to play a key role in the relative performance (Geiger et al., 2020b).

An interesting challenge is to explain this performance gap. In other words, we seek to qualitatively and quantitatively characterize the impact of the non-linear dynamics in the feature learning regime. The local linearizations (19) with time varying tangent kernel can give useful insights on this problem. Several recent works, which includes the contribution in Chapter 4 and other concurrent works (Kopitkov & Indelman, 2020; Paccolat et al., 2021; Fort et al., 2020; Ortiz-Jiménez et al., 2021), study how tangent kernels evolve during training, in the full deep learning regime.

4. Information in Deep Learning

In this section we introduce some basic concepts from information theory (Shannon, 1948), as a background for Chapter 5. We use capital letters $X, Z...$ for random variables, curves letters $\mathcal{X}, \mathcal{Z} \cdots$ for the sets of possible outcomes $x, z \cdots$, and $\mathbb{P}_X, \mathbb{P}_Z \cdots$ for probability distributions. For discrete distributions, we also denote by $p(x)$ the probability mass function, i.e., $p(x) = \mathbb{P}_X(X = x)$. We use the same notation $p(x)$ for the density of a continuous distribution over $\mathcal{X} \subset \mathbb{R}^d$ that is absolutely continuous with respect to the Lebesgue measure.

4.1. Entropy and mutual information

Let X be a discrete random variable. The Shannon entropy is a measure of uncertainty of the distribution,

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) = -\mathbb{E}_x \left[\log p(x)\right] \tag{21}$$

where it is understood that $0 \log(0) = 0$. For a finite sample space $\mathcal{X}$, the entropy is maximum, $H_{\max} = \log |\mathcal{X}|$, for the uniform distribution; and it is minimum, $H_{\min} = 0$, for deterministic ones, i.e., $p(x) = \delta_{xx_0}$ for some $x_0 \in \mathcal{X}$. The definition (21) can be extended to continuous distributions on some domain of $\mathbb{R}^d$ with well-defined density p with respect to the Lebesgue measure dx. This yields the differential entropy, $H(p) = -\int p(x) \log p(x) dx$.

Mutual information is a Shannon entropy-based measure of dependence between random variables. The mutual information between X and Z can be understood as the decrease of the uncertainty in X given Z:

$$I(X; Z) := H(X) - H(X \mid Z), \tag{22}$$

where $H(X \mid Z) := \mathbb{E}_z [H(X|Z = z)]$ is the conditional entropy of X given Z. The mutual information can also be expressed as the Kullback-Leibler (KL-) divergence between the joint, $\mathbb{P}_{XZ}$, and the product of the marginals $\mathbb{P}_X \otimes \mathbb{P}_Z$:

$$I(X,Z) = D_{KL}(\mathbb{P}_{XZ} \,||\, \mathbb{P}_X \otimes \mathbb{P}_Z) \tag{23}$$

where D_{KL} is defined as[4]:

$$D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) := \mathbb{E}_{\mathbb{P}} \left[\log \frac{d\mathbb{P}}{d\mathbb{Q}} \right] = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{24}$$

The intuitive meaning of Eqn. (23) is clear: the larger the divergence between the joint and the product of the marginals, the stronger the dependence between X and Z. This divergence, hence the mutual information, vanishes for fully independent variables.

4.2. Neural estimation

Many statistical divergences such as f-divergences (Ali & Silvey, 1966) and integral probability metrics (Müller, 1997) have variational formulations as the maximum of some optimization problem over a large class of functions. Parametric lower-bound estimators can then be defined by restricting the class of functions to some parametric family $\mathcal{F}$. Neural estimation consists of choosing $\mathcal{F}$ to be a neural network and to optimize over the parameters.

Let us consider the case of f-divergences. Given a convex function $f \colon [0, \infty) \to \mathbb{R}$ such that $f(1) = 0$, these are defined as

$$D_f(\mathbb{P} \,||\, \mathbb{Q}) := \mathbb{E}_{\mathbb{Q}} \left[f\left(\frac{d\mathbb{P}}{d\mathbb{Q}} \right) \right] = \int q(x) f\left(\frac{p(x)}{q(x)} \right) dx. \tag{25}$$

when $\mathbb{P} \ll \mathbb{Q}$ and $+\infty$ otherwise. These include the KL-divergence (23) by setting $f(t) = t \log(t)$. Variational formulations arises from Fenchel convex duality (Rockafellar, 1970). Thus, if f is convex and lower semi-continuous, it can be written in terms of its dual conjugate

[4]For continuous distributions, the KL divergence is defined by (24) if $\mathbb{P} \ll \mathbb{Q}$ and $+\infty$ otherwise. Note that, for non-singular distributions, the joint is absolutely continuous with respect to the product of marginals.

as $f(t) = \sup_{u \in \mathbb{R}} \{ut - f^*(u)\}$; substituting into (25) yields a supremum over all measurable functions such that the integral involved is well-defined (Keziou, 2003; Nguyen et al., 2010). Alternatively (Ruderman et al., 2012), one can view the f-divergence itself as a convex operator $D_{f,\mathbb{P}} \colon r \mapsto \mathbb{E}_{\mathbb{Q}}\left[f(r)\right]$ acting on the space $\Delta(\mathbb{Q})$ of positive functions with unit 1-norm with respect to $\mathbb{Q}$ (just like the Radon Nykodym derivative $d\mathbb{P}/d\mathbb{Q}$), and write it in terms of its dual conjugate $D^*_{f,\mathbb{P}}(T) := \sup_{r \in \Delta(\mathbb{Q})} \{\mathbb{E}_{\mathbb{Q}}[rT] - D_{f,\mathbb{P}}(r)\}$. Interestingly, the latter approach yields tighter parametric lower bound estimators.

More explicitly, for the KL-divergence, such constructions yield the key properties that for any reasonable scalar function T,

$$D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) \geq \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T]) \geq \mathbb{E}_{\mathbb{P}}[T] - \mathbb{E}_{\mathbb{Q}}[e^T - 1] \tag{26}$$

The first inequality leads to the so-called Donsker-Varadhan representation (Donsker & Varadhan, 1983) of the divergence. It is tight for optimal functions T^* that relate the distributions to the *Gibbs density* as,

$$d\mathbb{P} = \frac{1}{Z} e^{T^*} d\mathbb{Q}, \text{ where } Z = \mathbb{E}_{\mathbb{Q}}[e^{T^*}]. \tag{27}$$

The second inequality leads to the dual f-divergence representation. It is tight for the optimal function $T^* = \log \frac{d\mathbb{P}}{d\mathbb{Q}}$. In both cases, we see that a tight estimation of the KL divergence amounts to an estimation of the likelihood ratio $d\mathbb{P}/d\mathbb{Q}$. Neural estimators are obtained by taking the supremum of these inequalities over the parameters $\theta \in \Theta$ of a neural network T_θ. Chapter 5 uses this approach to propose a general purpose parametric estimator of mutual information.

Chapter 3

First Article: On the Spectral Bias of Neural Networks

Prologue

Article Details. **On the Spectral Bias of Neural Networks**. Nasim Rahaman*, Aristide Baratin*, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, Aaron Courville. The paper has been published at ICML 2019 (Rahaman et al., 2019).

* Co-first authors.

Personal Contribution. The original idea of studying the evolution of the Fourier modes of neural networks is from Nasim Rahaman, who conceived this project. My main contributions was to shape the theoretical results, help with the genesis of the idea and propose experiments. I took the lead, along with Nasim and with inputs of our co-authors, in writing the paper.

Discussion and Recent Developments. This project follows a line of work pioneered by Neyshabur et al. (2015), arguing that the optimization dynamics in deep learning induces some sort of implicit regularization – which in turns allows for generalization. It can also be viewed as an attempt to formalize the observations of Arpit et al. (2017) that neural networks prioritize learning 'simple' patterns of the data during training. By decomposing the input-ouput map into Fourier modes and using Fourier frequency as notion of complexity, our results indeed highlight a learning bias towards 'simple' functions that gradually increase in complexity during training.

Several subsequent works delve further into the hypothesis of our paper and offer interesting complementary insights. For example, the experiments of Zhang et al. (2021) suggest an intriguing non-monotonicity of the spectral bias, which mirrors the so-called epoch-wise double descent of the test error observed in Nakkiran et al. (2020). It was also shown that, early in training, the network predictions are aligned to that of a simple linear prediction

rule (Nakkiran et al., 2019) ; and more generally that networks rely preferentially on linearly-predictive features (Shah et al., 2020).

1. Introduction

The remarkable success of deep neural networks at generalizing to natural data is at odds with the traditional notions of model complexity and their empirically demonstrated ability to fit arbitrary random data to perfect accuracy (Zhang et al., 2017a; Arpit et al., 2017). This has prompted recent investigations of possible implicit regularization mechanisms inherent in the learning process which induce a bias towards low complexity solutions (Neyshabur et al., 2015; Soudry et al., 2017; Poggio et al., 2018; Neyshabur et al., 2017a).

In this work, we take a slightly shifted view on implicit regularization by suggesting that low-complexity functions are *learned faster* during training by gradient descent. We expose this bias by taking a closer look at neural networks through the lens of Fourier analysis. While they can approximate arbitrary functions, we find that these networks prioritize learning the low frequency modes, a phenomenon that we call *spectral bias*. This bias manifests itself not just in the process of learning, but also in the parameterization of the model itself: in fact, we show that the lower frequency components of trained networks are more robust to random parameter perturbations. Finally, we also expose and analyze the rather intricate interplay between the spectral bias and the geometry of the data manifold by showing that high frequencies get easier to learn when the data lies on a lower-dimensional manifold of complex shape embedded in the input space of the model. We focus the discussion on networks with rectified linear unit (ReLU) activations, whose continuous piece-wise linear structure enables an analytic treatment.

Contributions

(1) We exploit the continuous piecewise-linear structure of ReLU networks to evaluate its Fourier spectrum (Section 2).

(2) We find empirical evidence of a *spectral bias*: i.e. lower frequencies are learned first. We also show that lower frequencies are more robust to random perturbations of the network parameters (Section 3).

(3) We study the role of the shape of the data manifold: we show how complex manifold shapes can facilitate the learning of higher frequencies and develop a theoretical understanding of this behavior (Section 4).

2. Fourier analysis of ReLU networks

2.1. Preliminaries

Throughout the paper we call 'ReLU network' a scalar function $f : \mathbb{R}^d \mapsto \mathbb{R}$ defined by a neural network with L hidden layers of widths $d_1, \cdots d_L$ and a single output neuron:

$$f(\mathbf{x}) = \left(T^{(L+1)} \circ \sigma \circ T^{(L)} \circ \cdots \circ \sigma \circ T^{(1)}\right)(\mathbf{x}) \tag{1}$$

where each $T^{(k)} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ is an affine function ($d_0 = d$ and $d_{L+1} = 1$) and $\sigma(\mathbf{u})_i = \max(0, u_i)$ denotes the ReLU activation function acting elementwise on a vector $\mathbf{u} = (u_1, \cdots u_n)$. In the standard basis, $T^{(k)}(\mathbf{x}) = W^{(k)}\mathbf{x} + \mathbf{b}^{(k)}$ for some weight matrix $W^{(k)}$ and bias vector $\mathbf{b}^{(k)}$.

ReLU networks are known to be continuous piece-wise linear (CPWL) functions, where the linear regions are convex polytopes (Raghu et al., 2016; Montufar et al., 2014; Zhang et al., 2018a; Arora et al., 2018a). Remarkably, the converse also holds: every CPWL function can be represented by a ReLU network (Arora et al., 2018a, Theorem 2.1), which in turn endows ReLU networks with universal approximation properties. Given the ReLU network f from Eqn. 1, we can make the piecewise linearity explicit by writing,

$$f(\mathbf{x}) = \sum_{\epsilon} 1_{P_\epsilon}(\mathbf{x}) \left(W_\epsilon \mathbf{x} + \mathbf{b}_\epsilon\right) \tag{2}$$

where ϵ is an index for the linear regions P_ϵ and 1_{P_ϵ} is the indicator function on P_ϵ. As shown in Appendix A.2 in more detail, each region corresponds to an *activation pattern*[1] of all hidden neurons of the network, which is a binary vector with components conditioned on the sign of the input of the respective neuron. The $1 \times d$ matrix W_ϵ is given by

$$W_\epsilon = W^{(L+1)} W_\epsilon^{(L)} \cdots W_\epsilon^{(1)} \tag{3}$$

where $W_\epsilon^{(k)}$ is obtained from the original weight $W^{(k)}$ by setting its j^{th} column to zero whenever the neuron j of the k^{th} layer is inactive.

2.2. Fourier Spectrum

In the following, we study the structure of ReLU networks in terms of their Fourier representation, $f(\mathbf{x}) := (2\pi)^{d/2} \int \tilde{f}(\mathbf{k})\, e^{i\mathbf{k}\cdot\mathbf{x}} \mathbf{dk}$, where $\tilde{f}(\mathbf{k}) := \int f(\mathbf{x})\, e^{-i\mathbf{k}\cdot\mathbf{x}} \mathbf{dx}$ is the Fourier transform[2]. Lemmas 3.1 and 3.2 yield the explicit form of the Fourier components (we refer to Appendix A.3 for the proofs and technical details).

[1] We adopt the terminology of Raghu et al. (2016); Montufar et al. (2014).

[2] Note that general ReLU networks need not be squared integrable: for instance, the class of two-layer ReLU networks represent an arrangement of hyperplanes (Montufar et al., 2014) and hence grow linearly as $x \to \infty$.

Lemma 3.1. *The Fourier transform of ReLU networks decomposes as,*

$$\tilde{f}(\mathbf{k}) = i \sum_{\epsilon} \frac{W_\epsilon \mathbf{k}}{k^2} \tilde{1}_{P_\epsilon}(\mathbf{k}) \tag{4}$$

where $k = \|\mathbf{k}\|$ *and* $\tilde{1}_P(\mathbf{k}) = \int_P e^{-i\mathbf{k}\cdot\mathbf{x}} \mathbf{dx}$ *is the Fourier transform of the indicator function of* P*.*

The Fourier transform of the indicator over linear regions appearing in Eqn. 4 are fairly intricate mathematical objects. Diaz et al. (2016) develop an elegant procedure for evaluating it in arbitrary dimensions via a recursive application of Stokes theorem. We describe this procedure in detail[3] in Appendix A.3.2, and present here its main corollary.

Lemma 3.2. *Let* P *be a full dimensional polytope in* $\mathbb{R}^d$*. Its Fourier spectrum takes the form:*

$$\tilde{1}_P(\mathbf{k}) = \sum_{n=0}^{d} \frac{D_n(\mathbf{k}) 1_{G_n}(\mathbf{k})}{k^n} \tag{5}$$

where G_n *is the union of* n*-dimensional subspaces that are orthogonal to some* n*-codimensional face of* P*,* $D_n : \mathbb{R}^d \to \mathbb{C}$ *is in* $\Theta(1)$ $(k \to \infty)$ *and* 1_{G_n} *the indicator over* G_n*.*

Lemmas 3.1, 3.2 together yield the main result of this section.

Theorem 3.3. *The Fourier components of the ReLU network* f_θ *with parameters* θ *is given by the rational function:*

$$\tilde{f}_\theta(\mathbf{k}) = \sum_{n=0}^{d} \frac{C_n(\theta, \mathbf{k}) 1_{H_n^\theta}(\mathbf{k})}{k^{n+1}} \tag{6}$$

where H_n^θ *is the union of* n*-dimensional subspaces that are orthogonal to some* n*-codimensional faces of some polytope* P_ϵ *and* $C_n(\cdot, \theta) : \mathbb{R}^d \to \mathbb{C}$ *is* $\Theta(1)$ $(k \to \infty)$*.*

Note that Eqn 6 applies to general ReLU networks with arbitrary width and depth[4].

Discussion. We make the following two observations. First, the spectral decay of ReLU networks is highly anisotropic in large dimensions. In almost all directions of $\mathbb{R}^d$, we have a k^{-d-1} decay. However, the decay can be as slow as k^{-2} in specific directions orthogonal to the $d-1$ dimensional faces bounding the linear regions[5].

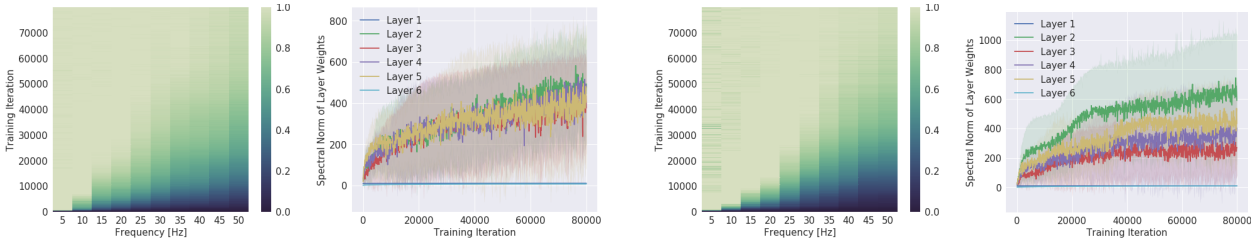
Second, the numerator in Eqn 6 is bounded by $N_f L_f$ (cf. Appendix A.3.3), where N_f is the number of linear regions and $L_f = \max_\epsilon \|W_\epsilon\|$ is the Lipschitz constant of the network.

In such cases, the Fourier transform is to be understood in the sense of tempered distributions acting on rapidly decaying smooth functions ϕ as $\langle \tilde{f}, \phi \rangle = \langle f, \tilde{\phi} \rangle$. See Appendix A.3 for a formal treatment.

[3]We also generalize the construction to tempered distributions.

[4]Symmetries that might arise due to additional assumptions can be used to further develop Eqn 6, see e.g. Eldan & Shamir (2016) for 2-layer networks.

[5]Note that such a rate is *not* guaranteed by piecewise smoothness alone. For instance, the function $\sqrt{|x|}$ is continuous and smooth everywhere except at $x = 0$, yet it decays as $k^{-1.5}$ in the Fourier domain.



(a) Equal Amplitudes **(b)** Increasing Amplitudes

Fig. 3.1. Left (a, b): Evolution of the spectrum (x-axis for frequency) during training (y-axis). The colors show the measured amplitude of the network spectrum at the corresponding frequency, normalized by the target amplitude at the same frequency (i.e. $|\tilde{f}_{k_i}|/A_i$) and the colorbar is clipped between 0 and 1. Right (a, b): Evolution of the spectral norm (y-axis) of each layer during training (x-axis). Figure-set (a) shows the setting where all frequency components in the target function have the same amplitude, and (b) where higher frequencies have larger amplitudes. **Gist**: We find that even when higher frequencies have larger amplitudes, the model prioritizes learning lower frequencies first. We also find that the spectral norm of weights increases as the model fits higher frequency, which is what we expect from Theorem 3.3.

Further, the Lipschitz constant L_f can be bounded as (cf. Appendix A.3.6):

$$L_f \leq \prod_{k=1}^{L+1} \|W^{(k)}\| \leq \|\theta\|_\infty^{L+1} \sqrt{d} \prod_{k=1}^{L} d_k \tag{7}$$

where $\|\cdot\|$ is the spectral norm and $\|\cdot\|_\infty$ the max norm, and d_k is the number of units in the k-th layer. This makes the bound on L_f scale exponentially in depth and polynomial in width. As for the number N_f of linear regions, Montufar et al. (2014) and Raghu et al. (2016) obtain tight bounds that exhibit the same scaling behaviour (Raghu et al., 2016, Theorem 1). In Appendix A.1.5, we qualitatively ablate over the depth and width of the network to expose how this reflects on the Fourier spectrum of the network.

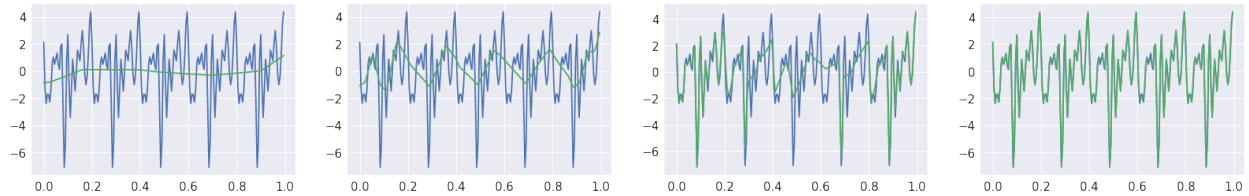
3. Lower Frequencies are Learned First

We now present experiments showing that networks tend to fit *lower frequencies first* during training. We refer to this phenomenon as the *spectral bias*, and discuss it in light of the results of Section 2.

3.1. Synthetic Experiments

Experiment 1. The setup is as follows[6]: Given frequencies $\kappa = (k_1, k_2, ...)$ with corresponding amplitudes $\alpha = (A_1, A_2, ...)$, and phases $\phi = (\varphi_1, \varphi_2, ...)$, we consider the mapping $\lambda : [0, 1] \rightarrow$

[6]More experimental details and additional plots are provided in Appendix A.1.1.



(a) Iteration 100 **(b)** Iteration 1000 **(c)** Iteration 10000 **(d)** Iteration 80000

Fig. 3.2. The learnt function (green) overlayed on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, ..., 45, 50)$, equal amplitudes and randomly sampled phases.

$\mathbb{R}$ given by

$$\lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i). \tag{8}$$

A 6-layer deep 256-unit wide ReLU network f_θ is trained to regress λ with $\kappa = (5, 10, ..., 45, 50)$ and $N = 200$ input samples spaced equally over $[0, 1]$; its spectrum $\tilde{f}_\theta(k)$ in expectation over $\varphi_i \sim U(0, 2\pi)$ is monitored as training progresses. In the first setting, we set equal amplitude $A_i = 1$ for all frequencies and in the second setting, the amplitude increases from $A_1 = 0.1$ to $A_{10} = 1$. Figure 3.1 shows the normalized magnitudes $|\tilde{f}_\theta(k_i)|/A_i$ at various frequencies, as training progresses with full-batch gradient descent. Further, Figure 3.2 shows the learned function at intermediate training iterations. The result is that lower frequencies (i.e. smaller k_i's) are regressed first, regardless of their amplitudes.

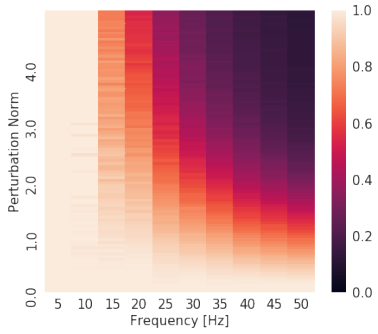
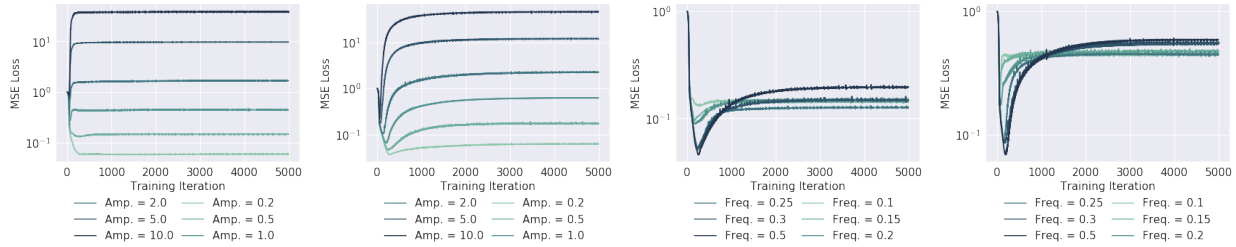


Fig. 3.3. Normalized spectrum of the model (x-axis for frequency, colorbar for magnitude) with perturbed parameters as a function of parameter perturbation (y-axis). The colormap is clipped between 0 and 1. We observe that the lower frequencies are more robust to parameter perturbations than the higher frequencies.

Experiment 2. Our goal here is to illustrate a phenomenon that complements the one highlighted above: lower frequencies are more *robust* to parameter perturbations. The set up is the same as in Experiment 1. The network is trained to regress a target function with frequencies $\kappa = (10, 15, 20, ..., 45, 50)$ and amplitudes $A_i = 1 \, \forall \, i$. After convergence to θ^*, we consider random (isotropic) perturbations $\theta = \theta^* + \delta\hat{\theta}$ of given magnitude δ, where $\hat{\theta}$ is a random unit vector in parameter space. We evaluate the network function f_θ at the perturbed



(a) $k = 0.1$ **(b)** $k = 1$ **(c)** $\beta = 0.5$ **(d)** $\beta = 1.$

Fig. 3.4. (a,b,c,d): Validation curves for various settings of noise amplitude β and frequency k. Corresponding training curves can be found in Figure A.2 in appendix A.1.3. **Gist**: Low frequency noise affects the network more than their high-frequency counterparts. Further, for high-frequency noise, one finds that the validation loss dips early in the training. Both these observations are explained by the fact that network readily fit lower frequencies, but learn higher frequencies later in the training.

parameters, and compute the magnitude of its discrete Fourier transform at frequencies k_i to obtain $|\tilde{f}_\theta(k_i)|$. We also average over 100 samples of $\hat{\theta}$ to obtain $|\tilde{f}_{\mathbb{E}\theta}(k_i)|$, which we normalize by $|\tilde{f}_{\theta*}(k_i)|$. Finally, we average over the phases ϕ (see Eqn 8). The result, shown in Figure 3.3, demonstrates that higher frequencies are significantly less robust than the lower ones, guiding the intuition that expressing higher frequencies requires the parameters to be finely-tuned to work together. In other words, parameters that contribute towards expressing high-frequency components occupy a small volume in the parameter space. We formalize this in Appendix A.4.

Discussion . Multiple theoretical aspects may underlie these observations. First, for a fixed architecture, recall that the numerator in Theorem 3.3 is[7] $\mathcal{O}(L_f)$ (where L_f is the Lipschitz constant of the function). However, L_f is bounded by the parameter norm, which can only increase gradually during training by gradient descent. This leads to the higher frequencies being learned[8] late in the optimization process. To confirm that the bound indeed increases as the model fits higher frequencies, we plot in Fig 3.1 the spectral norm of weights of each layer during training for both cases of constant and increasing amplitudes.

Second (cf. Appendix A.3.4), the exact form of the Fourier spectrum yields that for a fixed direction $\hat{\mathbf{k}}$, the spectral decay rate of the parameter gradient $\partial\tilde{f}/\partial\theta$ is at most one exponent of k lower than that of $\tilde{f}$. If for a fixed $\hat{\mathbf{k}}$ we have $\tilde{f} = \mathcal{O}(k^{-\Delta-1})$ where $1 \leq \Delta \leq d$, we obtain for the residual $h = f - \lambda$ and (continuous) training step t:

[7]The tightness of this bound is verified empirically in appendix A.1.5.
[8]This assumes that the Lipschitz constant of the (noisy) target function is larger than that of the network at initialization.

$$\left|\frac{d\tilde{h}(\mathbf{k})}{dt}\right| = \left|\frac{d\tilde{f}(\mathbf{k})}{dt}\right| = \underbrace{\left|\frac{d\tilde{f}(\mathbf{k})}{d\theta}\right|}_{\mathcal{O}(k^{-\Delta})} \overbrace{\left|\frac{d\theta}{dt}\right|}^{|\eta \cdot d\mathcal{L}/d\theta|} = \mathcal{O}(k^{-\Delta}) \tag{9}$$

where we use the fact that $d\theta/dt$ is just the learning rate times the parameter gradient of the loss which is independent[9] of k, and assume that the target function λ is fixed. Eqn 9 shows that the rate of change of the residual decays with increasing frequency, which is what we find in Experiment 1.

3.2. Real-Data Experiments

While Experiments 1 and 2 establish the spectral bias by explicitly evaluating the Fourier coefficients, doing so becomes prohibitively expensive for larger d (e.g. on MNIST). To tackle this, we propose the following set of experiments to measure the effect of spectral bias indirectly on MNIST.

Experiment 3. In this experiment, we investigate how the validation performance dependent on the frequency of noise added to the training target. We find that the best validation performance on MNIST is particularly insensitive to the magnitude of high-frequency noise, yet it is adversely affected by low-frequency noise. We consider a target (binary) function $\tau_0 : X \to \{0, 1\}$ defined on the space $X = [0, 1]^{784}$ of MNIST inputs. Samples $\{\mathbf{x}_i, \tau_0(\mathbf{x}_i)\}_i$ form a subset of the MNIST dataset comprising samples $\mathbf{x}_i$ belonging to two classes. Let $\psi_k(\mathbf{x})$ be a *noise function*:

$$\psi_k(\mathbf{x}) = \sin(k\|\mathbf{x}\|) \tag{10}$$

corresponding to a *radial wave* defined on the 784-dimensional input space[10]. The final target function τ_k is then given by $\tau_k = \tau_0 + \beta\psi_k$, where β is the effective amplitude of the noise. We fit the same network as in Experiment 1 to the target τ_k with the MSE loss. In the first set of experiments, we ablate over k for a pair of fixed βs, while in the second set we ablate over β for a pair of fixed ks. In Figure 3.4, we show the respective validation loss curves, where the validation set is obtained by evaluating τ_0 on a separate subset of the data, i.e. $\{\mathbf{x}_j, \tau_0(\mathbf{x}_j)\}_j$. Figure A.2 (in appendix A.1.3) shows the respective training curves.

Discussion. The profile of the loss curves varies significantly with the frequency of noise added to the target. In Figure 3.4a, we see that the validation performance is adversely

[9]Note however that the loss term might involve a sum or an integral over all frequencies, but the summation is over a different variable.

[10]The rationale behind using a radial wave is that it induces oscillations (simultaneously) along all spatial directions. Another viable option is to induce oscillations along the principle axes of the data: we have verified that the key trends of interest are preserved.

affected by the amplitude of the low-frequency noise, whereas Figure 3.4b shows that the amplitude of high-frequency noise does not significantly affect the best validation score. This is explained by the fact that the network readily fits the noise signal if it is low frequency, whereas the higher frequency noise is only fit later in the training. In the latter case, the dip in validation score early in the training is when the network has learned the low frequency true target function τ_0; the remainder of the training is spent learning the higher-frequencies in the training target τ, as we shall see in the next experiment. Figures 3.4c and 3.4d confirm that the dip in validation score exacerbates for increasing frequency of the noise. Further, we observe that for higher frequencies (e.g. $k = 0.5$), increasing the amplitude β does not significantly degrade the best performance at the dip, confirming that the network is fairly robust to the amplitude of high-frequency noise.

Finally, we note that the dip in validation score was also observed by Arpit et al. (2017) with i.i.d. noise[11] in a classification setting.

Experiment 4. To investigate the dip observed in Experiment 3, we now take a more direct approach by considering a generalized notion of frequency. To that end, we project the network function to the space spanned by the orthonormal eigenfunctions φ_n of the Gaussian RBF kernel Braun et al. (2006). These eigenfunctions φ_n (sorted by decreasing eigenvalues) resemble sinusoids Fasshauer (2011), and the index n can be thought of as being a proxy for the frequency, as can be seen from Figure 3.6 (see Appendix A.1.4 for additional details and supporting plots). While we will call $\tilde{f}[n]$ as the spectrum of the function f, it should be understood as $\tilde{f}[n] = \langle f_{\mathcal{H}}, \varphi_n \rangle_{\mathcal{H}}$, where $f_{\mathcal{H}} \in \text{span}\{\varphi_n\}_n$ and $f_{\mathcal{H}}(\mathbf{x}_i) = f(\mathbf{x}_i)$ on the MNIST samples $\mathbf{x}_i \in X$. This allows us to define a noise function as:

$$\psi_\gamma(\mathbf{x}) = \sum_n^N \left(\frac{n}{N}\right)^\gamma \varphi_n(\mathbf{x}) \tag{11}$$

where N is the number of available samples and $\gamma = 2$. Like in Experiment 3, the target function is given by $\tau = \tau_0 + \beta\psi$, and the same network is trained to regress τ. Figure 3.5 shows the (generalized) spectrum τ and τ_0, and that of f as training progresses. Figure A.4 (in appendix) shows the corresponding dip in validation loss, where the validation set is same as the training set but with true target function τ_0 instead of the noised target τ.

Discussion. From Figure 3.5, we learn that the drop in validation score observed in Figure 3.4 is exactly when the higher-frequencies of the noise signal are yet to be learned. As the network gradually learns the higher frequency eigenfunctions, the validation loss increases while the training loss continues to decrease. Thus these experiments show that the

[11]Recall that i.i.d. noise is white-noise, which has a constant Fourier spectrum magnitude in expectation, i.e. it also contains high-frequency components.

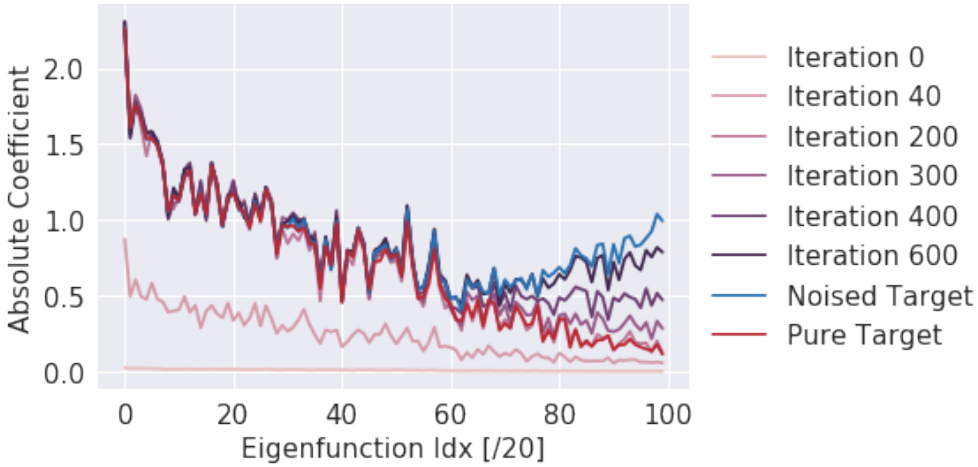


Fig. 3.5. Spectrum of the network as it is trained on MNIST target with high-frequency noise (*Noised Target*). We see that the network fits the true target at around the 200th iteration, which is when the validation score dips (Figure A.4 in appendix).

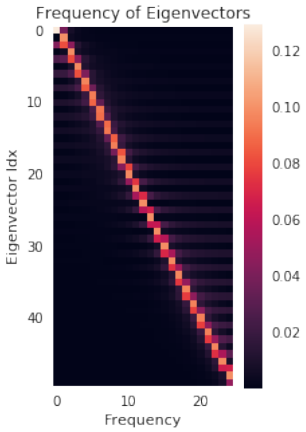


Fig. 3.6. Spectrum (x-axis for frequency, colorbar for magnitude) of the n-th (y-axis) eigenvector of the Gaussian RBF kernel matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, where the sample set is $\{x_i \in [0,1]\}_{i=1}^{50}$ is $N = 50$ uniformly spaced points between 0 and 1 and k is the Gaussian RBF kernel function. **Gist:** The eigenfunctions with increasing n roughly correspond to sinusoids of increasing frequency. Refer to Appendix A.1.4 for more details.

phenomenon of spectral bias persists on non-synthetic data and in high dimensional input spaces.

4. Not all Manifolds are Learned Equal

In this section, we investigate subtleties that arise when the data lies on a lower dimensional manifold embedded in the higher dimensional input space of the model. We find that the *shape* of the data-manifold impacts the learnability of high frequencies in a non-trivial way. As we shall see, this is because low frequency functions in the input space may have high

frequency components when restricted to lower dimensional manifolds of complex shapes. We demonstrate results in an illustrative minimal setting[12], free from unwanted confounding factors, and present a theoretical analysis of the phenomenon.

Manifold hypothesis. We consider the case where the data lies on a lower dimensional *data manifold* $\mathcal{M} \subset \mathbb{R}^d$ embedded in input space (Goodfellow et al., 2016b), which we assume to be the image $\gamma([0,1]^m)$ of some injective mapping $\gamma : [0,1]^m \to \mathbb{R}^d$ defined on a lower dimensional latent space $[0,1]^m$. Under this hypothesis and in the context of the standard regression problem, a target function $\tau : \mathcal{M} \to \mathbb{R}$ defined on the data manifold can be identified with a function $\lambda = \tau \circ \gamma$ defined on the latent space. Regressing τ is therefore equivalent to finding $f : \mathbb{R}^d \to \mathbb{R}$ such that $f \circ \gamma$ matches λ. Further, assuming that the data probability distribution μ supported on $\mathcal{M}$ is induced by γ from the uniform distribution U in the latent space $[0,1]^m$, the mean square error can be expressed as:

$$\begin{aligned} \mathrm{MSE}^{(\mathbf{x})}_{\mu}[f,\tau] = \mathbb{E}_{\mathbf{x}\sim\mu}|f(\mathbf{x}) - \tau(\mathbf{x})|^2 &= \\ \mathbb{E}_{\mathbf{z}\sim U}|(f(\gamma(\mathbf{z})) - \lambda(\mathbf{z})|^2 &= \mathrm{MSE}^{(\mathbf{z})}_{U}[f\circ\gamma,\lambda] \end{aligned} \tag{12}$$

Observe that there is a vast space of degenerate solutions f that minimize the mean squared error – namely all functions on $\mathbb{R}^d$ that yield the same function when restricted to the data manifold $\mathcal{M}$.

Our findings from the previous section suggest that neural networks are biased towards expressing a particular subset of such solutions, namely those that are low frequency. It is also worth noting that there exist methods that restrict the space of solutions: notably adversarial training (Goodfellow et al., 2014b) and Mixup (Zhang et al., 2017b).

Experimental set up. The experimental setting is designed to afford control over both the shape of the data manifold and the target function defined on it. We will consider the family of curves in $\mathbb{R}^2$ generated by mappings $\gamma_L : [0,1] \to \mathbb{R}^2$ given by

$$\begin{aligned} \gamma_L(z) = &\quad R_L(z)(\cos(2\pi z), \sin(2\pi z)) \\ \text{where} &\quad R_L(z) = 1 + \tfrac{1}{2}\sin(2\pi L z) \end{aligned} \tag{13}$$

Here, $\gamma_L([0,1])$ defines the data-manifold and corresponds to a flower-shaped curve with L petals, or a unit circle when $L = 0$ (see e.g. Fig 3.7). Given a signal $\lambda : [0,1] \to \mathbb{R}$ defined on the latent space $[0,1]$, the task entails learning a network $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f \circ \gamma_L$ matches the signal λ.

Experiment 5. The set-up is similar to that of Experiment 1, and λ is as defined in Eqn. 8 with frequencies $\kappa = (20, 40, ..., 180, 200)$, and amplitudes $A_i = 1 \,\forall\, i$. The model f is trained

[12]We include additional experiments on MNIST and CIFAR-10 in appendices A.1.6 and A.1.7.

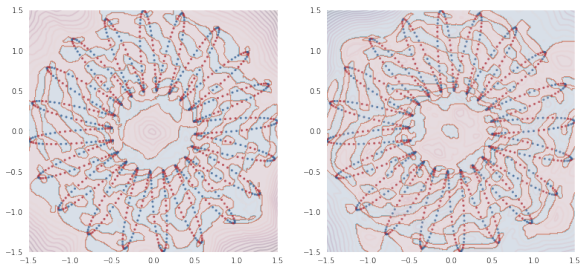
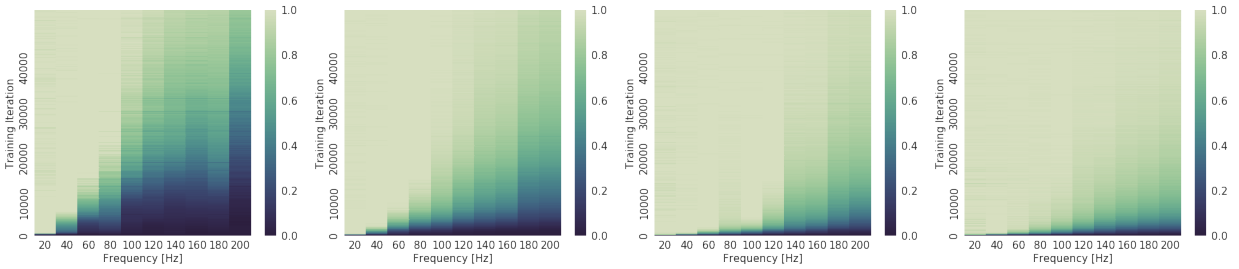
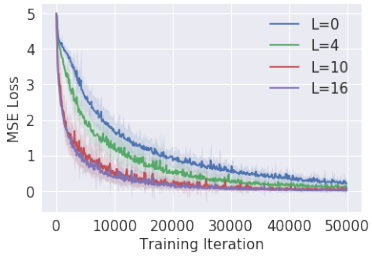


Fig. 3.7. Functions learned by two identical networks (up to initialization) to classify the binarized value of a sine wave of frequency $k = 200$ defined on a $\gamma_{L=20}$ manifold. Both yield close to perfect accuracy for the samples defined on the manifold (scatter plot), yet they differ significantly elsewhere. The shaded regions show the predicted class (Red or Blue) whereas contours show the confidence (absolute value of logits).



(a) $L = 0$ **(b)** $L = 4$ **(c)** $L = 10$ **(d)** $L = 16$



(e) Loss curves

Fig. 3.8. (a,b,c,d): Evolution of the network spectrum (x-axis for frequency, colorbar for magnitude) during training (y-axis) for the same target functions defined on manifolds γ_L for various L. Since the target function has amplitudes $A_i = 1$ for all frequencies k_i plotted, the colorbar is clipped between 0 and 1. (e): Corresponding learning curves. **Gist**: Some manifolds (here with larger L) make it easier for the network to learn higher frequencies than others.

on the dataset $\{\gamma_L(z_i), \lambda(z_i)\}_{i=1}^N$ with $N = 1000$ uniformly spaced samples z_i between 0 and 1. The spectrum of $f \circ \gamma_L$ in expectation over $\varphi_i \sim U(0, 2\pi)$ is monitored as training progresses, and the result is shown in Fig 3.8 for various L. Fig 3.8e shows the corresponding mean squared error curves. More experimental details in appendix A.1.2.

The results demonstrate a clear attenuation of the spectral bias as L grows. Moreover, Fig 3.8e suggests that the larger the L, the easier the learning task.

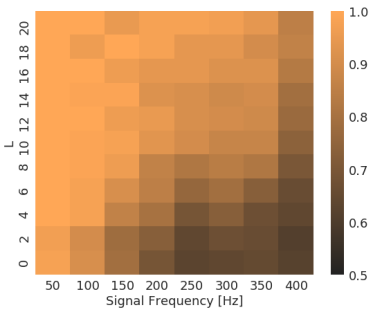


Fig. 3.9. Heatmap of training accuracies of a network trained to predict the binarized value of a sine wave of given frequency (x-axis) defined on γ_L for various L (y-axis).

Experiment 6. Here, we adapt the setting of Experiment 5 to binary classification by simply thresholding the function λ at 0.5 to obtain a binary target signal. To simplify visualization, we only use signals with a single frequency mode k, such that $\lambda(z) = \sin(2\pi kz + \varphi)$. We train the same network on the resulting classification task with cross-entropy loss[13] for $k \in \{50, 100, ..., 350, 400\}$ and $L \in \{0, 2, ..., 18, 20\}$. The heatmap in Fig 3.9 shows the classification accuracy for each (k, L) pair. Fig 3.7 shows visualizations of the functions learned by the same network, trained on $(k, L) = (200, 20)$ under identical conditions up to random initialization.

Observe that increasing L (i.e. going up a column in Fig 3.9) results in better (classification) performance for the same target signal. This is the same behaviour as we observed in Experiment 5 (Fig 3.8a-d), but now with binary cross-entropy loss instead of the MSE.

Discussion. These experiments hint towards a rich interaction between the shape of the manifold and the effective difficulty of the learning task. The key mechanism underlying this phenomenon (as we formalize below) is that the relationship between frequency spectrum of the network f and that of the fit $f \circ \gamma_L$ is mediated by the embedding map γ_L. In particular, we argue that a given signal defined on the manifold is easier to fit when the coordinate functions of the manifold embedding itself has high frequency components. Thus, in our experimental setting, the same signal embedded in a flower with more petals can be captured with lower frequencies of the network.

To understand this mathematically, we address the following questions: given a target function λ, how small can the frequencies of a solution f be such that $f \circ \gamma = \lambda$? And further, how does this relate to the geometry of the data-manifold $\mathcal{M}$ induced by γ? To find out, we

[13] We use Pytorch's `BCEWithLogitsLoss`. Internally, it takes a sigmoid of the network's output (the logits) before evaluating the cross-entropy.

write the Fourier transform of the composite function,

$$\begin{aligned} \widetilde{(f \circ \gamma)}(\mathbf{l}) &= \int \mathbf{dk} \tilde{f}(\mathbf{k}) P_\gamma(\mathbf{l}, \mathbf{k}) \\ \text{where } P_\gamma(\mathbf{l}, \mathbf{k}) &= \int_{[0,1]^m} \mathbf{dz}\, e^{i(\mathbf{k}\cdot\gamma(\mathbf{z}) - \mathbf{l}\cdot\mathbf{z})} \end{aligned} \tag{14}$$

The kernel P_γ depends on only γ and elegantly encodes the correspondence between frequencies $\mathbf{k} \in \mathbb{R}^d$ in input space and frequencies $\mathbf{l} \in \mathbb{R}^m$ in the latent space $[0,1]^m$. Following a procedure from Bergner et al., we can further investigate the behaviour of the kernel in the regime where the stationary phase approximation is applicable, i.e. when $l^2 + k^2 \to \infty$ (cf. section 3.2. of Bergner et al.). In this regime, the integral P_γ is dominated by critical points $\bar{\mathbf{z}}$ of its phase, which satisfy

$$\mathbf{l} = J_\gamma(\bar{\mathbf{z}})\, \mathbf{k} \tag{15}$$

where $J_\gamma(\mathbf{z})_{ij} = \nabla_i \gamma_j(\mathbf{z})$ is the $m \times d$ Jacobian matrix of γ. Non-zero values of the kernel correspond to pairs $(\mathbf{l}, \mathbf{k})$ such that Eqn 15 has a solution. Further, given that the components of γ (i.e. its coordinate functions) are defined on an interval $[0,1]^m$, one can use their Fourier series representation together with Eqn 15 to obtain a condition on their frequencies (shown in appendix A.3.7). More precisely, we find that the i-th component of the RHS in Eqn 15 is proportional to $\mathbf{p}\tilde{\gamma}_i[\mathbf{p}]k_i$ where $\mathbf{p} \in \mathbb{Z}^m$ is the frequency of the coordinate function γ_i. This yields that we can get arbitrarily large frequencies l_i if $\tilde{\gamma}_i[\mathbf{p}]$ is large[14] enough for large $\mathbf{p}$, even when k_i is fixed.

This is precisely what Experiments 5 and 6 demonstrate in a minimal setting. From Eqn 13, observe that the coordinate functions have a frequency mode at L. For increasing L, it is apparent that the frequency magnitudes l (in the latent space) that can be expressed with the same frequency k (in the input space) increases with increasing L. This allows the remarkable interpretation that the neural network function can express large frequencies on a manifold (l) with smaller frequencies w.r.t its input domain (k), provided that the coordinate functions of the data manifold embedding itself has high-frequency components.

5. Related Work

A number of works have focused on showing that neural networks are capable of approximating arbitrarily complex functions. Hornik et al. (1989); Cybenko (1989); Leshno et al. (1993) have shown that neural networks can be universal approximators when given sufficient width; more recently, Lu et al. (2017) proved that this property holds also for width-bounded networks. Montufar et al. (2014) showed that the number of linear regions of deep ReLU networks grows polynomially with width and exponentially with depth; Raghu et al. (2016) generalized this result and provided asymptotically tight bounds. There have been various

[14] Consider that the data-domain is bounded, implying that $\tilde{\gamma}$ cannot be arbitrarily scaled.

results of the benefits of depth for efficient approximation (Poole et al., 2016; Telgarsky, 2016; Eldan & Shamir, 2016). These analysis on the expressive power of deep neural networks can in part explain why over-parameterized networks can perfectly learn random input-output mappings (Zhang et al., 2017a).

Our work more directly follows the line of research on implicit regularization in neural networks trained by gradient descent (Neyshabur et al., 2015; Soudry et al., 2017; Poggio et al., 2018; Neyshabur et al., 2017a). In fact, while our Fourier analysis of deep ReLU networks also reflects the width and depth dependence of their expressivity, we focused on showing a learning bias of these networks towards simple functions with dominant lower frequency components. We view our results as a first step towards formalizing the findings of Arpit et al. (2017), where it is empirically shown that deep networks prioritize learning simple patterns of the data during training.

A few other works studied neural networks through the lens of harmonic analysis. For example, Candès (1999) used the ridgelet transform to build constructive procedures for approximating a given function by neural networks, in the case of oscillatory activation functions. This approach has been recently generalized to unbounded activation functions by Sonoda & Murata (2017). Eldan & Shamir (2016) use insights on the support of the Fourier spectrum of two-layer networks to derive a worse-case depth-separation result. Barron (1993) makes use of Fourier space properties of the target function to derive an architecture-dependent approximation bound. In a concurrent and independent work, Xu et al. (2018) make the same observation that lower frequencies are learned first. The subsequent work by Xu (2018) proposes a theoretical analysis of the phenomenon in the case of 2-layer networks with sigmoid activation, based on the spectrum of the sigmoid function.

In light of our findings, it is worth comparing the case of neural networks and other popular algorithms such that kernel machines (KM) and K-nearest neighbor classifiers. We refer to the Appendix A.5 for a detailed discussion and references. In summary, our discussion there suggests that 1. DNNs strike a good balance between function smoothness and expressivity/parameter-efficiency compared with KM; 2. DNNs learn a smoother function compared with KNNs since the spectrum of the DNN decays faster compared with KNNs in the experiments shown there.

6. Conclusion

We studied deep ReLU networks through the lens of Fourier analysis. Several conclusions can be drawn from our analysis. While neural networks can approximate arbitrary functions, we find that they favour *low frequency* ones – hence they exhibit a bias towards smooth functions – a phenomenon that we called *spectral bias*. We also illustrated how the geometry

of the data manifold impacts expressivity in a non-trivial way, as high frequency functions defined on complex manifolds can be expressed by lower frequency network functions defined in input space.

We view future work that explore the properties of neural networks in Fourier domain as promising. For example, the Fourier transform affords a natural way of measuring how fast a function can change within a small neighborhood in its input domain; as such, it is a strong candidate for quantifying and analyzing the *sensitivity* of a model – which in turn provides a natural measure of complexity (Novak et al., 2018). We hope to encourage more research in this direction.

Chapter 4

Second Article: Neural Tangent Feature Alignment

Prologue

Article Details. **Implicit Regularization via Neural Tangent Feature Alignment**. Aristide Baratin*, Thomas George*, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, Simon Lacoste-Julien. The paper has been published at AISTATS 2021.

* Co-first authors.

Personal contribution. I conceived and lead the project, contributed to all theoretical results and to the design of all experiments. Thomas George took the lead in implementing the experiments of Sections 3 and 4.2 and produced all corresponding figures. We used the NNGeometry library (George, 2021) for efficient evaluation of tangent kernels, which Thomas developed independently during the course of the project. I did most of the writing, with assistance from Thomas and all our co-authors.

Discussion and Recent Developments. A limitation of the work presented in the previous chapter is the somewhat ad-hoc notion of complexity based on Fourier decomposition. Here we take a new look at the spectral bias through the lens of tangent kernels (Jacot et al., 2018). This idea was concurrently explored by Bietti & Mairal (2019); Basri et al. (2019); Yang & Salman (2019), where spectral analysis of the tangent kernel have been investigated in the linearized regime where it remains constant during training (Jacot et al., 2018; Du et al., 2019b; Allen-Zhu et al., 2019), and for architectures and data (e.g, uniform data on the sphere) allowing for explicit computations (e.g., decomposition in terms of spherical harmonics). This is a setup where the spectral bias can be rigorously analyzed, i.e., we can get explicit information of the type of functions that are learned quickly and generalize well.

Even though we do not expect standard training to remain in the linearized regime, this pespective can also explain some of the observations of the previous chapter: for example, Fig. B.2 in Appendix B.1.5 illustrates that for a randomly initialized MLP on 1D uniform data, the tangent kernel's eigenvectors (evaluated using a large test set) ranked in decreasing order of the eigenvalues align well with sinusoids of increasing frequencies. This seems to be a general feature of smooth kernels on uniform data (Braun, 2005) (e.g., RBF kernels, see Fig 3.6 in Chap 3).

The main focus of the work below, however, is to study *deviations* from the linear regime, and the impact of feature learning. The strategy is to look at the evolution of the tangent kernel and its spectral decomposition during training, using metrics that are known to be correlated to performance in kernel learning. One such metric is the alignment with the class labels, which has been used for kernel selection (Cristianini et al., 2002; Cortes et al., 2012). Intuitively, it measures both how much the label vector is contained in the subspace spanned by the first kernel eigenvectors, and how much the corresponding eigenvalues contribute to the whole spectrum. Our results can also be understood in geometric terms, since tangent kernel and Fisher information metric share the non-zero part of their spectrum.

Several recent works follow a similar line and offer complimentary insights. The experiments of Fort et al. (2020) suggest that the tangent kernel evolves very rapidly in the first few epochs, before slowing down and stabilizing later in training. Paccolat et al. (2021); Shan & Bordelon (2021) study more explicitly, on tractable models, the alignment and compression mechanism that we highlight here. The experiments of Ortiz-Jiménez et al. (2021) also complement ours. In particular, while we argue in our work that the alignment of the tangent kernel is a form of implicit regularization, they illustrate with a toy example a scenario where the tangent kernel dynamics overfits the target vector and degrades the performance of the linearized network. More generally, as we mention in the conclusion below, a generic situation where we expect feature alignment to be detrimental is in the presence of spurious input-label correlations in the training data. Exploring the feature learning dynamics in this context is an important direction for future work.

1. Introduction

One important property of deep neural networks is their ability to generalize well on real data. Surprisingly, this is even true with very high-capacity networks *without explicit regularization* (Neyshabur et al., 2015; Zhang et al., 2017a; Hoffer et al., 2017). This seems at odds with the usual understanding of the bias-variance trade-off (Geman et al., 1992; Neal et al., 2018; Belkin et al., 2019a): highly complex models are expected to overfit the training data and perform poorly on test data (Hastie et al., 2009). Solving this apparent paradox

requires understanding the various learning biases induced by the training procedure, which can act as implicit regularizers (Neyshabur et al., 2015, 2017b).

In this paper, we help clarify one such implicit regularization mechanism, by examining the evolution of the *neural tangent features* (Jacot et al., 2018) learned by the network along the optimization paths. Our results can be understood from two complementary perspectives: a *geometric* perspective – the (uncentered) covariance of the tangent features defines a metric on the function class, akin to the Fisher information metric (e.g., Amari, 2016); and a *functional* perspective – through the tangent kernel and its RKHS. In standard supervised classification settings, our main observation is a dynamical alignment of the tangent features along a small number of task-relevant directions during training. We interpret this phenomenon as a combined mechanism of *feature selection* and *compression*. The intuition motivating this work is that such a mechanism allows large models to adapt their capacity to the task, which in turn underpins their generalization abilities.

Specifically, our main contributions are as follows:

(1) Through experiments with various architectures on MNIST and CIFAR10, we give empirical insights on how the tangent features and their kernel adapt to the task during training (Section 3). We observe in particular a sharp increase of the anisotropy of their spectrum early in training, as well as an increasing similarity with the class labels, as measured by *centered kernel alignment* (Cortes et al., 2012).

(2) Drawing upon intuitions from linear models (Section 4.1), we argue that such a dynamical alignment acts as *implicit regularizer*. We motivate a new heuristic complexity measure which captures this phenomenon, and empirically show better correlation with generalization compared to various measures proposed in the recent literature (Section 4).

2. Preliminaries

Let $\mathcal{F}$ be a class of functions (e.g a neural network) parametrized by $\mathbf{w} \in \mathbb{R}^P$. We restrict here to *scalar* functions $f_{\mathbf{w}} \colon \mathcal{X} \to \mathbb{R}$ to keep notation light.[1]

Tangent Features. We define the **tangent features** as the function gradients w.r.t the parameters,

$$\Phi_{\mathbf{w}}(\mathbf{x}) := \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}) \in \mathbb{R}^P. \tag{1}$$

The corresponding kernel $k_{\mathbf{w}}(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \Phi_{\mathbf{w}}(\mathbf{x}), \Phi_{\mathbf{w}}(\tilde{\mathbf{x}}) \rangle$ is the **tangent kernel** (Jacot et al., 2018). Intuitively, the tangent features govern how small changes in parameter affect the

[1]The extension to vector-valued functions, relevant for the multiclass classification setting, is presented in Appendix B.1, along with more mathematical details.

function's outputs,

$$\delta f_{\mathbf{w}}(\mathbf{x}) = \langle \delta \mathbf{w}, \Phi_{\mathbf{w}}(\mathbf{x}) \rangle + O(\|\delta \mathbf{w}\|^2). \tag{2}$$

More formally, the (uncentered) covariance matrix $g_{\mathbf{w}} = \mathbb{E}_{\mathbf{x}\sim\rho}\left[\Phi_{\mathbf{w}}(\mathbf{x})\Phi_{\mathbf{w}}(\mathbf{x})^\top\right]$ w.r.t the input distribution ρ acts as a **metric tensor** on $\mathcal{F}$: assuming $\mathcal{F} \subset L^2(\rho)$, this is the metric induced on $\mathcal{F}$ by pullback of the L^2 scalar product. It characterizes the geometry of the function class $\mathcal{F}$. Metric (as symmetric $P \times P$ matrices) and tangent kernels (as rank P integral operators) share the same spectrum (see Prop B.1 in Appendix B.1.3).

Spectral Bias. The structure of the tangent features impacts the evolution of the function during training. To formalize this, we introduce the covariance eigenvalue decomposition $g_{\mathbf{w}} = \sum_{j=1}^P \lambda_{\mathbf{w}j} \boldsymbol{v}_{\mathbf{w}j} \boldsymbol{v}_{\mathbf{w}j}^\top$, which summarizes the predominant directions in parameter space. Given n input samples $(\mathbf{x}_i)$ and $\mathbf{f}_{\mathbf{w}} \in \mathbb{R}^n$ the vector of outputs $f_{\mathbf{w}}(\mathbf{x}_i)$, consider gradient descent updates $\delta \mathbf{w}_{\text{GD}} = -\eta \nabla_{\mathbf{w}} L$ for some cost function $L := L(\mathbf{f}_{\mathbf{w}})$. The following elementary result (see Appendix B.1.5) shows how the corresponding function updates in the linear approximation (2), $\delta f_{\text{GD}}(\mathbf{x}) := \langle \delta \mathbf{w}_{\text{GD}}, \Phi_{\mathbf{w}}(\mathbf{x}) \rangle$, decompose in the **eigenbasis**[2] of the tangent kernel:

$$u_{\mathbf{w}j}(\mathbf{x}) = \frac{1}{\sqrt{\lambda_{\mathbf{w}j}}} \langle \boldsymbol{v}_{\mathbf{w}j}, \Phi_{\mathbf{w}}(\mathbf{x}) \rangle \tag{3}$$

Lemma 4.1 (Local Spectral Bias)**.** *The function updates decompose as* $\delta f_{GD}(\mathbf{x}) = \sum_{j=1}^P \delta f_j u_{\mathbf{w}j}(\mathbf{x})$ *with*

$$\delta f_j = -\eta \lambda_{\mathbf{w}j} (\boldsymbol{u}_{\mathbf{w}j}^\top \nabla_{\mathbf{f}_{\mathbf{w}}} L), \tag{4}$$

where $\boldsymbol{u}_{\mathbf{w}j} = [u_{\mathbf{w}j}(\mathbf{x}_1), \cdots u_{\mathbf{w}j}(\mathbf{x}_n)]^\top \in \mathbb{R}^n$ *and* $\nabla_{\mathbf{f}_{\mathbf{w}}}$ *denotes the gradient w.r.t the sample outputs.*

This illustrates how, from the point of view of function space, the metric/tangent kernel eigenvalues act as a mode-specific rescaling $\eta\lambda_{\mathbf{w}j}$ of the learning rate.[3] This is a local version of a well-known bias for linear models trained by gradient descent (e.g in linear regression, see Appendix B.1.5.2), which prioritizes learning functions within the top eigenspaces of the kernel. Several recent works (Bietti & Mairal, 2019; Basri et al., 2019; Yang & Salman, 2019) investigated such bias for neural networks, in *linearized* regimes where the tangent kernel remains constant during training (Jacot et al., 2018; Du et al., 2019b; Allen-Zhu et al., 2019). As a simple example, for a randomly initialized MLP on 1D uniform data, Fig. B.2 in Appendix B.1.5 shows an alignment of the tangent kernel eigenfunctions with Fourier modes of increasing frequency, in line with prior empirical observations (Rahaman et al., 2019; Xu et al., 2019) of a 'spectral bias' towards low-frequency functions.

[2]The functions $(u_{\mathbf{w}j})_{j=1}^P$ form an orthonormal family in $L^2(\rho)$, i.e. $\mathbb{E}_{\mathbf{x}\sim\rho}[u_{\mathbf{w}j}u_{\mathbf{w}j'}] = \delta_{jj'}$, and yield the spectral decomposition $k_{\mathbf{w}}(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{j=1}^P \lambda_{\mathbf{w}j} u_{\mathbf{w}j}(\mathbf{x}) u_{\mathbf{w}j}(\tilde{\mathbf{x}})$ of the tangent kernel as an integral operator (see Appendix B.1.3).

[3]Intuitively, the eigenvalue $\lambda_{\mathbf{w}j}$ can be thought of as defining a local 'learning speed' for the mode j.

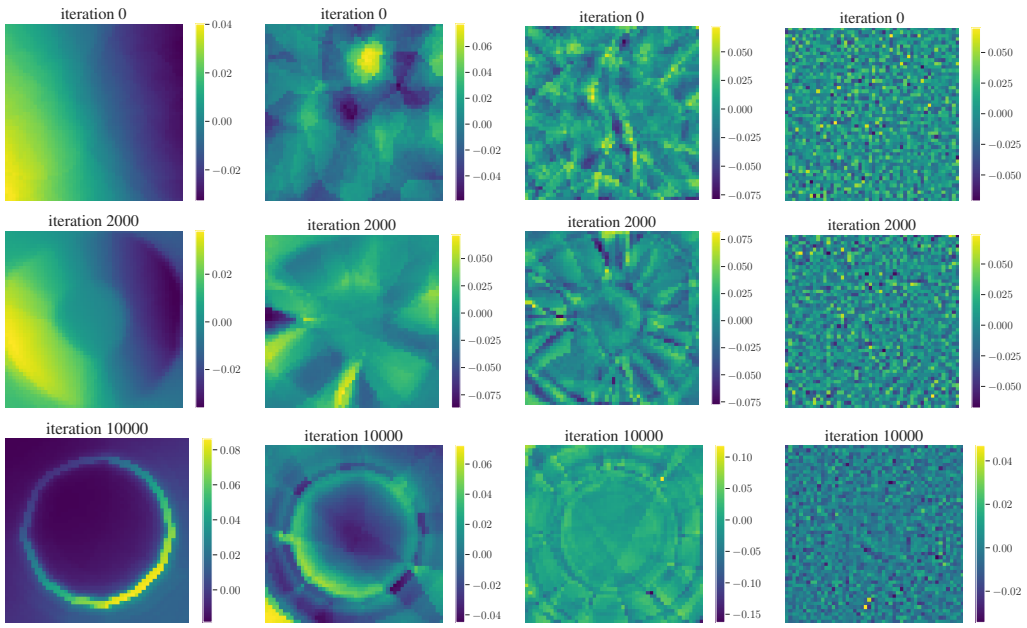


Fig. 4.1. Evolution of eigenfunctions of the tangent kernel, ranked in nonincreasing order of the eigenvalues (**in columns**), at various iterations during training (**in rows**), for the $2d$ Disk dataset. After a number of iterations, we observe modes corresponding to the class structure (e.g. boundary circle) in the top eigenfunctions. Combined with an increasing anistropy of the spectrum (e.g $\lambda_{20}/\lambda_1 = 1.5\%$ at iteration 0, 0.2% at iteration 2000), this illustrates a stretch of the tangent kernel, hence a (soft) compression of the model, along a small number of features that are highly correlated with the classes.

Tangent Features Adapt to the Task. By contrast, our aim in this paper is to highlight and discuss *non-linear* effects, in the (standard) regime where the tangent features and their kernel evolve during training (e.g., Geiger et al., 2020a; Woodworth et al., 2020).

As a first illustration of such effects, Fig. 4.1 shows visualizations of eigenfunctions of the tangent kernel (ranked in nonincreasing order of the eigenvalues), during training of a 6-layer deep 256-unit wide MLP by gradient descent of the binary cross entropy loss, on a simple classification task: $y(\mathbf{x}) = \pm 1$ depending on whether $\mathbf{x} \sim \text{Unif}[-1,1]^2$ is in the centered disk of radius $\sqrt{2/\pi}$ (details in Appendix B.3.1). After a number of iterations, we observe (rotation invariant) modes corresponding to the class structure (e.g. boundary circle) showing up in the *top* eigenfunctions of the learned kernel. We also note an increasing spectrum anisotropy – for example, the ratio λ_{20}/λ_1, which is 1.5% at iteration 0, has dropped to 0.2% at iteration 2000. The interpretation is that the tangent kernel (and the metric) *stretch* along a relatively small number of directions that are highly correlated with the classes during training. We quantify and investigate this effect in more detail below.

3. Neural Feature Alignment

In this section, we study in more detail the evolution of the tangent features during training. Our main results are to highlight (*i*) a sharp increase of the anisotropy of their spectrum early in training; (*ii*) an increasing similarity with the class labels, as measured by **centered kernel alignment** (CKA) (Cristianini et al., 2002; Cortes et al., 2012). We interpret this as a combined mechanism of feature selection and model compression.

3.1. Setup

We run experiments on MNIST (LeCun et al., 2010) and CIFAR10 (Krizhevsky & Hinton, 2009) with standard MLPs, VGG (Simonyan & Zisserman, 2014) and Resnet (He et al., 2016) architectures, trained by stochastic gradient descent (SGD) with momentum, using cross-entropy loss. We use PyTorch (Paszke et al., 2019) and NNGeometry (George, 2021) for efficient evaluation of tangent kernels.

In multiclass settings, tangent kernels evaluated on n samples carry additional class indices $y \in \{1 \cdots c\}$ and thus are $nc \times nc$ matrices, $(\boldsymbol{K}_{\mathbf{w}})_{ij}^{yy'} := k_{\mathbf{w}}(\mathbf{x}_i, y; \mathbf{x}_j, y')$ (details in Appendix B.1.4). In all our experiments, we evaluate tangent kernels on mini-batches of size $n = 100$ from both the training set and the test set; for $c = 10$ classes, this yields kernel matrices of size 1000×1000. We report results obtained from *centered* tangent features $\Phi_{\mathbf{w}}(\mathbf{x}) \to \Phi_{\mathbf{w}}(\mathbf{x}) - \mathbb{E}_{\mathbf{x}}\Phi_{\mathbf{w}}(\mathbf{x})$, though we obtain qualitatively similar results for uncentered features (see plots in Appendix B.3.2).

3.2. Spectrum Evolution

We first investigate the evolution of the tangent kernel *spectrum* for a VGG19 on CIFAR 10, trained with and without label noise (Fig. 4.2). The take away is an anisotropic increase of the spectrum during training. We report results for kernels evaluated on training examples (solid line) and test examples (dashed line).[4]

The first observation is a significant *increase* of the spectrum, early in training (note the log scale for the x-axis). By the time the model reaches 100% training accuracy, the maximum and average eigenvalues (Fig. 4.2, 2nd row) have gained more than 2 orders of magnitude.

The second observation is that this evolution is highly *anisotropic*, i.e larger eigenvalues increase faster than lower ones. This results in a (sharp) increase of spectrum anisotropy,

[4]The striking similarity of the plots for train and test kernels suggests that the spectrum of empirical tangent kernels is robust to sampling variations in our setting.

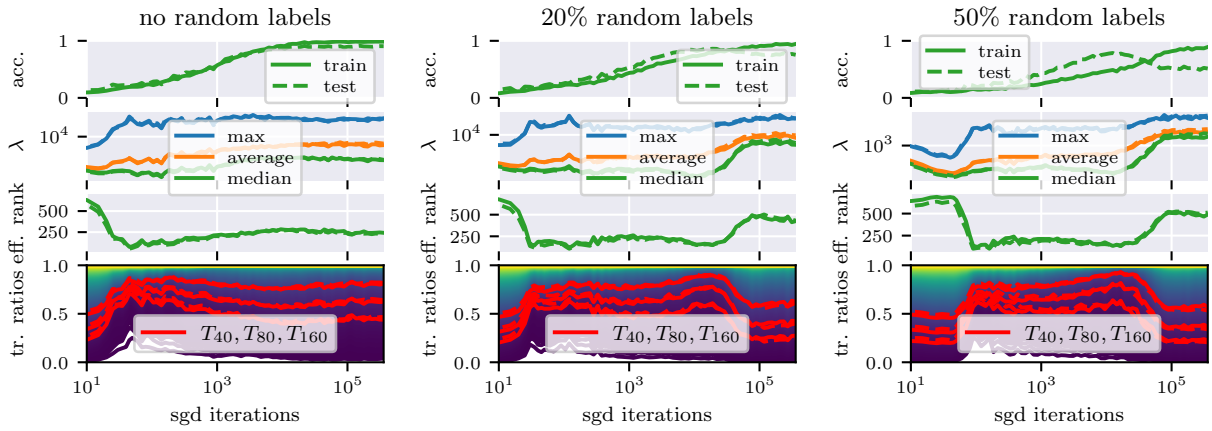


Fig. 4.2. Evolution of the tangent kernel **spectrum** (max, average and median eigenvalues), **effective rank** (5) and **trace ratios** (6) during training of a VGG19 on CIFAR10 with various ratio of random labels, using cross-entropy and SGD with batch size 100, learning rate 0.01 and momentum 0.9. Tangent kernels are evaluated on batches of size 100 from both the training set (solid lines) and the test set (dashed lines). The plots in the top row show train/test accuracy.

early in training. We quantify this using a notion of **effective rank** based on spectral entropy (Roy & Vetterli, 2007). Given a kernel matrix $\boldsymbol{K}$ in $\mathbb{R}^{r\times r}$ with (strictly) positive eigenvalues $\lambda_1, \cdots, \lambda_r$, let $\mu_j = \lambda_j / \sum_{i=1}^r \lambda_j$ be the trace-normalized eigenvalues. The effective rank is defined as $\text{erank} = \exp(H(\boldsymbol{\mu}))$ where $H(\boldsymbol{\mu})$ is the Shannon entropy,

$$\text{erank} = \exp(H(\boldsymbol{\mu})), \;\; H(\boldsymbol{\mu}) = -\sum_{j=1}^{r} \mu_j \log(\mu_j). \tag{5}$$

This effective rank is a real number between 1 and r, upper bounded by $\text{rank}(\boldsymbol{K})$, which measures the 'uniformity' of the spectrum through the entropy. We also track the various **trace ratios**

$$T_k = \sum_{j<k} \lambda_j / \sum_j \lambda_j, \tag{6}$$

which quantify the relative importance of the top k eigenvalues.

We note (Fig. 4.2, third row) a drop of the effective rank early in training (e.g. to less than 10% of its initial value in our experiments with no random labels; less than 20% when half of the labels are randomized). This can also be observed from the highlighted (in red) trace ratios T_{40}, T_{80} and T_{160} (Fig. 4.2, fourth row), e.g. the first top 40 eigenvalues (T_{40}), over 1000 in total, accounting for more than 70% of the total trace.

Remarkably, in the presence of high label noise, the effective rank of the tangent kernel (and hence that of the metric) evaluated on *training* examples (anti)-correlates nicely with the *test* accuracy: while decreasing and remaining relatively low during the learning phase (increase of test accuracy), it begins to rise again when overfitting starts (decrease of test

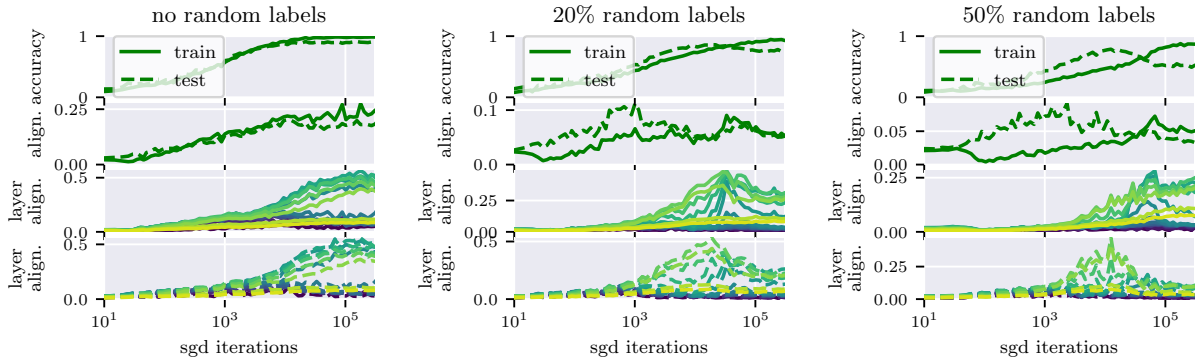


Fig. 4.3. Evolution of the (tangent) **feature alignment with class labels** as measured by CKA (7), during training of a VGG19 on CIFAR10 (same setup as in Fig. 4.2). Tangent kernels and label vectors are evaluated on batches of size 100 from both the training set (solid lines) and the test set (dashed lines). The plots in the last two rows show the alignment of tangent features associated to *each layer*. Layers are mapped to colors sequentially from input layer (-), through intermediate layers (-), to output layer (-). See Fig. B.5 and B.7 in Appendix B.3 for additional architectures and datasets.

accuracy). This suggests that this effective rank already provides a good proxy for the effective capacity of the network.

3.3. Alignment to class labels

We now include the evolution of the eigenvectors in our study. We investigate the similarity of the learned tangent features with the class label through centered kernel alignment. Given two kernel matrices $\boldsymbol{K}$ and $\boldsymbol{K}'$ in $\mathbb{R}^{r\times r}$, it is defined as (Cortes et al., 2012)

$$\mathrm{CKA}(\boldsymbol{K}, \boldsymbol{K}') = \frac{\mathrm{Tr}[\boldsymbol{K}_c \boldsymbol{K}'_c]}{\|\boldsymbol{K}_c\|_F \|\boldsymbol{K}'_c\|_F} \in [0,1] \tag{7}$$

where the subscript c denotes the feature centering operation, i.e. $\boldsymbol{K}_c = C\boldsymbol{K}C$ where $C = I_r - \frac{1}{r}\mathbf{1}\mathbf{1}^T$ is the centering matrix, and $\|\cdot\|_F$ is the Froebenius norm. CKA is a normalized version of the Hilbert-Schmidt Independence Criterion (Gretton et al., 2005) designed as a dependence measure for two sets of features. The normalization makes CKA invariant under isotropic rescaling.

Let $\boldsymbol{Y} \in \mathbb{R}^{nc}$ be the vector resulting from the concatenation of the one-hot label representations $\boldsymbol{Y}_i \in \mathbb{R}^c$ of the n samples. Similarity with the labels is measured through CKA with the rank-one kernel $\boldsymbol{K}_{\boldsymbol{Y}} := \boldsymbol{Y}\boldsymbol{Y}^\top$. Intuitively, $\mathrm{CKA}(\boldsymbol{K}, \boldsymbol{K}_{\boldsymbol{Y}})$ is high when $\boldsymbol{K}$ has low (effective) rank and such that the angle between $\boldsymbol{Y}$ and its top eigenspaces is small.[5] Maximizing such an index has been used as a criterion for kernel selection in the literature on learning kernels (Cortes et al., 2012).

[5]In the limiting case $\mathrm{CKA}(\boldsymbol{K}, \boldsymbol{K}_{\boldsymbol{Y}}) = 1$, the features are all aligned with each other and parallel to $\boldsymbol{Y}$.

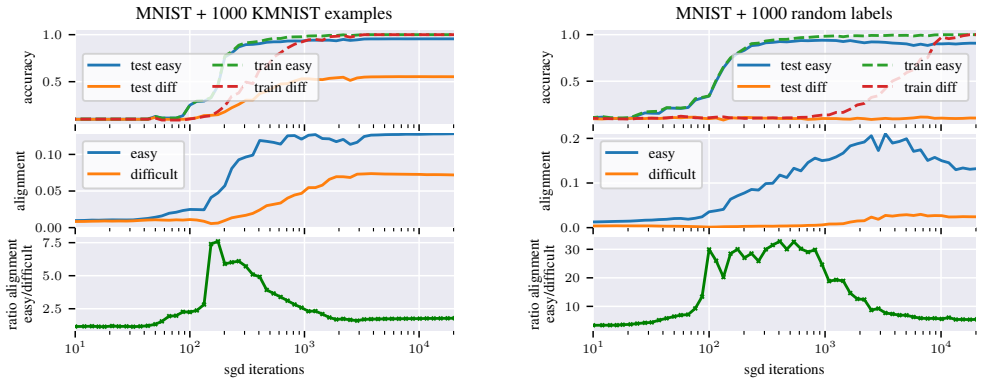


Fig. 4.4. Alignment *easy* versus *difficult*: We augment a dataset composed of 10.000 *easy* MNIST examples with 1000 *difficult* examples from 2 different setups: **(left)** 1000 MNIST examples with random label **(right)** 1000 KMNIST examples. We train a MLP with 6 layers of 80 hidden units using SGD with learning rate=0.02, momentum=0.9 and batch size=100. We observe that the alignment to (train) labels increases faster and to a higher value for the easy examples.

With the same setup as in Section 3.2, we observe (Fig. 4.3, 2nd row) an increasingly high CKA between the tangent kernel and the labels as training progresses. The trend is similar for other architectures and datasets (e.g., Fig. B.5 in Appendix B.3 shows CKA plots for MLP on MNIST and Resnets 18 on CIFAR10).

Interestingly, in the presence of high level noise, the CKA reaches a much higher value during the learning phase (increase of test accuracy) for tangent kernels and labels evaluated for *test* than for *train* inputs (note test labels are not randomized). Together with Equ. 4, this suggests a stronger learning bias towards features predictive of the *clean* labels. This is line with empirical observations that, in the presence of noise, deep networks 'learn patterns faster than noise' (Arpit et al., 2017) (see Section 3.4 below for additional insights).

We also report the alignments of the *layer-wise* tangent kernels. By construction, the tangent kernel, obtained by pairing features $\Phi_{w_p}(\mathbf{x})\Phi_{w_p}(\tilde{\mathbf{x}})$ and summing over all parameters w_p of the network, can also be expressed as the sum of layer-wise tangent kernels, $\boldsymbol{K}_{\mathbf{w}} = \sum_{\ell=1}^{L} \boldsymbol{K}_{\mathbf{w}}^{\ell}$, where $\boldsymbol{K}_{\mathbf{w}}^{\ell}$ results from summing only over parameters of the layer ℓ. We observe a high CKA, reaching more than 0.5 for a number of intermediate layers.[6] In the presence of high label noise, we note that CKAs tend to peak when the test accuracy does.

3.4. Hierarchical Alignment

A key aspect of the generalization question concerns the articulation between learning and memorization, in the presence of noise (Zhang et al., 2017a) or difficult examples (e.g.,

[6]We were expecting to see a gradually increasing CKA with ℓ; we do not have any intuitive explanation for the relatively low alignment observed for the very top layers.

Sagawa et al., 2020b). Motivated by this, we would like to probe the evolution of the tangent features *separately* in the directions of both types of examples in such settings. To do so, our strategy is to measure CKA for tangent kernels and label vectors evaluated on examples from two subsets of the same size in the training dataset – one with 'easy' examples, the other with 'difficult' ones. Our setup is to augment 10.000 MNIST training examples with 1000 difficult examples of 2 types: (*i*) examples with random labels and (*ii*) examples from the dataset KMNIST (Clanuwat et al., 2018). KMNIST images present features similar to MNIST digits (grayscale handwritten characters) but represent Japanese characters.

The results are shown in Fig. 4.4. As training progresses, we observe that the CKA on the easy examples increases faster (and to a higher value) than that on the difficult ones; in the case of the (structured) difficult examples from KMNIST, we also note an increase of the CKA later in training. This demonstrates a hierarchy in the adaptation of the kernel, measured by the ratio between both alignments. From the intuition developed in the paper (see spectral bias in Equ.(4)), we interpret this aspect of the non-linear dynamics as favoring a sequentialization of learning across patterns of different complexity ('easy patterns first'), a phenomenon analogous to one pointed out in the context of deep linear networks (Saxe et al., 2014; Lampinen et al., 2018; Gidel et al., 2019).

3.5. Ablation

Effect of depth. In order to study the influence of depth on alignment and test the robustness to the choice of seeds, we reproduce the experiment of the previous section for MLP with different depths, while varying parameter initialization and minibatch sampling. Our results, shown in Fig B.7 (Appendix B.3), suggest that the alignment effect is magnified as depth increases. We also observe that the ratio of the maximum alignment between easy and difficult examples is increased with depth, but stays high for a smaller number of iterations.

Effect of the learning rate. We observed in our experiments that increasing the learning rate tend to enhance alignment effects.[7] As an illustration, we reproduce in Fig. B.8 the same plots as in Fig. 4.2, for a learning rate reduced to 0.003. We observe a similar drop of the effective rank as in Fig. 4.2 at the beginning of training, but to a much (about 3 times) higher value.

[7]Note that for wide enough networks and small enough learning rate, we expect to recover the linear regime where the tangent features are constant during training (Jacot et al., 2018; Du et al., 2019b; Allen-Zhu et al., 2019).

4. Measuring Complexity

In this section, drawing upon intuitions from linear models, we illustrate in a simple setting how the alignment of tangent features can act as implicit regularization. By extrapolating Rademacher complexity bounds for linear models, we also motivate a new complexity measure for neural networks and compare its correlation to generalization against various measures proposed in the literature. We refer to Appendix B.2 for a review of classical results, further technical details, and proofs.

4.1. Insights from Linear Models

4.1.1. Setup. We restrict here to scalar functions $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$ linearly parametrized by $\mathbf{w} \in \mathbb{R}^P$. Such a function class defines a *constant* (tangent) kernel and geometry, as defined in Section 2. Given n input samples, the n features $\Phi(\mathbf{x}_i) \in \mathbb{R}^P$ yield an $n \times P$ feature matrix $\mathbf{\Phi}$.

Our discussion will be based on the (empirical) **Rademacher complexity**, which shows up in generalization bounds (Bartlett & Mendelson, 2002); see Appendix B.2.2 for a review. It measures how well $\mathcal{F}$ correlates with random noise on the sample set $\mathcal{S}$:

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\sigma} \in \{\pm 1\}^n} \left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(\mathbf{x}_i) \right]. \tag{8}$$

The Rademacher complexity depends on the size (or **capacity**) of the class $\mathcal{F}$. Constraints on the capacity, such as those induced by the implicit bias of the training algorithm, can reduce the Rademacher complexity and lead to sharper generalization bounds.

A standard approach for controlling capacity is in terms of the *norm* of the weight vector – usually the ℓ_2-norm. In general, given any invertible matrix $A \in \mathbb{R}^{P \times P}$, we may consider the norm $\|\mathbf{w}\|_A := \sqrt{\mathbf{w}^\top g_A \mathbf{w}}$ induced by the metric $g_A = AA^\top$. Consider the (sub)classes of functions induced by balls of given radius:

$$\mathcal{F}^A_{M_A} = \{ f_{\mathbf{w}} \colon \mathbf{x} \mapsto \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle \mid \|\mathbf{w}\|_A \leq M_A \}. \tag{9}$$

A direct extension of standard bounds for the Rademacher complexity (see Appendix B.2.3) yields,

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}^A_{M_A}) \leq (M_A/n) \|A^{-1}\mathbf{\Phi}^\top\|_{\mathrm{F}} \tag{10}$$

where $\|A^{-1}\mathbf{\Phi}^\top\|_{\mathrm{F}}$ is the Froebenius norm of the *rescaled* feature matrix.[8]

This freedom in the choice of rescaling matrix A raises the question of which of the norms $\|\cdot\|_A$ provide meaningful measures of the model's capacity. Recent works (Belkin et al.,

[8] We also have $\|A^{-1}\mathbf{\Phi}^\top\|_{\mathrm{F}} = \sqrt{\mathrm{Tr}\boldsymbol{K}_A}$ in terms of the (rescaled) kernel matrix $\boldsymbol{K}_A = \mathbf{\Phi} g_A^{-1} \mathbf{\Phi}^\top$.

2018; Muthukumar et al., 2020) pointed out that using ℓ_2 norm is not coherently linked with generalization in practice. We discuss this issue in Appendix B.2.5, illustrating how meaningful norms critically depend on the geometry defined by the features.

4.1.2. Feature Alignment as Implicit Regularization. Here we describe a simple procedure making the geometry *adaptive* along optimization paths. The goal is to illustrate in a simple setting how feature alignment can impact complexity and generalization, in a way that mimics the behaviour of a non-linear dynamics. The idea is to *learn* a rescaling metric at each iteration of our algorithm, using a local version of the bounds (10).

Complexity of Learning Flows. Since we are interested in functions $f_{\mathbf{w}}$ that result from an iterative algorithm, we consider functions $f_{\mathbf{w}} = \sum_t \delta f_{\mathbf{w}_t}$ written in terms of a sequence of updates[9] $\delta f_{\mathbf{w}_t}(\mathbf{x}) = \langle \delta \mathbf{w}_t, \Phi(\mathbf{x}) \rangle$ (we set $f_0 = 0$ to keep the notation simple), with *local* constraints on the parameter updates:

$$\mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{A}} = \{ f_{\mathbf{w}} \colon \mathbf{x} \mapsto \textstyle\sum_t \langle \delta \mathbf{w}_t, \Phi(\mathbf{x}) \rangle \mid \|\delta \mathbf{w}_t\|_{A_t} \leq m_t \} \tag{11}$$

The result (10) extends as follows.

Theorem 4.2 (Complexity of Learning Flows)**.** *Given any sequences* $\boldsymbol{A}$ *and* $\boldsymbol{m}$ *of invertible matrices* $A_t \in \mathbb{R}^{P \times P}$ *and positive numbers* $m_t > 0$*, we have the bound*

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{A}}) \leq \textstyle\sum_t (m_t/n) \|A_t^{-1} \boldsymbol{\Phi}^\top\|_{\mathrm{F}}. \tag{12}$$

Note that, by linear reparametrization invariance $\mathbf{w} \mapsto A^\top \mathbf{w}$, $\Phi \mapsto A^{-1}\Phi$, the *same* result can be formulated in terms of the sequence $\boldsymbol{\Phi} = \{\Phi_t\}_t$ of feature maps $\Phi_t = A_t^{-1}\Phi$. The function class (11) can equivalently be written as

$$\mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{\Phi}} = \{ f_{\mathbf{w}} \colon \mathbf{x} \mapsto \textstyle\sum_t \langle \tilde{\delta} \mathbf{w}_t, \Phi_t(\mathbf{x}) \rangle \mid \|\tilde{\delta} \mathbf{w}_t\|_2 \leq m_t \} \tag{13}$$

In this formulation, the result (12) reads:

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{\Phi}}) \leq \textstyle\sum_t (m_t/n) \|\boldsymbol{\Phi}_t\|_{\mathrm{F}}. \tag{14}$$

Optimizing the Feature Scaling. To obtain learning flows with lower complexity, Thm. 4.2 suggests modification of the algorithm to include, at each iteration t, a reparametrization step with a suitable matrix $\tilde{A}_t$ giving a low contribution to the bound (12). Applied to gradient descent (GD), this leads to a new update rule sketched in Fig. 4.5 (left), which we call **SuperNat**[10], where optimization in Step 2 is over a given class of reparametrization

[9] In order to not assume a specific upper bound on the number of iterations, we can think of the updates from an iterative algorithm as an infinite sequence $\{\delta \mathbf{w}_0, \cdots \delta \mathbf{w}_t, \cdots\}$ such that for some T, $\delta \mathbf{w}_t = 0$ for all $t > T$.
[10] This was meant to suggest that our gradient step is not merely natural, but supernatural.

SuperNat update ($\tilde{A}_0 = \boldsymbol{I}$, $\Phi_0 = \Phi$, $\boldsymbol{K}_0 = \boldsymbol{K}$):

(1) Perform gradient step $\widetilde{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t + \delta\mathbf{w}_{\mathrm{GD}}$
(2) Find minimizer $\tilde{A}_{t+1}$ of $\|\delta\mathbf{w}_{\mathrm{GD}}\|_{\tilde{A}}\|\tilde{A}^{-1}\boldsymbol{\Phi}_t^\top\|_{\mathrm{F}}$
(3) Reparametrize:

$$\mathbf{w}_{t+1} \leftarrow \tilde{A}_{t+1}^\top \widetilde{\mathbf{w}}_{t+1}, \Phi_{t+1} \leftarrow \tilde{A}_{t+1}^{-1}\Phi_t$$

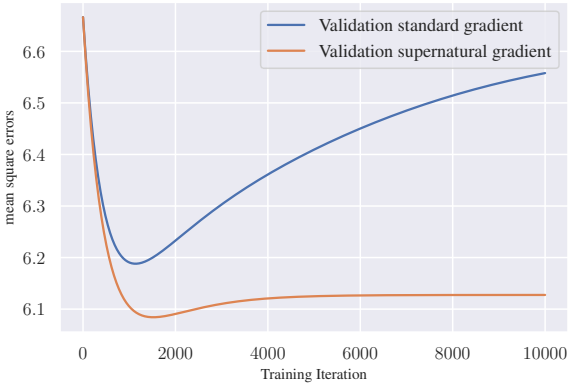


Fig. 4.5. (left) SuperNat algorithm and **(right)** validation curves obtained with standard and **SuperNat** gradient descent, on the noisy linear regression problem. At each iteration, **SuperNat** identifies dominant features and stretches the kernel along them, thereby slowing down and eventually freezing the learning dynamics in the noise direction. This naturally yields better generalization than standard gradient descent on this problem.

matrices. The successive reparametrizations yield a varying feature map $\Phi_t = A_t^{-1}\Phi$ where $A_t = \tilde{A}_0 \cdots \tilde{A}_t$.[11]

In the original representation Φ, **SuperNat** amounts to natural gradient descent (Amari, 1998) with respect to the local metric $g_{A_t} = A_t A_t^\top$. By construction, we also have $\delta f_{\mathbf{w}_t}(\mathbf{x}) = \langle \delta\mathbf{w}_{\mathrm{GD}}, \Phi_t(\mathbf{x})\rangle$ where $\delta\mathbf{w}_{\mathrm{GD}}$ are standard gradient descent updates in the linear model with feature map Φ_t.

As an example, let $\boldsymbol{\Phi} = \sum_{j=1}^n \sqrt{\lambda_j}\boldsymbol{u}_j\boldsymbol{v}_j^\top$ be the SVD of the feature matrix. We restrict to the class of matrices

$$\tilde{A}_{\boldsymbol{\nu}} = \sum_{j=1}^n \sqrt{\nu_j}\boldsymbol{v}_j\boldsymbol{v}_j^\top + \mathrm{Id}_{\mathrm{span}\{\boldsymbol{v}\}^\perp} \tag{15}$$

labelled by weights $\nu_j > 0, j = 1, \cdots, n$. With such a class, the action $\boldsymbol{\Phi}_t^\top \to A_{\boldsymbol{\nu}}^{-1}\boldsymbol{\Phi}_t^\top$ merely rescales the singular values $\lambda_{jt} \to \lambda_{jt}/\nu_j$, leaving the singular vectors unchanged. We work with gradient descent w.r.t a cost function L, so that $\delta\mathbf{w}_{\mathrm{GD}} = -\eta\nabla_{\mathbf{w}}L$.

Proposition 4.3. *Any minimizer in Step 2 of* **SuperNat** *over matrices $\tilde{A}_{\boldsymbol{\nu}}$ in the class (15), takes the form*

$$\nu_{jt}^* = \kappa\frac{1}{|\boldsymbol{u}_j^\top\nabla_{\mathbf{f_w}}L|} \tag{16}$$

where $\nabla_{\mathbf{f_w}}$ denotes the gradient w.r.t the sample outputs $f_{\mathbf{w}} := [f_{\mathbf{w}}(\mathbf{x}_1), \cdots f_{\mathbf{w}}(\mathbf{x}_n)]^\top$, for some constant $\kappa > 0$.

[11]Note that upon training a non-linear model, the updates of the tangent features take the same form $\Phi_t = \tilde{A}_t^{-1}\Phi_{t-1}$ as in Step 3 of **SuperNat**, the difference being that $\tilde{A}_t$ is now a differential operator, e.g. at first order $\tilde{A}_t = \mathrm{Id} - \delta\mathbf{w}_t^\top\frac{\partial}{\partial\mathbf{w}_t}$.

In this context, this yields the following update rule, up to isotropic rescaling, for the singular values of $\mathbf{\Phi}_t$:

$$\lambda_{j(t+1)} = |\boldsymbol{u}_j^\top \nabla_{\mathbf{f_w}} L| \lambda_{jt}. \tag{17}$$

In this illustrative setting, we see how the feature map (or kernel) adapts to the task, by stretching (resp. contracting) its geometry in directions $\boldsymbol{u}_j$ along which the residual $\nabla_{\mathbf{f_w}} L$ has large (resp. small) components. Intuitively, if a large component $|\boldsymbol{u}_j^\top \nabla_{\mathbf{f_w}} L|$ corresponds to signal and a small one $|\boldsymbol{u}_k^\top \nabla_{\mathbf{f_w}} L|$ corresponds to noise, then the ratio $\lambda_{jt}/\lambda_{kt}$ of singular values gets rescaled by the signal-to-noise ratio, thereby increasing the alignment of the learned features to the signal.

As a proof of concept, we consider the following regression setup. We consider a linear model with Gaussian features $\Phi = [\varphi, \varphi_{\text{noise}}] \in \mathbb{R}^{d+1}$ where $\varphi \sim \mathcal{N}(0,1)$ and $\varphi_{\text{noise}} \sim \mathcal{N}(0, \frac{1}{d} I_d)$. Given n input samples, the n features $\Phi(\mathbf{x}_i)$ yield $\boldsymbol{\varphi} \in \mathbb{R}^n$ and $\boldsymbol{\varphi}_{\text{noise}} \in \mathbb{R}^{n \times d}$. We assume the label vector takes the form $\boldsymbol{y} = \boldsymbol{\varphi} + P_{\text{noise}}(\boldsymbol{\epsilon})$, where Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I_n)$ is projected onto the noise features through $P_{\text{noise}} = \boldsymbol{\varphi}_{\text{noise}} \boldsymbol{\varphi}_{\text{noise}}^\top$. The model is trained by gradient descent of the mean squared loss and its **SuperNat** variant, where Step 2 uses the analytical solution of Proposition 4.3. We set $d = 10, \sigma^2 = 0.1$ and use $n = 50$ training points.

Fig 4.5 (right) shows test error obtained with standard and **SuperNat** gradient descent on this problem. At each iteration, **SuperNat** identifies dominant features (feature selection, here φ) and stretches the metric along them, thereby slowing down and eventually freezing the dynamics in the orthogonal (noise) directions (compression). The working hypothesis in this paper, supported by the observations of Section 3, is that for neural networks, such a (tangent) feature alignment is dynamically induced as an effect of non-linearity.

4.2. A New Complexity Measure for Neural Networks

Equ. (14) provides a bound of the Rademacher complexity for the function classes (11) specified by a *fixed* sequence of feature maps (see Appendix B.2.4 for a generalization to the multiclass setting). By extrapolation to the case of non-deterministic sequences of feature maps, we propose using

$$\mathcal{C}(f_{\mathbf{w}}) = \sum_t \|\delta \mathbf{w}_t\|_2 \|\mathbf{\Phi}_t\|_{\text{F}} \tag{18}$$

as a heuristic measure of complexity for neural networks, where $\mathbf{\Phi}_t$ is the learned tangent feature matrix[12] at training iteration t, and $\|\delta \mathbf{w}_t\|_2$ is the norm of the SGD update. Following a standard protocol for studying complexity measures, (e.g., Neyshabur et al., 2017a), Fig. 4.6 shows its behaviour for MLP on MNIST and VGG19 on CIFAR10 trained with cross entropy loss, with **(left)** fixed architecture and varying level of corruption in the labels and **(right)**

[12] In terms of tangent kernels, $\|\mathbf{\Phi}_t\|_{\text{F}} = \sqrt{\text{Tr} \boldsymbol{K}_t}$ where $\boldsymbol{K}_t$ is the tangent kernel (Gram) matrix.

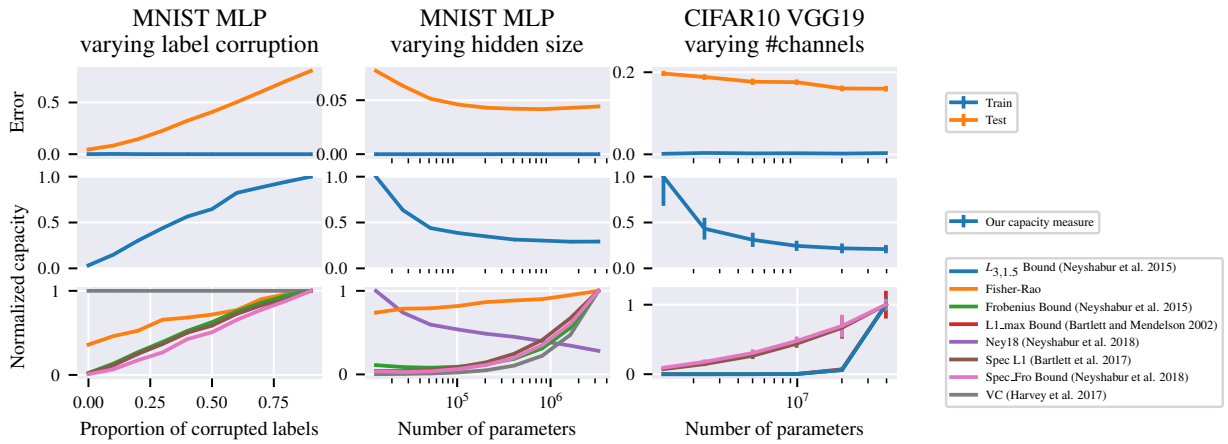


Fig. 4.6. Complexity measures on MNIST with a 1 hidden layer MLP **(left)** as we increase the hidden layer size, **(center)** for a fixed hidden layer of 256 units as we increase label corruption and **(right)** for a VGG19 on CIFAR10 as we vary the number of channels. All networks are trained until cross-entropy reaches 0.01. Our proposed complexity measure and the one by Neyshabur et al. 2018 are the only ones to correctly reflect the shape of the generalization gap in these settings.

varying hidden layer size/number of channels up to 4 millions parameters, against other capacity measures proposed in the recent literature. We observe that it correctly reflects the shape of the generalization gap.

5. Related Work

Role of Feature Geometry in Linear Models. Analysis of the relation between capacity and feature geometry can be traced back to early work on kernel methods (Schölkopf et al., 1999a), which lead to data-dependent error bounds in terms of the eigenvalues of the kernel Gram matrix (Schölkopf et al., 1999b).

Recently, new analysis of minimum norm interpolators and max margin solutions for overparametrized linear models emphasize the key role of feature geometry, and specifically feature anisotropy, in the generalization performance (Bartlett et al., 2019b; Muthukumar et al., 2019, 2020; Xie et al., 2020). Feature anisotropy combined to a high predictive power of the dominant features is the condition for a high centered alignment between kernel and class labels. In the context of neural networks, our results highlight the role of the non linear training dynamics in favouring such conditions.

Generalization Measures. There has been a large body of work on complexity/generalization measures for neural networks (see, Jiang et al., 2020, and references therein), some of which theoretically motivated by norm or margin based bounds (e.g., Neyshabur et al., 2019; Bartlett et al., 2017). Liang et al. (2019) proposed using the Fisher-Rao norm of the solution as a geometrically invariant complexity measure. By contrast,

our approach to measuring complexity takes into account the geometry along the whole optimization trajectories. Since the geometry we consider is defined through the gradient second moments, our perspectice is closely related to the notions of stiffness (Fort et al., 2019) and coherent gradients (Chatterjee, 2020).

Dynamics of Tangent Kernels. Several recent works investigated the 'feature learning' regime where neural tangent kernels evolve during training (Geiger et al., 2020a; Woodworth et al., 2020). Independent concurrent works highlight alignment and compression phenomena similar to the one we study here (Kopitkov & Indelman, 2020; Paccolat et al., 2021). We offer various complementary empirical insights, and frame the alignment mechanism from the point of view of implicit regularization.

6. Conclusion

Through experiments with modern architectures, we highlighted an effect of dynamical alignment of the neural tangent features and their kernel along a small number of task-dependent directions during training, reflected by an early drop of the effective rank and an increasing similarity with the class labels, as measured by centered kernel alignment. We interpret this effect as a combined mechanism of feature selection and model compression around dominant features.

Drawing upon intuitions from linear models, we argued that such a dynamical alignment acts as implicit regularizer. By extrapolating a new analysis of Rademacher complexity bounds for linear models, we also proposed a complexity measure that captures this phenomenon, and showed that it correlates with the generalization gap when varying the number of parameters, and when increasing the proportion of corrupted labels.

The results of this paper open several avenues for further investigation. The type of complexity measure we propose suggests new principled ways to design algorithms that learn the geometry in which to perform gradient descent (Srebro et al., 2011; Neyshabur et al., 2017b). Whether a procedure such as **SuperNat** can produce meaningful practical results for neural networks remains to be seen.

One of the consequences one can expect from the alignment effects highlighted here is to bias learning towards explaining most of the data with a small number of highly predictive features. While this feature selection ability might explain in part the performance of neural networks on a range of supervised tasks, it may also make them brittle under spurious correlation (e.g., Sagawa et al., 2020b) and underpin their notorious weakness to generalize out-of-distribution (e.g., Geirhos et al., 2020). Resolving this tension is an important challenge towards building more robust models.

Chapter 5

Third Article: Mutual Information Neural Estimation

Prologue

Article Details. **Mutual Information Neural Estimation.** Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, R Devon Hjelm. This paper was pusblished at ICML 2019.

Personal contributions. This project was conceived and lead by Ishamel Belghazi. I was in charge of the theoretical aspects of the paper and contributed to the general writing together with Ishmael and Devon, with assistance from all the other co-authors.

Discussion and Recent Developments. As we have seen in Chapter 2, many statistical divergences, such as f-divergences (including the KL-divergence) and integral probability metrics, have a variational formulation as the maximum of some optimization problem over sufficiently large classes of functions $\mathcal{F}$. The idea of *neural estimation* is to parametrize $\mathcal{F}$ using a neural network and optimize over the parameter space. This work introduces and study neural estimators of mutual information.

Since the publication of the paper, several works dwelt further on the problem of estimating mutual information from finite data. Notably, McAllester & Stratos (2020) describes a limitation of lower-bound estimators: they show that any distribution-free high confidence lower-bound requires a sample size that is exponential in the value of the bound. This suggests that empirical estimation can be unfeasible in practice when the mutual information is large. Since this results from a worse-case analysis on the distributions at play, it remains to be seen whether such limitations can be avoided upon suitable assumptions that would hold for realistic data. Other works highlighted bad bias-variance tradeoffs of variational estimators, and proposed ways to mitigate the problem (Poole et al., 2019; Song & Ermon,

2020). In practice, depending on the applications, some variational estimators are more stable and produce better results than others (see e.g., Hjelm et al., 2019).

The question of the consistency of neural estimation splits into an approximation problem related to the finite size of the network, and an empirical estimation problem related to the use of finite samples. We prove consistency of one of our estimators below, using standard techniques; we also give non-asymptotic sample complexity bounds for the estimation of the parametrized form of mutual information. A few works derived analogous bounds in the context of GANs (Arora et al., 2017; Zhang et al., 2018b). In a very recent work, Sreekumar & Goldfeld (2021) tackle the (much harder) problem of quantifying the approximation error and studying rigorously approximation-estimation tradeoffs.

Note that, as it has probably long been recognized among researchers working on GANs, as argued in Tschannen et al. (2020) (see also Huang et al. (2017)), the empirical success of neural divergences and mutual estimation estimators might be due less to properties of mutual information *per se* than to the specific parametrization of the estimator. A further understanding of how the choice of statistic network, and the inductive bias that it introduces, impact the properties of the estimator, is an interesting direction for future research.

Considering the ubiquitous role of mutual information in a number of applications, the paper had numerous follow-ups. A landmark example is an adaptation of the infomax principle (Linsker, 1988), Deep InfoMax (Hjelm et al., 2019), which performs unsupervised representation learning by maximizing an estimate of the mutual information between the inputs and their representation, or between different views of the data (Bachman et al., 2019; Tian et al., 2020).

1. Introduction

Mutual information is a fundamental quantity for measuring the relationship between random variables. In data science it has found applications in a wide range of domains and tasks, including biomedical sciences (Maes et al., 1997), blind source separation (BSS, e.g., independent component analysis, Hyvärinen et al., 2004), information bottleneck (IB, Tishby et al., 2000), feature selection (Kwak & Choi, 2002; Peng et al., 2005), and causality (Butte & Kohane, 2000).

Put simply, mutual information quantifies the dependence of two random variables X and Z. It has the form,

$$I(X;Z) = \int_{\mathcal{X}\times\mathcal{Z}} \log \frac{d\mathbb{P}_{XZ}}{d\mathbb{P}_X \otimes \mathbb{P}_Z} d\mathbb{P}_{XZ}, \tag{1}$$

where $\mathbb{P}_{XZ}$ is the joint probability distribution, and $\mathbb{P}_X = \int_{\mathcal{Z}} d\mathbb{P}_{XZ}$ and $\mathbb{P}_Z = \int_{\mathcal{X}} d\mathbb{P}_{XZ}$ are the marginals. In contrast to correlation, mutual information captures non-linear statistical dependencies between variables, and thus can act as a measure of true dependence (Kinney & Atwal, 2014).

Despite being a pivotal quantity across data science, mutual information has historically been difficult to compute (Paninski, 2003). Exact computation is only tractable for discrete variables (as the sum can be computed exactly), or for a limited family of problems where the probability distributions are known. For more general problems, this is not possible. Common approaches are non-parametric (e.g., binning, likelihood-ratio estimators based on support vector machines, non-parametric kernel-density estimators; see, Fraser & Swinney, 1986; Darbellay & Vajda, 1999; Suzuki et al., 2008; Kwak & Choi, 2002; Moon et al., 1995; Kraskov et al., 2004), or rely on approximate gaussianity of data distribution (e.g., Edgeworth expansion, Van Hulle, 2005). Unfortunately, these estimators typically do not scale well with sample size or dimension (Gao et al., 2014), and thus cannot be said to be general-purpose. Other recent works include Kandasamy et al. (2017); Singh & Póczos (2016); Moon et al. (2017).

In order to achieve a general-purpose estimator, we rely on the well-known characterization of the mutual information as the Kullback-Leibler (KL-) divergence (Kullback, 1997) between the joint distribution and the product of the marginals (i.e., $I(X;Z) = D_{KL}(\mathbb{P}_{XZ} \,||\, \mathbb{P}_X \otimes \mathbb{P}_Z)$). Recent work uses a dual formulation to cast the estimation of f-divergences (including the KL-divergence, see Nguyen et al., 2010) as part of an adversarial game between competing deep neural networks (Nowozin et al., 2016). This approach is at the cornerstone of generative adversarial networks (GANs, Goodfellow et al., 2014a), which train a generative model without any explicit assumptions about the underlying distribution of the data.

In this paper we demonstrate that exploiting dual optimization to estimate divergences goes beyond the minimax objective as formalized in GANs. We leverage this strategy to offer a general-purpose parametric neural estimator of mutual information based on dual representations of the KL-divergence (Ruderman et al., 2012), which we show is valuable in settings that do not necessarily involve an adversarial game. Our estimator is scalable, flexible, and completely trainable via back-propagation. The contributions of this paper are as follows:

- We introduce the Mutual Information Neural Estimator (MINE), which is scalable, flexible, and completely trainable via back-prop, as well as provide a thorough theoretical analysis.

- We show that the utility of this estimator transcends the minimax objective as formalized in GANs, such that it can be used in mutual information estimation, maximization, and minimization.
- We apply MINE to palliate mode-dropping in GANs and to improve reconstructions and inference in Adversarially Learned Inference (ALI, Dumoulin et al., 2016) on large scale datasets.
- We use MINE to apply the Information Bottleneck method Tishby et al. (2000) in a continuous setting, and show that this approach outperforms variational bottleneck methods (Alemi et al., 2016).

2. Background

2.1. Mutual Information

Mutual information is a Shannon entropy-based measure of dependence between random variables. The mutual information between X and Z can be understood as the decrease of the uncertainty in X given Z:

$$I(X;Z) := H(X) - H(X \mid Z), \tag{2}$$

where H is the Shannon entropy, and $H(X \mid Z)$ is the conditional entropy of Z given X. As stated in Eqn. 1 and the discussion above, the mutual information is equivalent to the Kullback-Leibler (KL-) divergence between the joint, $\mathbb{P}_{XZ}$, and the product of the marginals $\mathbb{P}_X \otimes \mathbb{P}_Z$:

$$I(X,Z) = D_{KL}(\mathbb{P}_{XZ} \mid\mid \mathbb{P}_X \otimes \mathbb{P}_Z), \tag{3}$$

where D_{KL} is defined as[1],

$$D_{KL}(\mathbb{P} \mid\mid \mathbb{Q}) := \mathbb{E}_{\mathbb{P}} \left[\log \frac{d\mathbb{P}}{d\mathbb{Q}} \right]. \tag{4}$$

whenever $\mathbb{P}$ is absolutely continuous with respect to $\mathbb{Q}$[2].

The intuitive meaning of Eqn. 3 is clear: the larger the divergence between the joint and the product of the marginals, the stronger the dependence between X and Z. This divergence, hence the mutual information, vanishes for fully independent variables.

[1] Although the discussion is more general, we can think of $\mathbb{P}$ and $\mathbb{Q}$ as being distributions on some compact domain $\Omega \subset \mathbb{R}^d$, with density p and q respect the Lebesgue measure λ, so that $D_{KL} = \int p \log \frac{p}{q} d\lambda$.

[2] and infinity otherwise.

2.2. Dual representations of the KL-divergence.

A key technical ingredient of MINE are *dual representations* of the KL-divergence. We will primarily work with the Donsker-Varadhan representation (Donsker & Varadhan, 1983), which results in a tighter estimator; but will also consider the dual f-divergence representation (Keziou, 2003; Nguyen et al., 2010; Nowozin et al., 2016).

The Donsker-Varadhan representation. The following theorem gives a representation of the KL-divergence (Donsker & Varadhan, 1983):

Theorem 5.1 (Donsker-Varadhan representation)**.** *The KL divergence admits the following dual representation:*

$$D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T]), \tag{5}$$

where the supremum is taken over all functions T such that the two expectations are finite.

Proof. See the Supplementary Material.

A straightforward consequence of Theorem 5.1 is as follows. Let $\mathcal{F}$ be *any* class of functions $T : \Omega \to \mathbb{R}$ satisfying the integrability constraints of the theorem. We then have the lower-bound[3]:

$$D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) \geq \sup_{T\in\mathcal{F}} \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T]). \tag{6}$$

Note also that the bound is *tight* for optimal functions T^* that relate the distributions to the *Gibbs density* as,

$$d\mathbb{P} = \frac{1}{Z} e^{T^*} d\mathbb{Q}, \text{ where } Z = \mathbb{E}_{\mathbb{Q}}[e^{T^*}]. \tag{7}$$

The f-divergence representation. It is worthwhile to compare the Donsker-Varadhan representation to the f-divergence representation proposed in Nguyen et al. (2010); Nowozin et al. (2016), which leads to the following bound:

$$D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) \geq \sup_{T\in\mathcal{F}} \mathbb{E}_{\mathbb{P}}[T] - \mathbb{E}_{\mathbb{Q}}[e^{T-1}]. \tag{8}$$

Although the bounds in Eqns. 6 and 8 are tight for sufficiently large families $\mathcal{F}$, the Donsker-Varadhan bound is *stronger* in the sense that, for any fixed T, the right hand side of Eqn. 6 is larger[4] than the right hand side of Eqn. 8. We refer to the work by Ruderman et al. (2012) for a derivation of both representations in Eqns. 6 and 8 from the unifying perspective of Fenchel duality. In Section 3 we discuss versions of MINE based on these two representations, and numerical comparisons are performed in Section 4.

[3]The bound in Eqn. 6 is known as the *compression lemma* in the PAC-Bayes literature (Banerjee, 2006).
[4]To see this, just apply the identity $x \geq e \log x$ with $x = \mathbb{E}_{\mathbb{Q}}[e^T]$.

3. The Mutual Information Neural Estimator

In this section we formulate the framework of the Mutual Information Neural Estimator (MINE). We define MINE and present a theoretical analysis of its consistency and convergence properties.

3.1. Method

Using both Eqn. 3 for the mutual information and the dual representation of the KL-divergence, the idea is to choose $\mathcal{F}$ to be the family of functions $T_\theta : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ parametrized by a deep neural network with parameters $\theta \in \Theta$. We call this network the *statistics network*. We exploit the bound:

$$I(X;Z) \geq I_\Theta(X,Z), \tag{9}$$

where $I_\Theta(X,Z)$ is the *neural information measure* defined as

$$I_\Theta(X,Z) = \sup_{\theta \in \Theta} \mathbb{E}_{\mathbb{P}_{XZ}}[T_\theta] - \log(\mathbb{E}_{\mathbb{P}_X \otimes \mathbb{P}_Z}[e^{T_\theta}]). \tag{10}$$

The expectations in Eqn. 10 are estimated using empirical samples[5] from $\mathbb{P}_{XZ}$ and $\mathbb{P}_X \otimes \mathbb{P}_Z$ or by shuffling the samples from the joint distribution along the batch axis. The objective can be maximized by gradient ascent.

It should be noted that Eqn. 10 actually *defines* a new class information measures. The expressive power of neural networks insures that they can approximate the mutual information with arbitrary accuracy with large enough networks.

In what follows, given a distribution $\mathbb{P}$, we denote by $\hat{\mathbb{P}}^{(n)}$ as the empirical distribution associated to n *i.i.d.* samples.

Definition 5.1 (Mutual Information Neural Estimator (MINE))**.** *Let $\mathcal{F} = \{T_\theta\}_{\theta \in \Theta}$ be the set of functions parametrized by a neural network. MINE is defined as,*

$$\widehat{I(X;Z)}_n = \sup_{\theta \in \Theta} \mathbb{E}_{\mathbb{P}^{(n)}_{XZ}}[T_\theta] - \log(\mathbb{E}_{\mathbb{P}^{(n)}_X \otimes \hat{\mathbb{P}}^{(n)}_Z}[e^{T_\theta}]). \tag{11}$$

Details on the implementation of MINE are provided in Algorithm 1. An analogous definition and algorithm also hold for the f-divergence formulation in Eqn. 8, which we refer to as MINE-f. Since Eqn. 8 lower-bounds Eqn. 6, it generally leads to a *looser* estimator of the mutual information, and numerical comparisons of MINE with MINE-f can be found in Section 4. However, in a mini-batch setting, the SGD gradients of MINE are biased. We address this in the next section.

[5]Note that samples $\bar{x} \sim \mathbb{P}_X$ and $\bar{z} \sim \mathbb{P}_Z$ from the marginals are obtained by simply dropping x, z from samples $(\bar{x}, z)$ and $(x, \bar{z}) \sim \mathbb{P}_{XZ}$.

Algorithm 1 MINE

$\theta \leftarrow$ initialize network parameters
repeat
  Draw b minibatch samples from the joint distribution:
  $(\boldsymbol{x}^{(1)}, \boldsymbol{z}^{(1)}), \dots, (\boldsymbol{x}^{(b)}, \boldsymbol{z}^{(b)}) \sim \mathbb{P}_{XZ}$
  Draw b samples from the Z marginal distribution:
  $\bar{\boldsymbol{z}}^{(1)}, \dots, \bar{\boldsymbol{z}}^{(b)} \sim \mathbb{P}_Z$
  Evaluate the lower-bound:
$\mathcal{V}(\theta) \leftarrow \frac{1}{b}\sum_{i=1}^{b} T_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}) - \log(\frac{1}{b}\sum_{i=1}^{b} e^{T_\theta(\boldsymbol{x}^{(i)}, \bar{\boldsymbol{z}}^{(i)})})$
  Evaluate bias corrected gradients (e.g., moving average):
  $\widehat{G}(\theta) \leftarrow \widetilde{\nabla}_\theta \mathcal{V}(\theta)$
  Update the statistics network parameters:
  $\theta \leftarrow \theta + \widehat{G}(\theta)$
until convergence

3.2. Correcting the bias from the stochastic gradients

A naive application of stochastic gradient estimation leads to the gradient estimate:

$$\widehat{G}_B = \mathbb{E}_B[\nabla_\theta T_\theta] - \frac{\mathbb{E}_B[\nabla_\theta T_\theta \, e^{T_\theta}]}{\mathbb{E}_B\left[e^{T_\theta}\right]}. \tag{12}$$

where, in the second term, the expectations are over the samples of a minibatch B, leads to a biased estimate of the full batch gradient[6].

Fortunately, the bias can be reduced by replacing the estimate in the denominator by an exponential moving average. For small learning rates, this improved MINE gradient estimator can be made to have arbitrarily small bias. We found in our experiments that this improves all-around performance of MINE.

3.3. Theoretical properties

In this section we analyze the consistency and convergence properties of MINE. All the proofs can be found in the Supplementary Material.

3.3.1. Consistency. MINE relies on a choice of (i) a statistics network and (ii) n samples from the data distribution $\mathbb{P}_{XZ}$.

Definition 5.2 (Strong consistency)**.** *The estimator $\widehat{I(X;Z)}_n$ is strongly consistent if for all $\epsilon > 0$, there exists a positive integer N and a choice of statistics network such that:*

$$\forall n \geq N, \quad |I(X,Z) - \widehat{I(X;Z)}_n| \leq \epsilon, \; a.e.$$

[6]From the optimization point of view, the f-divergence formulation has the advantage of making the use of SGD with unbiased gradients straightforward.

where the probability is over a set of samples.

In a nutshell, the question of consistency is divided into two problems: an *approximation* problem related to the size of the family, $\mathcal{F}$, and an *estimation* problem related to the use of empirical measures. The first problem is addressed by universal approximation theorems for neural networks (Hornik, 1989). For the second problem, classical consistency theorems for extremum estimators apply (Van de Geer, 2000) under mild conditions on the parameter space.

This leads to the two lemmas below. The first lemma states that the neural information measures $I_\Theta(X,Z)$, defined in Eqn. 10, can approximate the mutual information with arbitrary accuracy:

Lemma 5.2 (approximation)**.** *Let $\epsilon > 0$. There exists a neural network parametrizing functions T_θ with parameters θ in some compact domain $\Theta \subset \mathbb{R}^k$, such that*

$$|I(X,Z) - I_\Theta(X,Z)| \leq \epsilon, \text{ a.e.}$$

The second lemma states the almost sure convergence of MINE to a neural information measure as the number of samples goes to infinity:

Lemma 5.3 (estimation)**.** *Let $\epsilon > 0$. Given a family of neural network functions T_θ with parameters θ in some bounded domain $\Theta \subset \mathbb{R}^k$, there exists an $N \in \mathbb{N}$, such that*

$$\forall n \geq N, \quad |\, \widehat{I(X;Z)}_n - I_\Theta(X,Z) \,| \leq \epsilon, \text{ a.e.} \tag{13}$$

Combining the two lemmas with the triangular inequality, we have,

Theorem 5.4. *MINE is strongly consistent.*

3.3.2. Sample complexity. In this section we discuss the *sample complexity* of our estimator. Since the focus here is on the empirical estimation problem, we assume that the mutual information is well enough approximated by the neural information measure $I_\Theta(X,Z)$. The theorem below is a refinement of Lemma 5.3: it gives how many samples we need for an empirical estimation of the neural information measure at a given accuracy and with high confidence.

We make the following assumptions: the functions T_θ are L-Lipschitz with respect to the parameters θ, and both T_θ and e^{T_θ} are M-bounded (i.e., $|T_\theta|, e^{T_\theta} \leq M$).[7] The domain $\Theta \subset \mathbb{R}^d$ is bounded, so that $\|\theta\| \leq K$ for some constant K. The theorem below shows a sample complexity of $\widetilde{O}\left(\frac{d \log d}{\epsilon^2}\right)$, where d is the dimension of the parameter space.

[7]We thank David McAllester for pointing out a mistake in the original published version of this work, which assumed only $|T_\theta| \leq M$. The bound (15) has an exponential dependence in the range of the statistic network T_θ.

Theorem 5.5. *Given any values ϵ,δ of the desired accuracy and confidence parameters, we have,*

$$\Pr\left(|\widehat{I(X;Z)}_n - I_\Theta(X,Z)| \leq \epsilon\right) \geq 1-\delta, \tag{14}$$

whenever the number n of samples satisfies

$$n \geq \frac{2M^2(d\log(16KL\sqrt{d}/\epsilon) + 2dM + \log(2/\delta))}{\epsilon^2}. \tag{15}$$

4. Empirical comparisons

Before diving into applications, we perform some simple empirical evaluation and comparisons of MINE. The objective is to show that MINE is effectively able to estimate mutual information and account for non-linear dependence.

4.1. Comparing MINE to non-parametric estimation

We compare MINE and MINE-f to the k-NN-based non-parametric estimator found in Kraskov et al. (2004). In our experiment, we consider multivariate Gaussian random variables, X_a and X_b, with componentwise correlation, $corr(X_a^i, X_b^j) = \delta_{ij}\,\rho$, where $\rho \in (-1,1)$ and $\delta_i j$ is Kronecker's delta. As the mutual information is invariant to continuous bijective transformations of the considered variables, it is enough to consider standardized Gaussians marginals. We also compare MINE (using the Donsker-Varadhan representation in Eqn. 6) and MINE-f (based on the f-divergence representation in Eqn. 8).

Our results are presented in Figs. 5.1. We observe that both MINE and Kraskov's estimation are virtually indistinguishable from the ground truth when estimating the mutual information between bivariate Gaussians. MINE shows marked improvement over Krakov's when estimating the mutual information between twenty dimensional random variables. We also remark that MINE provides a tighter estimate of the mutual information than MINE-f.

4.2. Capturing non-linear dependencies

An important property of mutual information between random variables with relationship $Y = f(X) + \sigma \odot \epsilon$, where f is a deterministic non-linear transformation and ϵ is random noise, is that it is invariant to the deterministic nonlinear transformation, but should only depend on the amount of noise, $\sigma \odot \epsilon$. This important property, that guarantees the quantification dependence without bias for the relationship, is called *equitability* (Kinney & Atwal, 2014). Our results (Fig. 5.2) show that MINE captures this important property.

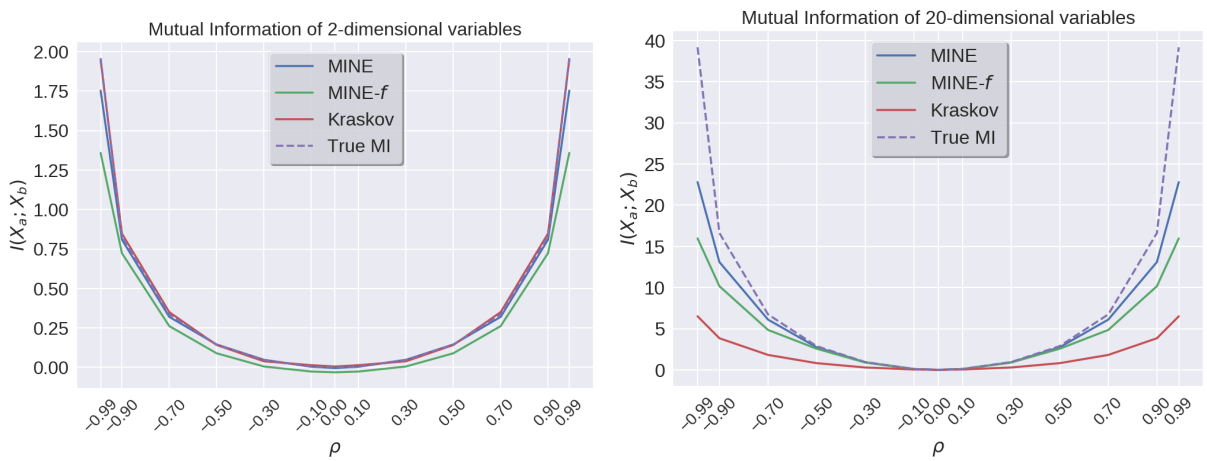


Fig. 5.1. Mutual information between two multivariate Gaussians with component-wise correlation $\rho \in (-1,1)$.

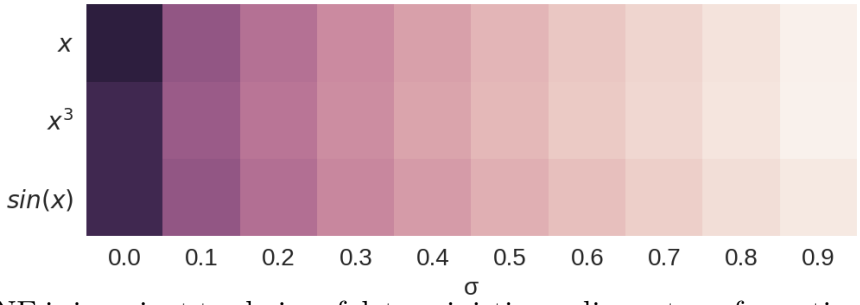


Fig. 5.2. MINE is invariant to choice of deterministic nonlinear transformation. The heatmap depicts mutual information estimated by MINE between 2-dimensional random variables $X \sim \mathcal{U}(-1,1)$ and $Y = f(X) + \sigma \odot \epsilon$, where $f(x) \in \{x, x^3, sin(x)\}$ and $\epsilon \sim \mathcal{N}(0, I)$.

5. Applications

In this section, we use MINE to present applications of mutual information and compare to competing methods designed to achieve the same goals. Specifically, by using MINE to maximize the mutual information, we are able to improve mode representation and reconstruction of generative models. Finally, by minimizing mutual information, we are able to effectively implement the information bottleneck in a continuous setting.

5.1. Maximizing mutual information to improve GANs

Mode collapse (Che et al., 2016; Dumoulin et al., 2016; Donahue et al., 2016; Salimans et al., 2016; Metz et al., 2017; Saatchi & Wilson, 2017; Nguyen et al., 2017; Lin et al., 2017; Ghosh et al., 2017) is a common pathology of generative adversarial networks (GANs, Goodfellow et al., 2014a), where the generator fails to produces samples with sufficient diversity (i.e., poorly represent some modes).

GANs as formulated in Goodfellow et al. (2014a) consist of two components: a discriminator, $D : \mathcal{X} \to [0, 1]$ and a generator, $G : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X}$ is a domain such as a compact subspace of $\mathbb{R}^n$. Given $Z \in \mathcal{Z}$ follows some simple prior distribution (e.g., a spherical Gaussian with density, $\mathbb{P}_Z$), the goal of the generator is to match its output distribution to a target distribution, $\mathbb{P}_X$ (specified by the data samples). The discriminator and generator are optimized through the *value function*,

$$\min_G \max_D V(D,G) := \mathbb{E}_{\mathbb{P}_X}[D(X)] + \mathbb{E}_{\mathbb{P}_Z}[\log(1 - D(G(Z))]. \tag{16}$$

A natural approach to diminish mode collapse would be regularizing the generator's loss with the neg-entropy of the samples. As the sample entropy is intractable, we propose to use the mutual information as a proxy.

Following Chen et al. (2016), we write the prior as the concatenation of noise and code variables, $Z = [\boldsymbol{\epsilon}, \boldsymbol{c}]$. We propose to palliate mode collapse by maximizing the mutual information between the samples and the code. $I(G([\boldsymbol{\epsilon}, \boldsymbol{c}]); \boldsymbol{c}) = H(G([\boldsymbol{\epsilon}, \boldsymbol{c}])) - H(G([\boldsymbol{\epsilon}, \boldsymbol{c}]) \mid \boldsymbol{c})$. The generator objective then becomes,

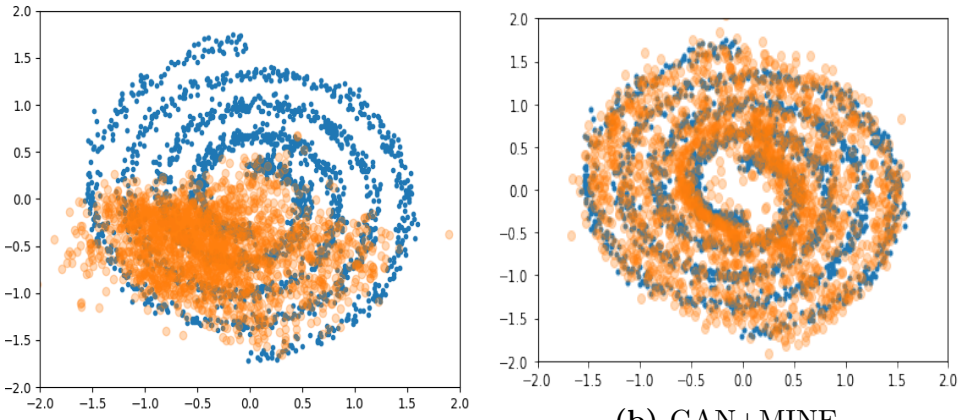
$$\arg\max_G \mathbb{E}[\log(D(G([\boldsymbol{\epsilon}, \boldsymbol{c}])))] + \beta I(G([\boldsymbol{\epsilon}, \boldsymbol{c}]); \boldsymbol{c}). \tag{17}$$

As the samples $G([\boldsymbol{\epsilon}, \boldsymbol{c}])$ are differentiable w.r.t. the parameters of G, and the statistics network being a differentiable function, we can maximize the mutual information using back-propagation and gradient ascent by only specifying this additional loss term. Since the mutual information is theoretically unbounded, we use adaptive gradient clipping (see the Supplementary Material) to ensure that the generator receives learning signals similar in magnitude from the discriminator and the statistics network.

Related works on mode-dropping. Methods to address mode dropping in GANs can readily be found in the literature. Salimans et al. (2016) use mini-batch discrimination. In the same spirit, Lin et al. (2017) successfully mitigates mode dropping in GANs by modifying the discriminator to make decisions on multiple real or generated samples. Ghosh et al. (2017) uses multiple generators that are encouraged to generate different parts of the target distribution. Nguyen et al. (2017) uses two discriminators to minimize the KL and reverse KL divergences between the target and generated distributions. Che et al. (2016) learns a reconstruction distribution, then teach the generator to sample from it, the intuition being that the reconstruction distribution is a de-noised or *smoothed* version of the data distribution, and thus easier to learn. Srivastava et al. (2017) minimizes the reconstruction error in the latent space of bi-directional GANs (Dumoulin et al., 2016; Donahue et al., 2016). Metz et al. (2017) includes many steps of the discriminator's optimization as part of the generator's objective. While Chen et al. (2016) maximizes the mutual information between the code

and the samples, it does so by minimizing a variational upper bound on the conditional entropy (Barber & Agakov, 2003) therefore ignoring the entropy of the samples. Chen et al. (2016) makes no claim about mode-dropping.

Experiments: Spiral, 25-Gaussians datasets. We apply MINE to improve mode coverage when training a generative adversarial network (GAN, Goodfellow et al., 2014a). We demonstrate using Eqn. 17 on the spiral and the 25-Gaussians datasets, comparing two models, one with $\beta = 0$ (which corresponds to the orthodox GAN as in Goodfellow et al. (2014a)) and one with $\beta = 1.0$, which corresponds to mutual information maximization.



(a) GAN **(b)** GAN+MINE

Fig. 5.3. The generator of the GAN model without mutual information maximization after 5000 iterations suffers from mode collapse (has poor coverage of the target dataset) compared to GAN+MINE on the spiral experiment.

Our results on the spiral (Fig. 5.3) and the 25-Gaussians (Fig. 5.4) experiments both show improved mode coverage over the baseline with no mutual information objective. This confirms our hypothesis that maximizing mutual information helps against mode-dropping in this simple setting.

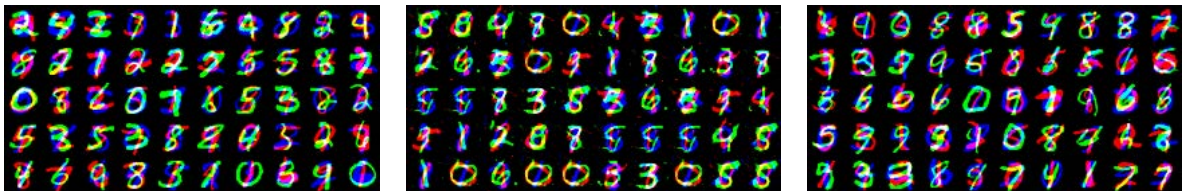
Experiment: Stacked MNIST. Following Che et al. (2016); Metz et al. (2017); Srivastava et al. (2017); Lin et al. (2017), we quantitatively assess MINE's ability to diminish mode dropping on the stacked MNIST dataset which is constructed by stacking three randomly sampled MNIST digits. As a consequence, stacked MNIST offers 1000 modes. Using the same architecture and training protocol as in Srivastava et al. (2017); Lin et al. (2017), we train a GAN on the constructed dataset and use a pre-trained classifier on 26,000 samples to count the number of modes in the samples, as well as to compute the KL divergence between the sample and expected data distributions. Our results in Table 5.1 demonstrate the effectiveness of MINE in preventing mode collapse on Stacked MNIST.

(a) Original data **(b)** GAN **(c)** GAN+MINE

Fig. 5.4. Kernel density estimate (KDE) plots for GAN+MINE samples and GAN samples on 25 Gaussians dataset.

	Stacked MNIST	
	Modes (Max 1000)	KL
DCGAN	99.0	3.40
ALI	16.0	5.40
Unrolled GAN	48.7	4.32
VEEGAN	150.0	2.95
PacGAN	1000.0 ± 0.0	$0.06 \pm 1.0\text{e}^{-2}$
GAN+MINE (Ours)	1000.0 ± 0.0	$0.05 \pm 6.9\text{e}^{-3}$

Table 5.1. Number of captured modes and Kullblack-Leibler divergence between the training and samples distributions for DCGAN (Radford et al., 2015), ALI (Dumoulin et al., 2016), Unrolled GAN (Metz et al., 2017), VeeGAN (Srivastava et al., 2017), PacGAN (Lin et al., 2017).



(a) Training set **(b)** DCGAN **(c)** DCGAN+MINE

Fig. 5.5. Samples from the Stacked MNIST dataset along with generated samples from DCGAN and DCGAN with MINE. While DCGAN only shows a very limited number of modes, the inclusion of MINE generates a much better representative set of samples.

5.2. Maximizing mutual information to improve inference in bi-directional adversarial models

Adversarial bi-directional models were introduced in Adversarially Learned Inference (ALI, Dumoulin et al., 2016) and BiGAN (Donahue et al., 2016) and are an extension of GANs which incorporate a reverse model, $F : \mathcal{X} \to \mathcal{Z}$ jointly trained with the generator. These

models formulate the problem in terms of the value function in Eqn. 16 between two joint distributions, $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z} \mid \boldsymbol{x})p(\boldsymbol{x})$ and $q(\boldsymbol{x}, \boldsymbol{z}) = q(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$ induced by the forward (encoder) and reverse (decoder) models, respectively[8].

One goal of bi-directional models is to do inference as well as to learn a good generative model. Reconstructions are one desirable property of a model that does both inference and generation, but in practice ALI can lack fidelity (i.e., reconstructs less faithfully than desired, see Li et al., 2017; Ulyanov et al., 2017; Belghazi et al., 2018). To demonstrate the connection to mutual information, it can be shown (see the Supplementary Material for details) that the reconstruction error, $\mathcal{R}$, is bounded by,

$$\mathcal{R} \leq D_{KL}(q(\boldsymbol{x}, \boldsymbol{z}) \,||\, p(\boldsymbol{x}, \boldsymbol{z})) - I_q(\boldsymbol{x}, \boldsymbol{z}) + H_q(\boldsymbol{z}) \tag{18}$$

If the joint distributions are matched, $H_q(\boldsymbol{z})$ tends to $H_p(\boldsymbol{z})$, which is fixed as long as the prior, $p(\boldsymbol{z})$, is itself fixed. Subsequently, maximizing the mutual information minimizes the expected reconstruction error.

Assuming that the generator is the same as with GANs in the previous section, the objectives for training a bi-directional adversarial model then become:

$$\begin{aligned} \arg\max_{D} \quad & \mathbb{E}_{q(\boldsymbol{x},\boldsymbol{z})}[\log D(\boldsymbol{x}, \boldsymbol{z})] + \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{z})}[\log (1 - D(\boldsymbol{x}, \boldsymbol{z}))] \\ \arg\max_{F,G} \quad & \mathbb{E}_{q(\boldsymbol{x},\boldsymbol{z})}[\log (1 - D(\boldsymbol{x}, \boldsymbol{z}))] + \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{z})}[\log D(\boldsymbol{x}, \boldsymbol{z})] \\ & + \beta I_q(\boldsymbol{x}, \boldsymbol{z}). \end{aligned} \tag{19}$$

Related works. Ulyanov et al. (2017) improves reconstructions quality by forgoing the discriminator and expressing the adversarial game between the encoder and decoder. Kumar et al. (2017) augments the bi-directional objective by considering the reconstruction and the corresponding encodings as an additional fake pair. Belghazi et al. (2018) shows that a Markovian hierarchical generator in a bi-directional adversarial model provide a hierarchy of reconstructions with increasing levels of fidelity (increasing reconstruction quality). Li et al. (2017) shows that the expected reconstruction error can be diminished by minimizing the conditional entropy of the observables given the latent representations. The conditional entropy being intractable for general posterior, Li et al. (2017) proposes to augment the generator's loss with an adversarial cycle consistency loss (Zhu et al., 2017) between the observables and their reconstructions.

Experiment: ALI+MINE. In this section we compare MINE to existing bi-directional adversarial models. As the decoder's density is generally intractable, we use three different metrics to measure the fidelity of the reconstructions with respect to the samples; (i) the

[8]We switch to density notations for convenience throughout this section.

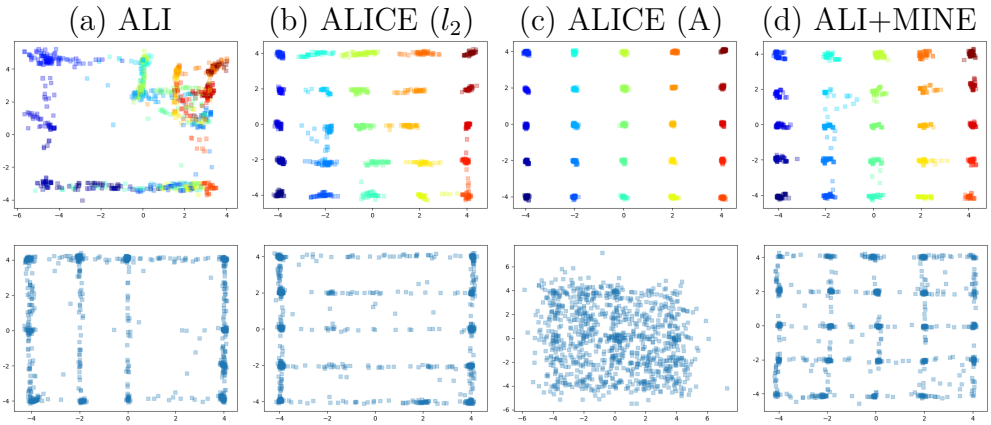


Fig. 5.6. Reconstructions and model samples from adversarially learned inference (ALI) and variations intended to increase improve reconstructions. Shown left to right are the baseline (ALI), ALICE with the l_2 loss to minimize the reconstruction error, ALICE with an adversarial loss, and ALI+MINE. Top to bottom are the reconstructions and samples from the priors. ALICE with the adversarial loss has the best reconstruction, though at the expense of poor sample quality, where as ALI+MINE captures all the modes of the data in sample space.

Model	Recons. Error	Recons. Acc.(%)	MS-SSIM
MNIST			
ALI	14.24	45.95	0.97
ALICE(l_2)	3.20	99.03	0.97
ALICE(Adv.)	5.20	98.17	0.98
MINE	9.73	96.10	0.99
CelebA			
ALI	53.75	57.49	0.81
ALICE(l_2)	8.01	32.22	0.93
ALICE(Adv.)	92.56	48.95	0.51
MINE	36.11	76.08	0.99

Table 5.2. Comparison of MINE with other bi-directional adversarial models in terms of euclidean reconstruction error, reconstruction accuracy, and MS-SSIM on the MNIST and CelebA datasets. MINE does a good job compared to ALI in terms of reconstructions. Though the explicit reconstruction based baselines (ALICE) can sometimes do better than MINE in terms of reconstructions related tasks, they consistently lag behind in MS-SSIM scores and reconstruction accuracy on CelebA.

euclidean reconstruction error, (*ii*) *reconstruction accuracy*, which is the proportion of labels preserved by the reconstruction as identified by a pre-trained classifier; (*iii*) the Multi-scale structural similarity metric (MS-SSIM, Wang et al., 2004) between the observables and their reconstructions.

We train MINE on datasets of increasing order of complexity: a toy dataset composed of 25-Gaussians, MNIST (LeCun, 1998), and the CelebA dataset (Liu et al., 2015). Fig. 5.6 shows the reconstruction ability of MINE compared to ALI. Although ALICE does perfect reconstruction (which is in its explicit formulation), we observe significant mode-dropping in the sample space. MINE does a balanced job of reconstructing along with capturing all the modes of the underlying data distribution.

Next, we measure the fidelity of the reconstructions over ALI, ALICE, and MINE. Tbl. 2 compares MINE to the existing baselines in terms of euclidean reconstruction errors, reconstruction accuracy, and MS-SSIM. On MNIST, MINE outperforms ALI in terms of reconstruction errors by a good margin and is competitive to ALICE with respect to reconstruction accuracy and MS-SSIM. Our results show that MINE's effect on reconstructions is even more dramatic when compared to ALI and ALICE on the CelebA dataset.

5.3. Information Bottleneck

The Information Bottleneck (IB, Tishby et al., 2000) is an information theoretic method for extracting relevant information, or yielding a representation, that an input $X \in \mathcal{X}$ contains about an output $Y \in \mathcal{Y}$. An optimal representation of X would capture the relevant factors and compress X by diminishing the irrelevant parts which do not contribute to the prediction of Y. IB was recently covered in the context of deep learning (Tishby & Zaslavsky, 2015), and as such can be seen as a process to construct an approximation of the minimally sufficient statistics of the data. IB seeks an encoder, $q(Z \mid X)$, that induces the Markovian structure $X \to Z \to Y$. This is done by minimizing the IB Lagrangian,

$$\mathcal{L}[q(Z \mid X)] = H(Y|Z) + \beta I(X,Z), \tag{20}$$

which appears as a standard cross-entropy loss augmented with a regularizer promoting minimality of the representation (Achille & Soatto, 2017). Here we propose to estimate the regularizer with MINE.

Related works. In the discrete setting, Tishby et al. (2000) uses the Blahut-Arimoto Algorithm Arimoto (1972), which can be understood as cyclical coordinate ascent in function spaces. While IB is successful and popular in a discrete setting, its application to the continuous setting was stifled by the intractability of the continuous mutual information. Nonetheless, IB was applied in the case of jointly Gaussian random variables in Chechik et al. (2005).

In order to overcome the intractability of $I(X;Z)$ in the continuous setting, Alemi et al. (2016); Kolchinsky et al. (2017); Chalk et al. (2016) exploit the variational bound of Barber & Agakov (2003) to approximate the conditional entropy in $I(X;Z)$. These approaches differ

only on their treatment of the marginal distribution of the bottleneck variable: Alemi et al. (2016) assumes a standard multivariate normal marginal distribution, Chalk et al. (2016) uses a Student-t distribution, and Kolchinsky et al. (2017) uses non-parametric estimators. Due to their reliance on a variational approximation, these methods require a tractable density for the approximate posterior, while MINE does not.

Experiment: Permutation-invariant MNIST classification. Here, we demonstrate an implementation of the IB objective on permutation invariant MNIST using MINE. We compare to the Deep Variational Bottleneck (DVB, Alemi et al., 2016) and use the same empirical setup. As the DVB relies on a variational bound on the conditional entropy, it therefore requires a tractable density. Alemi et al. (2016) opts for a conditional Gaussian encoder $\boldsymbol{z} = \mu(\boldsymbol{x}) + \sigma \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. As MINE does not require a tractable density, we consider three type of encoders: (i) a Gaussian encoder as in Alemi et al. (2016); (ii) an *additive noise encoder*, $\boldsymbol{z} = enc(\boldsymbol{x} + \sigma \odot \epsilon)$; and (iii) a *propagated noise encoder*, $\boldsymbol{z} = enc([\boldsymbol{x}, \epsilon])$. Our results can be seen in Tbl. 5.3, and this shows MINE as being superior in these settings.

Model	Misclass. rate(%)
Baseline	1.38%
Dropout	1.34%
Confidence penalty	1.36%
Label Smoothing	1.40%
DVB	1.13%
DVB + Additive noise	1.06%
MINE(Gaussian) (ours)	1.11%
MINE(Propagated) (ours)	1.10%
MINE(Additive) (ours)	1.01%

Table 5.3. Permutation Invariant MNIST misclassification rate using Alemi et al. (2016) experimental setup for regularization by confidence penalty (Pereyra et al., 2017), label smoothing (Pereyra et al., 2017), Deep Variational Bottleneck(DVB) (Alemi et al., 2016) and MINE. The misclassification rate is averaged over ten runs. In order to control for the regularizing impact of the additive Gaussian noise in the additive conditional, we also report the results for DVB with additional additive Gaussian noise at the input. All non-MINE results are taken from Alemi et al. (2016).

6. Conclusion

We proposed a mutual information estimator, which we called the mutual information neural estimator (MINE), that is scalable in dimension and sample-size. We demonstrated

the efficiency of this estimator by applying it in a number of settings. First, a term of mutual information can be introduced alleviate mode-dropping issue in generative adversarial networks (GANs, Goodfellow et al., 2014a). Mutual information can also be used to improve inference and reconstructions in adversarially-learned inference (ALI, Dumoulin et al., 2016). Finally, we showed that our estimator allows for tractable application of Information bottleneck methods (Tishby et al., 2000) in a continuous setting.

Chapter 6

Conclusion

In this thesis, we described the results of some preliminary investigations towards a better understanding of generalization in deep learning. We formulated the hypothesis that neural network training is biased towards learning simple functions supported on a small number of highly predictive features. As an attempt to understand the role of feature learning in such a bias, we studied the evolution of the neural tangent kernel and its spectrum during training. We highlighted a combined mechanism of feature selection and geometrical compression, and argue that it helps the network adapt to the intrinsic complexity of the problem. Finally, as a contribution to the deep learning toolbox, we proposed and studied a new estimator of mutual information, which plays a central role in some approaches to representation learning; and we used it as an attempt to enhance deep generative models.

This work opens several avenues for further investigation. Understanding the representation learning dynamics is key to explain generalization. This is a hard problem, but theoretical insights can be derived in simplified settings. For example, for deep linear networks trained on linearly separable data with logistic loss, Ji & Telgarsky (2019) analytically derive a mechanism, closely related to the one described in Chapter 4, of a dynamical alignment of the weight matrices along the max-margin direction. The hierarchical feature alignment observed in Section 3.4 deserves deeper analysis. Further experiments could explore training on more hierarchically structured data such as the dataset used in Saxe et al. (2019). Known results about the sequential learning dynamics of deep linear networks (Saxe et al., 2014; Lampinen et al., 2018; Gidel et al., 2019) or in matrix factorization (Gunasekar et al., 2017) should be extended to the non-linear case. Precise conditions under which 'subdominant' patterns of hierarchical data can be learned without affecting the performance of the dominant ones should be investigated; this can viewed as a generalization of the phenomenon of benign overfitting (Bartlett et al., 2019b, 2021a). An important theoretical problem is also to amend the classical generalization bound techniques so as to integrate and explain such a behaviour.

More broadly, we seek a deeper theoretical understanding and a better control of the way the network distributes its capacity during training depending on geometrical and statistical skews in the data, and articulates learning and memorization to both fit the head and the (long) tail of the data distribution (Sagawa et al., 2020a,b; Hooker et al., 2020; Feldman & Zhang, 2020). This requires reliable metrics that quantify memorization, e.g., through the notion of influence each training example has on the model's test predictions (Koh & Liang, 2017; Barshan et al., 2020), and use these as regularizers to bias learning towards more diverse representations and more robust models.

References

A Achille and S Soatto. Emergence of invariance and disentanglement in deep representations. *arXiv preprint 1706.01350v2[cs.LG]*, 2017.

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

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1): 131–142, 1966. doi: https://doi.org/10.1111/j.2517-6161.1966.tb00626.x.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. volume 97 of *Proceedings of Machine Learning Research*, pp. 242–252, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276, 1998.

Shun-Ichi Amari. *Information Geometry and Its Applications*, volume 194. Springer, 2016.

Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018a.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *ICML*, pp. 224–232, 2017.

Sanjeev Arora, Rong Ge, Behnam @phdthesis, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018b.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations?, 2019.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

A Banerjee. On baysian bounds. *ICML*, pp. 81–88, 2006.

David Barber and Felix Agakov. The im algorithm: a variational approach to information maximization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pp. 201–208. MIT Press, 2003.

Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1899–1909, Online, 26–28 Aug 2020. PMLR.

Peter Bartlett and Shai Ben-David. Hardness results for neural network approximation problems. In Paul Fischer and Hans Ulrich Simon (eds.), *Computational Learning Theory*, pp. 50–62, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49097-5.

Peter Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear vc dimension bounds for piecewise polynomial networks. In M. Kearns, S. Solla, and D. Cohn (eds.), *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1999.

Peter L. Bartlett. For valid generalization, the size of the weights is more important than the size of the network. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, pp. 134–140, Cambridge, MA, USA, 1996. MIT Press.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR*, 2002.

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *NIPS*, 2017.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019a.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *arXiv preprint arXiv:1906.11300[stat.ML]*, 2019b.

Peter L. Bartlett, Philip M. Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907378117.

Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *arXiv:2103.09177 [math.ST]*, 2021a.

Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *arxiv preprint arXiv 2103.0917 [math.ST]*, 2021b.

Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *Advances in Neural Information Processing Systems 32*, pp. 4761–4771. 2019.

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Mohamed Ishmael Belghazi, Sai Rajeswar, Olivier Mastropietro, Jovana Mitrovic, Negar Rostamzadeh, and Aaron Courville. Hierarchical adversarially learned inference. *arXiv preprint arXiv:1802.01071*, 2018.

Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *ICML*, 2018.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik 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.

Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019b.

Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-François Paiement, Pascal Vincent, and Marie Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Steven Bergner, Torsten Möller, Daniel Weiskopf, and David J Muraki. A spectral analysis of function concatenations and its implications for sampling in direct volume visualization.

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. In *Advances in Neural Information Processing Systems 32*, pp. 12893–12904. 2019.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773.

Avrim Blum and Ronald Rivest. Training a 3-node neural network is np-complete. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1989.

Siegfried Bös. Statistical mechanics approach to early stopping and weight decay. *Phys. Rev. E*, 58:833–844, Jul 1998. doi: 10.1103/PhysRevE.58.833.

S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities. A nonasymptotic theory of independence*. Oxford University Press, 2013.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch (eds.), *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pp. 169–207. Springer, 2003. ISBN 3-540-23122-6.

Mikio L Braun. *Spectral properties of the kernel matrix and their relation to kernel methods in machine learning*. PhD thesis, Universitäts-und Landesbibliothek Bonn, 2005.

Mikio L Braun, Tilman Lange, and Joachim M Buhmann. Model selection in kernel methods based on a spectral analysis of label information. In *Joint Pattern Recognition Symposium*, pp. 344–353. Springer, 2006.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478 [cs.LG]*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric

Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Sorelle A. Friedler and Christo Wilson (eds.), *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pp. 77–91. PMLR, 2018.

Atul J Butte and Isaac S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pac Symp Biocomput*, volume 5, pp. 26, 2000.

Emmanuel J Candès. Harmonic analysis of neural networks. *Applied and Computational Harmonic Analysis*, 6(2):197–218, 1999.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Matthew Chalk, Olivier Marre, and Gasper Tkacik. Relevant sparse codes with variational information bottleneck. In *Advances in Neural Information Processing Systems*, pp. 1957–1965, 2016.

Satrajit Chatterjee. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *International Conference on Learning Representations*, 2020.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

Gal Chechik, Amir Globerson, Naftali Tishby, and Yair Weiss. Information bottleneck for gaussian variables. *Journal of Machine Learning Research*, 6(Jan):165–188, 2005.

Kumar Chellapilla, Sidd Puri, and Patrice Simard. High Performance Convolutional Neural Networks for Document Processing. In Guy Lorette (ed.), *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), October 2006. Université de Rennes 1, Suvisoft. http://www.suvisoft.com.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative

adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 3036–3046. Curran Associates, Inc., 2018.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. 2018.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *JMLR*, 13(1):795–828, 2012. ISSN 1532-4435.

Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *NIPS*. 2002.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

Alexander D'Amour, Katherine Heller, Dan Moldovan, and *et al.* Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395 [cs.LG]*, 2020.

Amit Daniely, Sivan Sabato, Shai Ben-David, and Shai Shalev-Shwartz. Multiclass learnability and the erm principle. In Sham M. Kakade and Ulrike von Luxburg (eds.), *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19 of *Proceedings of Machine Learning Research*, pp. 207–232, Budapest, Hungary, 09–11 Jun 2011. PMLR.

Georges A Darbellay and Igor Vajda. Estimation of the information by an adaptive partitioning of the observation space. *IEEE Transactions on Information Theory*, 45(4):1315–1321, 1999.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Luc Devroye, László Györfi, and Gábor Lugosi. Consistency of the k-nearest neighbor rule. In *A Probabilistic Theory of Pattern Recognition*, pp. 169–185. Springer, 1996.

Ricardo Diaz, Quang-Nhat Le, and Sinai Robins. Fourier transforms of polytopes, solid angle sums, and discrete volume. *arXiv preprint arXiv:1602.08593*, 2016.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

David L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. In *AMS Conference on math challenges of the 21st century*, 2000.

M.D Donsker and S.R.S Varadhan. Asymptotic evaluation of certain markov process expectations for large time, iv. *Communications on Pure and Applied Mathematics*, 36(2):183?212, 1983.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2018.

Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *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*, pp. 1675–1685. PMLR, 2019a.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019b.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the 33rd Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

Gintare Karolina Dziugaite, Alexandre Drouin, Brady Neal, Nitarshan Rajkumar, Ethan Caballero, Linbo Wang, Ioannis Mitliagkas, and Daniel M. Roy. In search of robust measures of generalization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pp. 907–940, 2016.

Akram Erraqabi, Aristide Baratin, Yoshua Bengio, and Simon Lacoste-Julien. A3t: Adversarially augmented adversarial training. *Machine Deception Workshop, NIPS 2017*, 2017.

Gregory E Fasshauer. Positive definite kernels: past, present and future. *Dolomite Research Notes on Approximation*, 4:21–63, 2011.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *arXiv preprint arXiv:2008.03703 [cs.LG]*, 2020.

Stanislav Fort, Pawel Krzysztof Nowak, Stanislaw Jastrzebski, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

Stanislav Fort, Karolina Gintare Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *arXiv preprint arXiv:2010.15110 [cs.LG]*, 2020.

Andrew M Fraser and Harry L Swinney. Independent coordinates for strange attractors from mutual information. *Physical review A*, 33(2):1134, 1986.

Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Efficient estimation of mutual information for strongly dependent variables. *Arxiv preprint arXiv:1411.2003[cs.IT]*, 2014.

Mario Geiger, Stefano Spigler, Stéphane d'Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115, 2019.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. 2020(11):113301, nov 2020a. doi: 10.1088/1742-5468/abc4de.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, nov 2020b. doi: 10.1088/1742-5468/abc4de.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *arxiv preprint arXiv:2004.07780 [cs.CV]*, 2020.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992. doi: 10.1162/neco.1992.4.1.1.

Thomas George. NNGeometry: Easy and Fast Fisher Information Matrices and Neural Tangent Kernels in PyTorch, February 2021.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *arXiv preprint arXiv:2006.13409[stat.ML]*, 2020.

Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. *arXiv preprint arXiv:1704.02906*, 2017.

Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 3202–3211. Curran Associates, Inc., 2019.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

Surbhi Goel, Adam R. Klivans, Pasin Manurangsi, and Daniel Reichman. Tight hardness results for training depth-2 relu networks. In *ITCS*, pp. 22:1–22:14, 2021.

Micah Goldblum, Jonas Geiping, Avi Schwarzschild, Michael Moeller, and Tom Goldstein. Truth or backpropaganda? an empirical investigation of deep learning theory. *arXiv:1910.00359 [cs.LG]*, 2020.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014a.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016a. ISBN 0262035618, 9780262035613.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016b. http://www.deeplearningbook.org.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Arthur Gretton, Olivier Bousquet, Alexander Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms, 2005.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 6151–6159. Curran Associates, Inc., 2017.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In Jennifer Dy and Andreas Krause (eds.), *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1832–1841, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

L Györfi and E. C van der Meulen. Density-free convergence properties of various estimators of entropy. *Computational Statistics and Data Analysis*, 5:425?436, 1987.

Barbara Hammer and Kai Gersmann. A note on the universal approximation capability of support vector machines. *Neural Processing Letters*, 17(1):43–53, 2003.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv:1903.08560 [math.ST]*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *NIPS*, 2017.

Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248 [cs.LG]*, 2020.

K Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2: 359–366, 1989.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Gabriel Huang, Hugo Berard, Ahmed Touati, Gauthier Gidel, Pascal Vincent, and Simon Lacoste-Julien. Parametric adversarial divergences are good task losses for generative modeling. *arXiv preprintarXiv:1708.02511*, 2017.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NIPS*, pp. 8571–8580. 2018.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, 2009. doi: 10.1109/ICCV.2009.5459469.

Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations*, 2019.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020.

J. Stephen Judd. On the complexity of loading shallow neural networks. *J. Complex.*, 4(3): 177–192, 1988. doi: 10.1016/0885-064X(88)90019-2.

K Kandasamy, A Krishnamurthy, B Poczos, L Wasserman, and J.M Robins. Nonparametric von mises estimators for entropies, divergences and mutual informations. *NIPS*, 2017.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Pathological spectra of the fisher information metric and its variants in deep neural networks. *arXiv:1910.05992 [stat.ML]*, 2019a.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. *AISTATS 2019*, 2019b.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Amor Keziou. Dual representation of phi-divergences and applications. 336:857–862, 05 2003.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Justin B Kinney and Gurinder S Atwal. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences*, 111(9):3354–3359, 2014.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Artemy Kolchinsky, Brendan D Tracey, and David H Wolpert. Nonlinear information bottleneck. *arXiv preprint arXiv:1705.02436*, 2017.

Esben L Kolsbjerg, Michael N Groves, and Bjork Hammer. An automated nudged elastic band method. *The Journal of chemical physics*, 145(9):094107, 2016.

D. Kopitkov and V. Indelman. Neural spectrum alignment: Empirical study. In *International Conference on Artificial Neural Networks (ICANN)*, September 2020.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

Abhishek Kumar, Prasanna Sattigeri, and P Thomas Fletcher. Improved semi-supervised learning with gans using manifold invariances. *arXiv preprint arXiv:1705.08850*, 2017.

Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on parzen window. *IEEE transactions on pattern analysis and machine intelligence*, 24(12): 1667–1671, 2002.

Andrew K Lampinen, Andrew K Lampinen, and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. *arXiv.org*, 2018.

S. Lang. *Fundamentals of Differential Geometry*. Graduate Texts in Mathematics. Springer New York, 2012. ISBN 9781461205418.

John Langford and Rich Caruana. (not) bounding the true error. In T. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer Science & Business, New York, 2013.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8572–8583. Curran Associates, Inc., 2019.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2018.

Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Chunyuan Li, Hao Liu, Changyou Chen, Yunchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. Towards understanding adversarial learning for joint distribution matching. *arXiv preprint arXiv:1709.01215*, 2017.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *Proceedings of Machine Learning Research*,

volume 89, pp. 888–896, 2019.

Junyang Lin, An Yang, Jinze Bai, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Yong Li, Wei Lin, Jingren Zhou, and Hongxia Yang. M6-10t: A sharing-delinking paradigm for efficient multi-trillion parameter pretraining. *arXiv preprint arXiv:2110.03888 [cs.LG]*, 2021.

Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *arXiv preprint arXiv:1712.04086*, 2017.

R. Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988. doi: 10.1109/2.36.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

Bruno Loureiro, Cédric Gerbelot, Hugo Cui, Sebastian Goldt, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Capturing the learning curves of generic features maps for realistic data sets with a teacher-student model. *CoRR*, abs/2102.08127, 2021.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pp. 6231–6239, 2017.

Siyuan Ma and Mikhail Belkin. Diving into the shallows: a computational perspective on large-scale shallow learning. In *Advances in Neural Information Processing Systems*, pp. 3781–3790, 2017.

Wolfgang Maass. Neural nets with superlinear vc-dimension. *Neural Computation*, 6(5): 877–884, 1994. doi: 10.1162/neco.1994.6.5.877.

Frederik Maes, Andre Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens. Multimodality image registration by maximization of mutual information. *IEEE transactions on Medical Imaging*, 16(2):187–198, 1997.

David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 875–884. PMLR, 26–28 Aug 2020.

David A. McAllester. Pac-bayesian model averaging. In *In Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pp. 164–170. ACM Press, 1999.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv:1908.05355 [math.ST]*, 2019.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33): E7665–E7671, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1806579115.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. 2017.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2019.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

K.R Moon, K Sricharan, and A. O Hero III. Ensemble estimation of mutual information. *arXiv preprint arXiv:1701.08083*, 2017.

Young-Il Moon, Balaji Rajagopalan, and Upmanu Lall. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3):2318, 1995.

Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai. Harmless interpolation of noisy data in regression. *arXiv preprint arXiv:1903.09139[cs.LG]*, 2019.

Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Sahai, Hsu, and Anant Sahai. Classification vs regression in overparameterized regimes: Does the loss function matter? *arXiv preprint arXiv:2005.08054 [cs.LG]*, 2020.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997. doi: 10.2307/1428011.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 11615–11626. Curran Associates, Inc., 2019a.

Vaishnavh Nagarajan and Zico Kolter. Deterministic PAC-bayesian generalization bounds for deep networks via generalizing noise-resilience. In *International Conference on Learning Representations*, 2019b.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. SGD on neural networks learns functions of increasing complexity. In *NeurIPS 2019 (spotlight)*, volume abs/1905.11604, 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.

B. K. Natarajan. On learning sets and functions. *Machine Learning*, 4:67–97, 1989.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv:1810.08591 [cs.LG]*, 2018.

MEJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5): 323–351, 2005. doi: 10.1080/00107510500052444.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *ICLR workshop track*, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5949–5958, 2017a.

Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv:1705.03071 [cs.LG]*, 2017b.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *International Conference on Learning Representations (ICLR)*, 2019.

Tu Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2667–2677, 2017.

XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pp. 271–279, 2016.

Manfred Opper. Statistical mechanics of learning: Generalization. *The Handbook of Brain Theory and Neural Networks, 922-925.*, 1995.

Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. What can linearized neural networks actually say about generalization? *arXiv:2106.06770 [cs.LG]*, 2021.

Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural networks. *Journal Of Statistical Mechanics-Theory And Experiment*, 2021(4):044001, 2021.

Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6): 1191–1253, 2003.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *ICLR Workshop*, 2017.

T Poggio, K Kawaguchi, Q Liao, B Miranda, L Rosasco, X Boix, J Hidary, and HN Mhaskar. Theory of deep learning iii: the non-overfitting puzzle. Technical report, Technical report, CBMM memo 073, 2018.

Ben Poole, Subhaneil Lahiri, Maithreyi Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3360–3368. Curran Associates, Inc., 2016.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.),

Proceedings of the 36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, pp. 5171–5180. PMLR, 09–15 Jun 2019.

Ida Mengyi Pu. *Fundamental data compression*. Butterworth-Heinemann, 2006. ISBN 1-281-03498-3.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pp. 63–71. Springer, 2004.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

G Rockafellar. *Convex Analysis*. Princeton U, 1970.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pp. 606–610. IEEE, 2007.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arxiv preprint arXiv:1609.04747 [cs.LG]*, 2016.

Avraham Ruderman, Mark Reid, Darío García-García, and James Petterson. Tighter variational representations of f-divergences via restriction to probability measures. *arXiv preprint arXiv:1206.4664*, 2012.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

Yunus Saatchi and Andrew G Wilson. Bayesian gan. In *Advances in Neural Information Processing Systems*, pp. 3625–3634, 2017.

Itay Safran and Ohad Shamir. Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks. In *Proceedings of the 34th International Conference on Machine*

Learning, pp. 2979–2987. PMLR, 2017.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020a.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. *arXiv:2005.04345 [cs.LG]*, 2020b.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT)*, volume PMLR 99, pp. 2667–2690, 2019.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *In International Conference on Learning Representations*, 2014.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1820226116.

B. Schölkopf, S. Mika, C. J.C. Burges, P. Knirsch, K. R. Muller, G. Ratsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *Trans. Neur. Netw.*, 10(5): 1000–1017, September 1999a. ISSN 1045-9227.

B. Schölkopf, J. Shawe-Taylor, AJ. Smola, and RC. Williamson. Kernel-dependent support vector error bounds. In *Artificial Neural Networks, 1999. ICANN 99*, volume 470 of *Conference Publications*, pp. 103–108. Max-Planck-Gesellschaft, IEEE, 1999b.

Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Zidek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020. doi: 10.1038/s41586-019-1923-7.

Valery Serov. *Fourier series, Fourier transform and their applications to mathematical physics*. Springer, 2017.

H. S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Phys. Rev. A*, 45:6056–6091, Apr 1992. doi: 10.1103/PhysRevA.45.6056.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *NeurIPS*, 2020.

Shai Shalev-Schwartz and Shai Ben-David. *Understanding Machine Learning - from Theory to Algorithms.* Cambridge university press, 2014.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014. ISBN 978-1-10-705713-5.

Haozhe Shan and Blake Bordelon. Rapid feature evolution accelerates learning in neural networks. *arXiv:2105.14301 [stat.ML]*, 2021.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

S Singh and B Póczos. Finite-sample analysis of fixed-k nearest neighbor density functional estimators. *arXiv preprint 1606.01554*, 2016.

Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, 2020.

Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *arXiv preprint arXiv:1710.10345*, 2017.

Michael Spivak. *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus*. CRC press, 2018.

Nati Srebro, Karthik Sridharan, and Ambuj Tewari. On the universality of online mirror descent. In *Advances in Neural Information Processing Systems 24*. 2011.

Sreejith Sreekumar and Ziv Goldfeld. Neural estimation of statistical divergences. *arXiv:2110.03652 [math.ST]*, 2021.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv preprint arXiv:1705.07761*, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Taiji Suzuki, Masashi Sugiyama, Jun Sese, and Takafumi Kanamori. Approximating mutual information by maximum likelihood density ratio estimation. In *New challenges for feature selection in data mining and knowledge discovery*, pp. 5–20, 2008.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644 [cs.LG]*, 2020.

Matus Telgarsky. Benefits of depth in neural networks. *Conference on Learning Theory (COLT), 2016*, 2016.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pp. 1–5. IEEE, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2020.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Adversarial generator-encoder networks. *arXiv preprint arXiv:1704.02304*, 2017.

Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi: 10.1145/1968.1972.

Sara Van de Geer. *Empirical Processes in M-estimation*. Cambridge University Press, 2000.

Marc M Van Hulle. Edgeworth approximation of multivariate differential entropy. *Neural computation*, 17(9):1903–1910, 2005.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michael Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Remi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wunsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z.

Van H. Vu. On the infeasibility of training neural networks with small squared errors. In M. Jordan, M. Kearns, and S. Solla (eds.), *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

James Vuckovic, Aristide Baratin, and Remi Tachet des Combes. On the regularity of attention. *arXiv preprint arXiv:2007.02876 [stat.ML]*, 2020.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13: 600–612, 2004.

Timothy L. H. Watkin, Albrecht Rau, and Michael Biehl. The statistical mechanics of learning a rule. *Rev. Mod. Phys.*, 65:499–556, Apr 1993. doi: 10.1103/RevModPhys.65.499.

Francis Williams, Matthew Trager, Daniele Panozzo, Claudio Silva, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. *arXiv:2002.09277 [cs.LG]*, 2020.

Yuege Xie, Rachel Ward, Holger Rauhut, and Chou Hung-Hsu. Weighted optimization: better generalization by smoother interpolation. *arXiv preprint arXiv:2006.08495*, 2020.

Han Xu, Ya Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. *arXiv preprint arXiv:1807.01251*, 2018.

Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. In Tom Gedeon, Kok Wai Wong, and Minho Lee (eds.), *Neural Information Processing*, pp. 264–274, Cham, 2019. Springer International Publishing. ISBN 978-3-030-36708-4.

Zhiqin John Xu. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548 [stat.ML]*, 2020.

Greg Yang and Edward J. Hu. Tensor programs IV: feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 2021.

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. *ICML*, 2021.

Greg Yang and Hadi Salman. A fine grained spectral perspective on neural networks. *arxiv preprint arXiv:1907.10599[cs.LG]*, 2019.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2017.07.002.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017b.

Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. *arXiv preprint arXiv:1805.07091*, 2018a.

Pengchuan Zhang, Qiang Liu, Dengyong Zhou, Tao Xu, and Xiaodong He. On the discrimination-generalization tradeoff in GANs. In *International Conference on Learning Representations*, 2018b.

Xiao Zhang, Haoyi Xiong, and Dongrui Wu. Rethink the connections among generalization, memorization, and the spectral bias of dnns. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 3392–3398. ijcai.org, 2021. doi: 10.24963/ijcai.2021/467.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*,

2017.

Appendix A

Spectral Bias: Supplementary Material

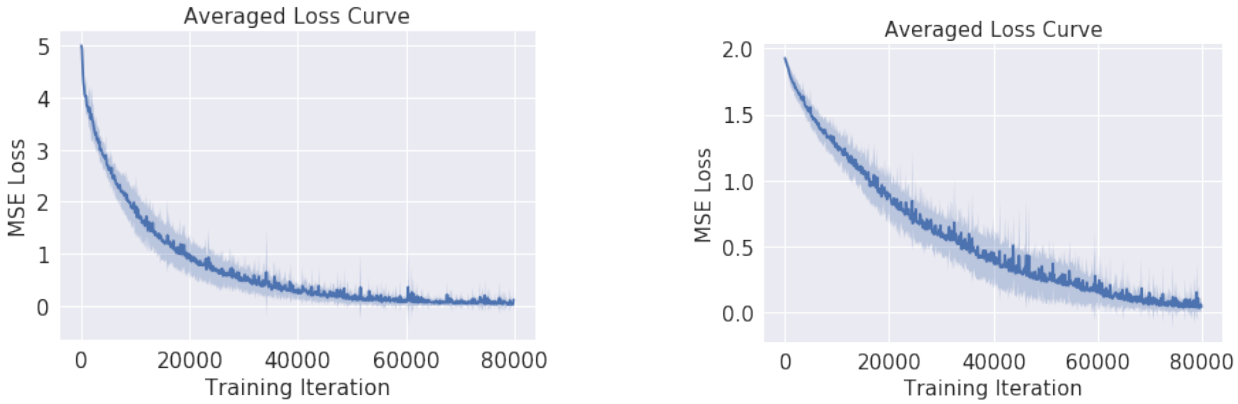
A.1. Experimental Details

A.1.1. Experiment 1

We fit a 6 layer ReLU network with 256 units per layer f_θ to the target function λ, which is a superposition of sine waves with increasing frequencies:

$$\lambda : [0,1] \to \mathbb{R},\ \lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i)$$

where $k_i = (5, 10, 15, ..., 50)$, and φ_i is sampled from the uniform distribution $U(0, 2\pi)$. In the first setting, we set equal amplitude for all frequencies, i.e. $A_i = 1 \,\forall\, i$, while in the second setting we assign larger amplitudes to the higher frequencies, i.e. $A_i = (0.1, 0.2, ..., 1)$. We sample λ on 200 uniformly spaced points in $[0, 1]$ and train the network for 80000 steps of full-batch gradient descent with Adam (Kingma & Ba, 2015). Note that we do not use stochastic gradient descent to avoid the stochasticity in parameter updates as a confounding factor. We evaluate the network on the same 200 point grid every 100 training steps and compute the magnitude of its (single-sided) discrete fourier transform at frequencies k_i which we denote with $|\tilde{f}_{k_i}|$. Finally, we plot in figure 3.1 the normalized magnitudes $\frac{|\tilde{f}_{k_i}|}{A_i}$ averaged over 10 runs (with different sets of sampled phases φ_i). We also record the spectral norms of the weights at each layer as the training progresses, which we plot in figure 3.1 for both settings (the spectral norm is evaluated with 10 power iterations). In figure 3.2, we show an example target function and the predictions of the network trained on it (over the iterations), and in figure A.1 we plot the loss curves.



(a) Equal Amplitudes. **(b)** Increasing Amplitudes.

Fig. A.1. Loss curves averaged over multiple runs. (cf. Experiment 1)

A.1.2. Experiment 5

We use the same 6-layer deep 256-unit wide network and define the target function

$$\lambda : \mathcal{D} \to \mathbb{R},\ z \mapsto \lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i)$$

where $k_i = (20, 40, ..., 180, 200)$, $A_i = 1\,\forall\, i$ and $\varphi \sim U(0, 2\pi)$. We sample ϕ on a grid with 1000 uniformly spaced points between 0 and 1 and map it to the input domain via γ_L to obtain a dataset $\{(\gamma_L(z_j), \lambda(z_j))\}_{j=0}^{999}$, on which we train the network with 50000 full-batch gradient descent steps of Adam. On the same 1000-point grid, we evaluate the magnitude of the (single-sided) discrete Fourier transform of $f_\theta \circ \gamma_L$ every 100 training steps at frequencies k_i and average over 10 runs (each with a different set of sampled z_i's). Fig 3.8 shows the evolution of the spectrum as training progresses for $L = 0, 4, 10, 16$, and Fig 3.8e shows the corresponding loss curves.

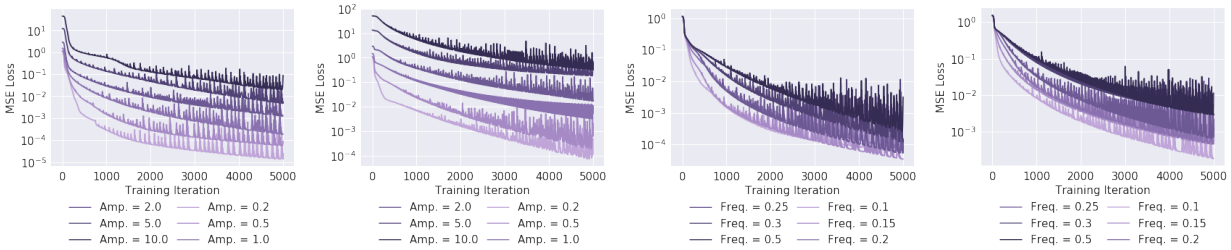
A.1.3. Experiment 3

In Figure A.2, we show the training curves corresponding to Figure 3.4.

A.1.4. Experiment 4

Consider the Gaussian Radial Basis Kernel, given by:

$$k : X \times X \to \mathbb{R}, k_\sigma(\mathbf{x}, \boldsymbol{y}) \mapsto \exp\left(\frac{\|\mathbf{x} - \boldsymbol{y}\|}{\sigma^2}\right) \tag{1}$$



(a) $k = 0.1$ **(b)** $k = 1$ **(c)** $\beta = 0.5$ **(d)** $\beta = 1.$

Fig. A.2. (a,b,c,d): Training curves for various settings of noise amplitude β and frequency k corresponding to Figure 3.4.

where X is a compact subset of $\mathbb{R}^d$ and $\sigma \in \mathbb{R}_+$ is defined as the width of the kernel[1]. Since k is positive definite Fasshauer (2011), Mercer's Theorem can be invoked to express it as:

$$k(\mathbf{x}, \boldsymbol{y}) = \sum_{n=1}^{\infty} \lambda_i \varphi_n(\mathbf{x}) \varphi_n(\boldsymbol{y}) \tag{2}$$

where φ_n is the eigenfunction of k satisfying:

$$\int k(\mathbf{x}, \boldsymbol{y}) \varphi_n(\boldsymbol{y}) \mathbf{dy} = \langle k(\mathbf{x}, \cdot), \varphi_n \rangle = \lambda_n \varphi_n(\mathbf{x}) \tag{3}$$

Due to positive definiteness of the kernel, the eigenvalues λ_i are non-negative and the eigenfunctions φ_n form an orthogonal basis of $L^2(X)$, i.e. $\langle \varphi_i, \varphi_j \rangle = \delta_{ij}$. The analogy to the final case is easily seen: let $X = \mathbf{x}_{i=1}^N$ be the set of samples, $f : X \to \mathbb{R}$ a function. One obtains (cf. Chapter 4 Rasmussen (2004)):

$$\langle k(\mathbf{x}, \cdot), f \rangle = \sum_{i=1}^{N} k(\mathbf{x}, \mathbf{x}_i) f_i \tag{4}$$

where $f_i = f(\mathbf{x}_i)$. Now, defining K as the positive definite kernel matrix with elements $K_{ij} = k(\mathbf{x}_i \mathbf{x}_j)$, we consider it's eigendecomposition $V \Lambda V^T$ where Λ is the diagonal matrix of (w.l.o.g sorted) eigenvalues $\lambda_1 \leq ... \leq \lambda_N$ and the columns of V are the corresponding eigenvectors. This yields:

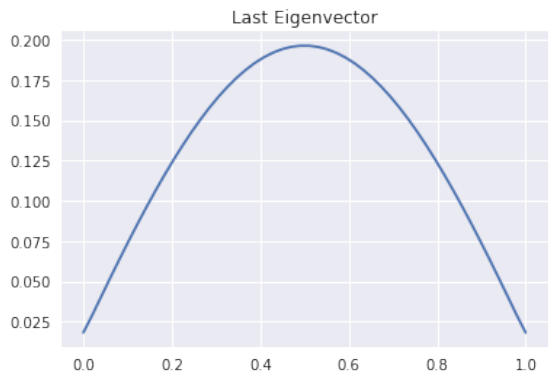
$$k(\mathbf{x}_i, \mathbf{x}_j) = K_{ij} = (V \Lambda V^T)_{ij} = \sum_{n=1}^{N} \lambda_n v_{ni} v_{nj}$$

Like in Braun et al. (2006), we define the *spectrum* $\tilde{f}[n]$ of the function f as:

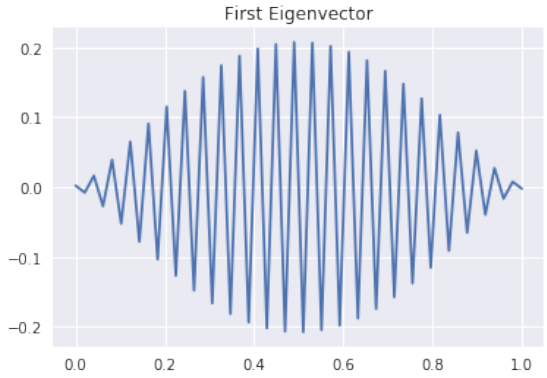
$$\tilde{f}[n] = \mathbf{f} \cdot \mathbf{v}_n \tag{5}$$

where $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_N))$. The value n can be thought of a generalized notion of *frequency*. Indeed, it is known Fasshauer (2011); Rasmussen (2004), for instance, that the eigenfunctions φ_n resemble sinusoids with increasing frequencies (for increasing n or decreasing λ_n). In

[1]We drop the subscript σ to simplify the notation.



(a) Eigenvector with the largest eigenvalue ($n = 1$).



(b) Eigenvector with the smallest eigenvalue ($n = 50$).

Fig. A.3. Two extreme eigenvectors of the Gaussian RBF kernel for 50 uniformly spaced samples between 0 and 1.

Figure A.3, we plot the eigenvectors $\mathbf{v}_0$ and $\mathbf{v}_N$ for $\{\mathbf{x}_i\}_{i=1}^{50}$ uniformly spaced between $[0, 1]$. Further, in Figure 3.6 we evaluate the discrete Fourier transform of all $N = 50$ eigenvectors, and find that the eigenfunction index n does indeed coincide with frequency k. Finally, we remark that the link between signal complexity and the spectrum is extensively studied in Braun et al. (2006).

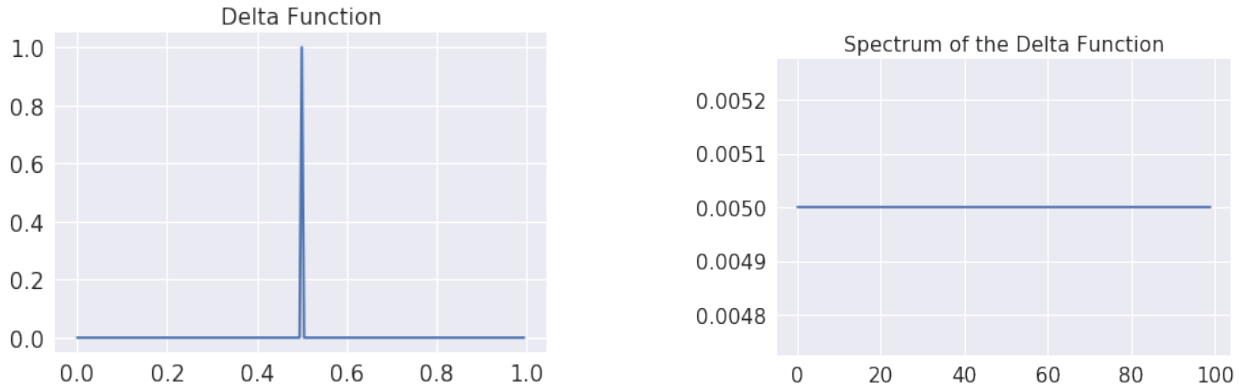
Fig. A.4. Loss curves for the Figure 3.5. We find that the validation loss dips at around the 200th iteration.

A.1.4.1. Loss Curves Accompanying Figure 3.5.

A.1.5. Qualitative Ablation over Architectures

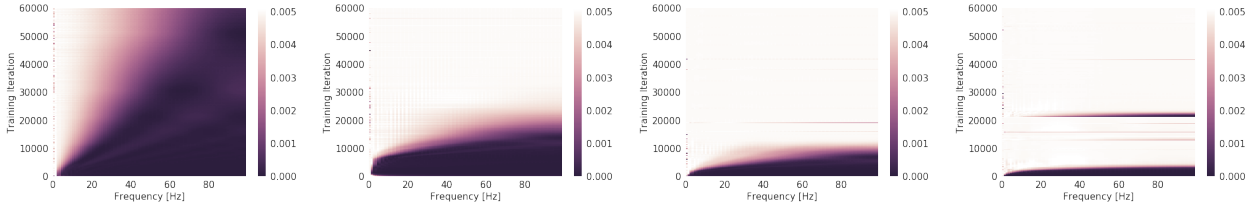
Theorem 3.3 exposes the relationship between the fourier spectrum of a network and its depth, width and max-norm of parameters. The following experiment is a qualitative ablation study over these variables.

Experiment 7. In this experiment, we fit various networks to the δ-function at $x = 0.5$ (see Fig A.5a). Its spectrum is constant for all frequencies (Fig A.5b), which makes it particularly useful for testing how well a given network can fit large frequencies. Fig A.8 shows the



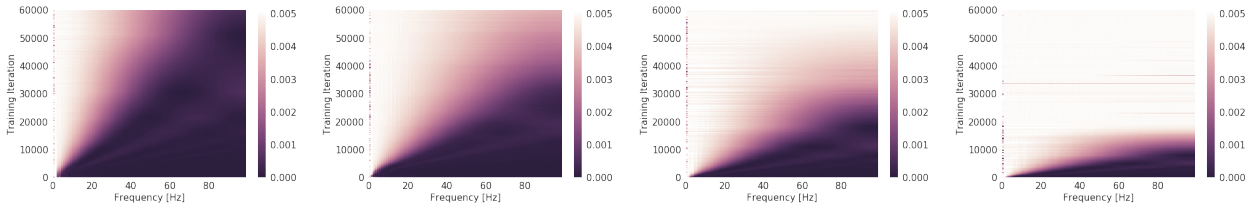
(a) Sampled δ-function at $x = 0.5$. **(b)** Constant Spectrum of the δ-function.

Fig. A.5. The target function used in Experiment 7.



(a) Depth = 3. **(b)** Depth = 4. **(c)** Depth = 5. **(d)** Depth = 6.

Fig. A.6. Evolution with training iterations (y-axis) of the Fourier spectrum (x-axis for frequency, and colormap for magnitude) for a network with **varying depth**, width = 16 and weight clip = 10. The spectrum of the target function is a constant 0.005 for all frequencies.



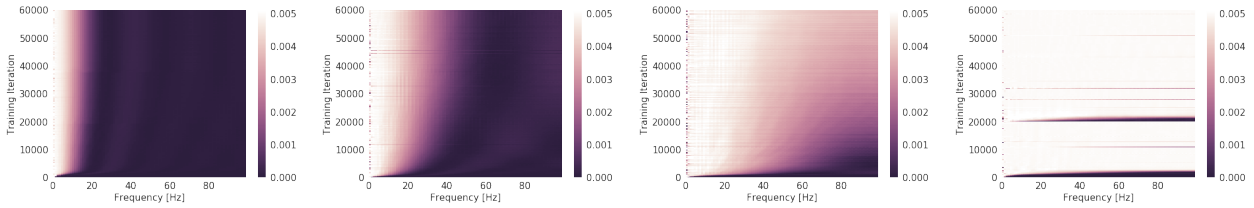
(a) Width = 16. **(b)** Width = 32. **(c)** Width = 64. **(d)** Width = 128.

Fig. A.7. Evolution with training iterations (y-axis) of the Fourier spectrum (x-axis for frequency, and colormap for magnitude) for a network with **varying width**, depth = 3 and weight clip = 10. The spectrum of the target function is a constant 0.005 for all frequencies.

ablation over weight clip (i.e. max parameter max-norm), Fig A.6 over depth and Fig A.7 over width. Fig A.9 exemplarily shows how the network prediction evolves with training iterations. All networks are trained for 60K iterations of full-batch gradient descent under identical conditions (Adam optimizer with $lr = 0.0003$, no weight decay).

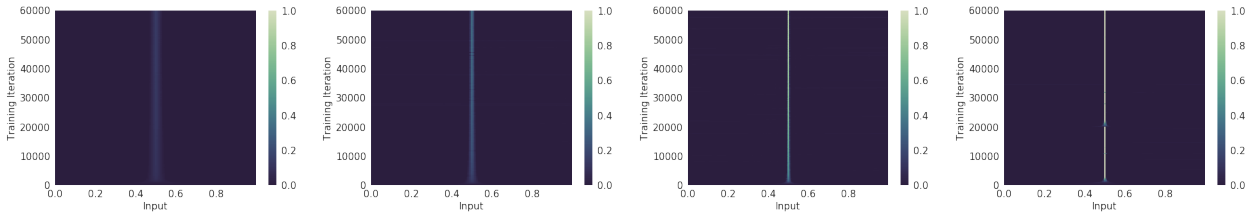
We make the following observations.

(a) Fig A.6 shows that increasing the depth (for fixed width) significantly improves the network's ability to fit higher frequencies (note that the depth increases linearly).



(a) Weight Clip = 0.1. **(b)** Weight Clip = 0.15. **(c)** Weight Clip = 0.2. **(d)** Weight Clip = 2.

Fig. A.8. Evolution with training iterations (y-axis) of the Fourier spectrum (x-axis for frequency, and colormap for magnitude) for a network with **varying weight clip**, depth = 6 and width = 64. The spectrum of the target function is a constant 0.005 for all frequencies.



(a) Weight Clip = 0.1. **(b)** Weight Clip = 0.15. **(c)** Weight Clip = 0.2. **(d)** Weight Clip = 2.

Fig. A.9. Evolution with training iterations (y-axis) of the network prediction (x-axis for input, and colormap for predicted value) for a network with **varying weight clip**, depth = 6 and width = 64. The target function is a δ peak at $x = 0.5$.

(b) Fig A.7 shows that increasing the width (for fixed depth) also helps, but the effect is considerably weaker (note that the width increases exponentially).

(c) Fig A.8 shows that increasing the weight clip (or the max parameter max-norm) also helps the network fit higher frequencies.

The above observations are all consistent with Theorem 3.3, and further show that lower frequencies are learned first (i.e. the spectral bias, cf. Experiment 1). Further, Figure A.8 shows that constraining the Lipschitz constant (weight clip) prevents the network from learning higher frequencies, furnishing evidence that the $\mathcal{O}(L_f)$ bound can be tight.

A.1.6. MNIST: A Proof of Concept

In the following experiment, we show that given two manifolds of the same dimension – one flat and the other not – the task of learning random labels is harder to solve if the input samples lie on the same manifold. We demonstrate on MNIST under the assumption that the manifold hypothesis is true, and use the fact that the spectrum of the target function we use (white noise) is constant in expectation, and therefore independent of the underlying coordinate system when defined on the manifold.

Experiment 8. In this experiment, we investigate if it is easier to learn a signal on a more realistic data-manifold like that of MNIST (assuming the manifold hypothesis is true), and

compare with a flat manifold of the same dimension. To that end, we use the 64-dimensional feature-space $\mathcal{E}$ of a denoising[2] autoencoder as a proxy for the real data-manifold of unknown number of dimensions. The decoder functions as an embedding of $\mathcal{E}$ in the input space $X = \mathbb{R}^{784}$, which effectively amounts to training a network on the reconstructions of the autoencoder. For comparision, we use an injective embedding[3] of a 64-dimensional hyperplane in X. The latter is equivalent to sampling 784-dimensional vectors from $U([0, 1])$ and setting all but the first 64 components to zero. The target function is white-noise, sampled as scalars from the uniform distribution $U([0, 1])$. Two identical networks are trained under identical conditions, and Fig A.10 shows the resulting loss curves, each averaged over 10 runs.

This result complements the findings of Arpit et al. (2017) and Zhang et al. (2017a), which show that it's easier to fit random labels to random inputs if the latter is defined on the full dimensional input space (i.e. the dimension of the flat manifold is the same as that of the input space, and not that of the underlying data-manifold being used for comparison).

Fig. A.10. Loss curves of two identical networks trained to regress white-noise under identical conditions, one on MNIST reconstructions from a DAE with 64 encoder features (blue), and the other on 64-dimensional random vectors (green).

A.1.7. Cifar-10: It's All Connected

We have seen that deep neural networks are biased towards learning low frequency functions. This should have as a consequence that isolated *bubbles* of constant prediction are rare. This in turn implies that given any two points in the input space and a network function that predicts the same class for the said points, there should be a path connecting them such that the network prediction does not change along the path. In the following, we present an experiment where we use a path finding method to find such a path between all Cifar-10 input samples indeed exist.

[2]This experiment yields the same result if variational autoencoders are used instead.
[3]The xy-plane is $\mathbb{R}^3$ an injective embedding of a subset of $\mathbb{R}^2$ in $\mathbb{R}^3$.

Fig. A.11. Path between CIFAR-10 adversarial examples (e.g. "frog" and "automobile", such that all images are classified as "airplane").

Experiment 9. Using AutoNEB Kolsbjerg et al. (2016), we construct paths between (adversarial) Cifar-10 images that are classified by a ResNet20 to be all of the same target class. AutoNEB bends a linear path between points in some space $\mathbb{R}^m$ so that some maximum energy along the path is minimal. Here, the space is the input space of the neural network, i.e. the space of $32 \times 32 \times 3$ images and the logit output of the ResNet20 for a given class is minimized. We construct paths between the following points in image space:

- From one training image to another,
- from a training image to an adversarial,
- from one adversarial to another.

We only consider pairs of images that belong to the same class c (or, for adversarials, that originate from another class $\neq c$, but that the model classifies to be of the specified class c). For each class, we randomly select 50 training images and select a total of 50 random images from all other classes and generate adversarial samples from the latter. Then, paths between all pairs from the whole set of images are computed.

The AutoNEB parameters are chosen as follows: We run four NEB iterations with 10 steps of SGD with learning rate 0.001 and momentum 0.9. This computational budget is similar to that required to compute the adversarial samples. The gradient for each NEB step is computed to maximize the logit output of the ResNet-20 for the specified target class c. We use the formulation of NEB without springs Draxler et al. (2018).

The result is very clear: We can find paths between *all* pairs of images for all CIFAR10 labels that do not cross a single decision boundary. This means that all paths belong to the same connected component regarding the output of the DNN. This holds for all possible

Fig. A.12. Each row is a path through the image space from an adversarial sample (right) to a true training image (left). All images are classified by a ResNet-20 to be of the class of the training sample on the right with at least 95% softmax certainty. This experiment shows we can find a path from adversarial examples (right, Eg. "(cat)") that are classified as a particular class ("airplane") are connected to actual training samples from that class (left, "airplane") such that all samples along that path are also predicted by the network to be of the same class.

combinations of images in the above list. Figure A.12 shows connecting training to adversarial images and Figure A.11 paths between pairs of adversarial images. Paths between training images are not shown, they provide no further insight. Note that the paths are strikingly simple: Visually, they are hard to distinguish from the linear interpolation. Quantitatively, they are essentially (but not exactly) linear, with an average length $(3.0 \pm 0.3)\%$ longer than the linear connection.

A.2. The Continuous Piecewise Linear Structure of Deep ReLU Networks

We consider the class of ReLU network functions $f : \mathbb{R}^d \mapsto \mathbb{R}$ defined by Eqn. 1. Following the terminology of Raghu et al. (2016); Montufar et al. (2014), each linear region of the network then corresponds to a unique *activation pattern*, wherein each hidden neuron is assigned an activation variable $\epsilon \in \{-1, 1\}$, conditioned on whether its input is positive or negative. ReLU networks can be explictly expressed as a sum over all possible activation patterns, as in the following lemma.

Lemma A.1. *Given L binary vectors $\epsilon^{(1)}, \cdots \epsilon^{(L)}$ with $\epsilon^{(k)} \in \{-1, 1\}^{d_k}$, let $T^{(k)}_{\epsilon^{(k)}} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ the affine function defined by $T^{(k)}_{\epsilon^{(k)}}(\mathbf{u})_i = (T^{(k)}(\mathbf{u}))_i$ if $(\epsilon_k)_i = 1$, and 0 otherwise. ReLU*

network functions, as defined in Eqn. 1, can be expressed as

$$f(\mathbf{x}) = \sum_{\epsilon^{(1)},\cdots\epsilon^{(L)}} 1_{P_{f,\epsilon}}(\mathbf{x}) \left(T^{(L+1)} \circ T^{(L)}_{\epsilon^{(L)}} \circ \cdots \circ T^{(1)}_{\epsilon^{(1)}}\right)(\mathbf{x}) \tag{6}$$

where 1_P denotes the indicator function of the subset $P \subset \mathbb{R}^d$, and $P_{f,\epsilon}$ is the polytope defined as the set of solutions of the following linear inequalities (for all $k = 1, \cdots, L$):

$$(\epsilon_k)_i \left(T^{(k)} \circ T^{(k-1)}_{\epsilon^{(k-1)}} \circ \cdots \circ T^{(1)}_{\epsilon^{(1)}}\right)(\mathbf{x})_i \geq 0, \quad i = 1, \cdots d_k \tag{7}$$

f is therefore affine on each of the polytopes $P_{f,\epsilon}$, which finitely partition the input space $\mathbb{R}^d$ to convex polytopes. Remarkably, the correspondence between ReLU networks and CPWL functions goes both ways: Arora et al. (2018a) show that every CPWL function can be represented by a ReLU network, which in turn endows ReLU networks with the universal approximation property.

Finally, in the standard basis, each affine map $T^{(k)} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ is specified by a weight matrix $W^{(k)} \in \mathbb{R}^{d_{k-1}} \times \mathbb{R}^{d_k}$ and a bias vector $b^{(k)} \in \mathbb{R}^{d_k}$. In the linear region $P_{f,\epsilon}$, f can be expressed as $f_\epsilon(x) = W_\epsilon x + b_\epsilon$, where in particular

$$W_\epsilon = W^{(L+1)} W^{(L)}_{\epsilon_L} \cdots W^{(1)}_{\epsilon_1} \in \mathbb{R}^{1 \times d}, \tag{8}$$

where $W^{(k)}_\epsilon$ is obtained from $W^{(k)}$ by setting its jth column to zero whenever $(\epsilon_k)_j = -1$.

A.3. Fourier Analysis of ReLU Networks

A.3.1. Proof of Lemma 3.1

Proof. **Case 1: The function f has compact support.** The vector-valued function $\mathbf{k} f(\mathbf{x}) e^{i\mathbf{k}\cdot\mathbf{x}}$ is continuous everywhere and has well-defined and continuous gradients almost everywhere. So by Stokes' theorem (see e.g Spivak (2018)), the integral of its divergence is a pure boundary term. Since we restricted to functions with compact support, the theorem yields

$$\int \nabla_{\mathbf{x}} \cdot \left[\mathbf{k} f(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}}\right] \mathbf{dx} = 0 \tag{9}$$

The integrand is $(\mathbf{k} \cdot (\nabla_{\mathbf{x}} f)(\mathbf{x}) - ik^2 f(\mathbf{x})) e^{-i\mathbf{k}\cdot\mathbf{x}}$, so we deduce,

$$\hat{f}(\mathbf{k}) = \frac{1}{-ik^2} \mathbf{k} \cdot \int (\nabla_{\mathbf{x}} f)(\mathbf{x})\, e^{-i\mathbf{k}\cdot\mathbf{x}} \tag{10}$$

Now, within each polytope of the decomposition (6), f is affine so its gradient is a constant vector, $\nabla_{\mathbf{x}} f_\epsilon = W_\epsilon^T$, which gives the desired result (3.1).

Case 2: The function f does not have compact support. Without the assumption of compact support, the function f is not squared-integrable. The Fourier transform therefore only exists in the sense of distributions, as defined below.

Let $\mathcal{S}$ be the Schwartz space over $\mathbb{R}^d$ of rapidly decaying test functions which together with its derivatives decay to zero as $x \to \infty$ faster than any power of x. A tempered distribution is a continuous linear functional on $\mathcal{S}$. A function f that doesn't grow faster than a polynomial at infinity can be identified with a tempered distribution T_f as:

$$T_f : \mathcal{S} \to \mathbb{R},\ \varphi \mapsto \langle f, \varphi \rangle = \int_{\mathbb{R}^d} f(\mathbf{x})\varphi(\mathbf{x})\mathbf{dx} \tag{11}$$

In the following, we shall identify T_f with f. The Fourier transform $\tilde{f}$ of the tempered distribution is defined as:

$$\langle \tilde{f}, \varphi \rangle := \langle f, \tilde{\varphi} \rangle \tag{12}$$

where $\tilde{\varphi}$ is the Fourier transform of φ. In this sense, the standard notion of the Fourier transform is generalized to functions that are not squared-integrable.

Consider the continuous piecewise-linear ReLU network $f : \mathbb{R}^d \to \mathbb{R}$. Since it can grow at most linearly, we interpret it as a tempered distribution on $\mathbb{R}^d$. Recall that the linear regions P_ϵ are enumerated by ϵ. Let f_ϵ be the restriction of f to P_ϵ, making $f_\epsilon(\mathbf{x}) = W_\epsilon^T \mathbf{x}$. The distributional derivative of f is given by:

$$\nabla_{\mathbf{x}} f = \sum_\epsilon \nabla_{\mathbf{x}} f_\epsilon \cdot 1_{P_\epsilon} = \sum_\epsilon W_\epsilon^T 1_{P_\epsilon} \tag{13}$$

where 1_{P_ϵ} is the indicator over P_ϵ and we used $\nabla_{\mathbf{x}} f_\epsilon = W_\epsilon^T$. It then follows from elementary properties of Schwartz spaces (see e.g. Chapter 16 of Serov (2017)) that:

$$[\widetilde{\nabla_{\mathbf{x}} f}](\mathbf{k}) = -i\mathbf{k}\tilde{f}(\mathbf{k}) \tag{14}$$

$$\implies \tilde{f}(\mathbf{k}) = \tfrac{1}{-ik^2}\mathbf{k} \cdot [\widetilde{\nabla_{\mathbf{x}} f}](\mathbf{k}) \tag{15}$$

Together with Eqn 13 and linearity of the Fourier transform, this gives the desired result (3.1).

□

A.3.2. Fourier Transform of Polytopes

A.3.2.1. Theorem 1 of Diaz et al. (2016). Let F be a m dimensional polytope in $\mathbb{R}^d$, such that $1 \leq m \leq d$. Denote by $\mathbf{k} \in \mathbb{R}^d$ a vector in the Fourier space, by $\phi_{\mathbf{k}}(x) = -\mathbf{k} \cdot \mathbf{x}$ the linear phase function, by $\tilde{F}$ the Fourier transform of the indicator function on F, by ∂F the boundary of F and by vol_m the m-dimensional (Hausdorff) measure. Let $\text{Proj}_F(\mathbf{k})$ be the

orthogonal projection of $\mathbf{k}$ on to F (obtained by removing all components of $\mathbf{k}$ orthogonal to F). Given a $m-1$ dimensional facet G of F, let $\mathbf{N}_F(G)$ be the unit normal vector to G that points out of F. It then holds:

1. If $\text{Proj}_F(\mathbf{k}) = 0$, then $\phi_{\mathbf{k}}(x) = \Phi_{\mathbf{k}}$ is constant on F, and we have:

$$\tilde{F} = \text{vol}_F(F)e^{i\Phi_{\mathbf{k}}} \tag{16}$$

2. But if $\text{Proj}_F(\mathbf{k}) \neq 0$, then:

$$\tilde{F} = i \sum_{G \in \partial F} \frac{\text{Proj}_F(\mathbf{k}) \cdot \mathbf{N}_F(G)}{\|\text{Proj}_F(\mathbf{k})\|^2} \tilde{G}(\mathbf{k}) \tag{17}$$

A.3.2.2. Discussion. The above theorem provides a recursive relation for computing the Fourier transform of an arbitrary polytope. More precisely, the Fourier transform of a m-dimensional polytope is expressed as a sum of fourier transforms over the $m-1$ dimensional boundaries of the said polytope (which are themselves polytopes) times a $\mathcal{O}(k^{-1})$ *weight* term (with $k = \|\mathbf{k}\|$). The recursion terminates if $\text{Proj}_F(\mathbf{k}) = 0$, which then yields a constant.

To structure this computation, Diaz et al. (2016) introduce a book-keeping device called the *face poset* of the polytope. It can be understood as a weighted directed acyclic graph (DAG) with polytopes of various dimensions as its nodes. We start at the root node which is the full dimensional polytope P (i.e. we initially set $m = n$). For all of the codimension-one boundary faces F of P, we then draw an edge from the root P to node F and weight it with a term given by:

$$W_{F,G} = i\frac{\text{Proj}_F(\mathbf{k}) \cdot \mathbf{N}_F(G)}{\|\text{Proj}_F(\mathbf{k})\|^2} \tag{18}$$

and repeat the process iteratively for each F. Note that the weight term is $\mathcal{O}(k^{-1})$ where $\text{Proj}_F(\mathbf{k}) \neq 0$. This process yields tree paths $T : F_0 = P \to F_1 \to ... \to F_{|T|}$ where each $F_{i+1} \in \partial F_i$ has one dimension less than F_i. For a given path and $\mathbf{k}$, the terminal node for this path, F_{n_T}, is the first polytope for which $\text{Proj}_{F_{n_T}}(\mathbf{k}) = 0$. The final Fourier transform is obtained by multiplying the weights along each path and summing over all tree paths:

$$\tilde{1}_P(\mathbf{k}) = \sum_T \prod_{i=0}^{|T|-1} W_{F_i,F_{i+1}} \text{vol}_{F_{|T|}}(F_{|T|}) e^{i\Phi_{\mathbf{k}}} \tag{19}$$

where $\Phi^{(T)} = \mathbf{k} \cdot \mathbf{x}_0^T$ for an arbitrary point $\mathbf{x}_0^T$ in $F_{|T|}$.

To write this as a weighted sum of indicator functions, as in Lemma 3.2, let $\mathcal{T}_n$ denote the set of all tree paths T of length n, i.e. $|T| = n$. For a tree path T, let $S(T)$ be the orthogonal to the terminal node F_n, i.e the vectors $\mathbf{k}$ such that $\text{Proj}_{F_n}(\mathbf{k}) = 0$. The sum over T in Eqn

(19) can be split as:

$$\tilde{1}_P = \sum_{n=0}^{d} \frac{1_{G_n}}{k^n} \sum_{T \in \mathcal{T}_n} 1_{S(T)} \prod_{i=0}^{n-1} \bar{W}_{F_i^T, F_{i+1}^T} \mathrm{vol}_{F_n^T}(F_n^T) e^{i\Phi_{\mathbf{k}}^{(T)}} \tag{20}$$

where $\bar{W}_{F,G} = kW_{F,G}$ and $G_n = \bigcup_{T \in \mathcal{T}_n} S(T)$. In words, G_n is the set of all vectors $\mathbf{k}$ that are orthogonal to some n-codimensional face of the polytope. We identify:

$$D_q = \sum_{T \in \mathcal{T}_n} 1_{S(T)} \prod_{i=0}^{n-1} \bar{W}_{F_i^T, F_{i+1}^T} \mathrm{vol}_{F_n^T}(F_n^T) e^{i\Phi_{\mathbf{k}}^{(T)}} \tag{21}$$

and $D_0(\mathbf{k}) = \mathrm{vol}(P)$ to obtain Lemma 3.2. Observe that D_n depends on k only via the phase term $e^{i\Phi_{\mathbf{k}}^{(T)}}$, implying that $D_n = \Theta(1)\,(k \to \infty)$.

Informally, for a generic vector $\mathbf{k}$, all paths terminate at the zero-dimensional vertices of the original polytope, i.e. $\dim(F_n) = 0$, implying the length of the path n equals the number of dimensions d, yielding a $\mathcal{O}(k^{-d})$ spectrum. The exceptions occur if a path terminates prematurely, because $\mathbf{k}$ happens to lie orthogonal to some $d-r$-dimensional face F_r in the path, in which case we are left with a $\mathcal{O}(k^{-r})$ term (with $r < d$) which dominates asymptotically. Note that all vectors orthogonal to the $d - r$ dimensional face F_r lie on a r-dimensional subspace of $\mathbb{R}^d$. Since a polytope has a finite number of faces (of any dimension), the $\mathbf{k}$'s for which the Fourier transform is $\mathcal{O}(k^{-r})$ (instead of $\mathcal{O}(k^{-d})$) lies on a finite union of closed subspaces of dimension r (with $r < d$). The Lebesgue measure of all such lower dimensional subspaces for all such r is 0, leading us to the conclusion that the spectrum decays as $\mathcal{O}(k^{-d})$ for *almost all* $\mathbf{k}$'s.

A.3.3. On Theorem 3.3

Equation 6 can be obtained by swapping the (finite) sum over ϵ in Lemma 3.1 with that over the paths T in Eqn 20. In particular, we have:

$$\tilde{f} = \sum_{n=0}^{d} \frac{1_{H_n}}{k^{n+1}} \sum_{\epsilon} W_\epsilon D_n^\epsilon 1_{G_n^\epsilon} \tag{22}$$

Now, the sum $\sum_\epsilon W_\epsilon D_n^\epsilon(\hat{\mathbf{k}}) I_{G_n^\epsilon}(\mathbf{k})$ is supported on the union:

$$H_n = \bigcup_{\epsilon} G_n^\epsilon \tag{23}$$

Identifying:

$$C_n(\cdot, \theta) = \sum_{\epsilon} W_\epsilon D_n^\epsilon 1_{G_n^\epsilon} \tag{24}$$

where $C_n(\cdot, \theta) = \mathcal{O}(1)\,(k \to \infty)$, we obtain Theorem 3.3. Further, if N_f is the number of linear regions of the network and $L_f = \max_\epsilon \|W_\epsilon\|$, we see that $C_n = \mathcal{O}(L_f N_f)$. Indeed, in Appendix A.1.5, we empirically find that relaxing the constraint on the weight clip (which can be identified with L_f) enabled the network to fit higher frequencies, implying that the $\mathcal{O}(L_f)$ bound can be tight.

A.3.4. Spectral Decay Rate of the Parameter Gradient

Proposition A.2. *Let θ be a generic parameter of the network function f. The spectral decay rate of $\partial \tilde{f}/\partial\theta$ is $\mathcal{O}(k\tilde{f})$.*

Proof. For a fixed $\hat{\mathbf{k}}$, observe from Eqn 22 and Eqn 21 that the only terms dependent on k are the pure powers k^{-n-1} and the phase terms $e^{i\Phi_{\mathbf{k}}^{(T)}}$, where $\Phi_{\mathbf{k}}^{(T)} = k\hat{\mathbf{k}} \cdot \mathbf{x}_0^{q(T)}$. However, the term $\mathbf{x}_0^{q(T)}$ is in general a function of θ, and consequently the partial derivative of $e^{i\Phi_{\mathbf{k}}^{(T)}}$ w.r.t θ yields a term that is proportional to k. This term now dominates the asymptotic behaviour as $k \to \infty$, adding an extra power of k to the total spectral decay rate of $\tilde{f}$. □

Therefore, if $f = \mathcal{O}(k^{-\Delta-1})$ where Δ is the codimension of the highest dimensional polytope $\hat{\mathbf{k}}$ is orthogonal to, we have that $\partial f/\partial\theta = \mathcal{O}(k^{-\Delta})$.

A.3.5. Convergence Rate of a Network Trained on Pure-Frequency Targets

In this section, we derive an asymptotic bound on the convergence rate under the assumption that the target function has only one frequency component.

Proposition A.3. *Let $\lambda : [0, 1] \to \mathbb{R}$ be a target function sampled in its domain at N uniformly spaced points. Suppose that its Fourier transform after sampling takes the form: $\tilde{\lambda}(k) = A_0\delta_{k,k_0}$, where δ is the Kronecker delta. Let f be a neural network trained with full-batch gradient descent with learning rate η on the Mean Squared Error, and denote by f_t the state of the network at time t. Let $h(\cdot, t) = f_t - \lambda$ be the residual at time t. We have that:*

$$\left|\frac{\partial \tilde{h}(k_0, t)}{\partial t}\right| = \mathcal{O}(k_0^{-1}) \tag{25}$$

Proof. Consider that:

$$\left|\frac{\partial \tilde{h}(k_0)}{\partial t}\right| = \left|\tfrac{\partial \tilde{f}(k_0)}{\partial \theta}\right| \left|\tfrac{\partial \theta}{\partial t}\right| \tag{26}$$

$$= \left|\eta \tfrac{\partial \tilde{f}}{\partial \theta}\right| \left|\tfrac{\partial \mathcal{L}[\tilde{f}, \tilde{\lambda}]}{\partial \theta}\right| \tag{27}$$

where $\mathcal{L}$ is the sampled MSE loss and the first term is $\mathcal{O}(k_0^{-1})$ as can be seen from Proposition A.2. With Parceval's Theorem, we obtain:

$$\begin{aligned}\mathcal{L}[f,\lambda] &= \textstyle\sum_{x=0}^{N-1} |f(x)-\lambda(x)|^2 = \sum_{k=-N/2}^{N/2-1} |\tilde{f}(k)-\tilde{\lambda}(k)|^2 \\ &= \mathcal{L}[\tilde{f},\tilde{\lambda}] \end{aligned} \tag{28}$$

For the magnitude of parameter gradient, we obtain:

$$\begin{aligned}\left|\frac{\partial \mathcal{L}[\tilde{f},\tilde{\lambda}]}{\partial\theta}\right| &= 2\left|\textstyle\sum_{k=-N/2}^{N/2-1} \mathrm{Re}[\tilde{f}(k)-\tilde{\lambda}(k)]\frac{\partial\tilde{f}(k)}{\partial\theta}\right| \\ &\leq 2\textstyle\sum_{k=-N/2}^{N/2-1} |\tilde{f}(k)-\tilde{\lambda}(k)|\left|\frac{\partial\tilde{f}(k)}{\partial\theta}\right| \\ &\leq 2\left|A_0\frac{\partial\tilde{f}(k_0)}{\partial\theta}\right| + 2\textstyle\sum_{k=-N/2}^{N/2-1}\left|\tilde{f}(k)\frac{\partial\tilde{f}(k)}{\partial\theta}\right| \end{aligned} \tag{29}$$

where in the last line we used that $\tilde{\lambda}$ is a Kronecker-δ in the Fourier domain. Now, the second summand does not depend on k_0, but the first summand is again $\mathcal{O}(k_0^{-1})$. □

A.3.6. Proof of the Lipschtiz bound

Proposition A.4. *The Lipschitz constant L_f of the ReLU network f is bound as follows (for all ϵ):*

$$\|W_\epsilon\| \leq L_f \leq \prod_{k=1}^{L+1} \|W^{(k)}\| \leq \|\theta\|_\infty^{L+1}\sqrt{d}\prod_{k=1}^{L} d_k \tag{30}$$

Proof. The first equality is simply the fact that $L_f = \max_\epsilon \|W_\epsilon\|$, and the second inequality follows trivially from the parameterization of a ReLU network as a chain of function compositions[4], together with the fact that the Lipschitz constant of the ReLU function is 1 (cf. Miyato et al. (2019), equation 7). To see the third inequality, consider the definition of the spectral norm of a $I \times J$ matrix W:

$$\|W\| = \max_{\|\mathbf{h}\|=1} \|W\mathbf{h}\| \tag{31}$$

Now, $\|W\mathbf{h}\| = \sqrt{\sum_i |\mathbf{w}_i \cdot \mathbf{h}|}$, where $\mathbf{w}_i$ is the i-th row of the weight matrix W and $i = 1, ..., I$. Further, if $\|\mathbf{h}\| = 1$, we have $|\mathbf{w}_i \cdot \mathbf{h}| \leq \|\mathbf{w}_i\|\|\mathbf{h}\| = \|\mathbf{w}_i\|$. Since $\|\mathbf{w}_i\| = \sqrt{\sum_j |w_{ij}|}$ (with $j = 1, ..., J$) and $|w_{ij}| \leq \|\theta\|_\infty$, we find that $\|\mathbf{w}_i\| \leq \sqrt{J}\|\theta\|_\infty$. Consequently, $\sqrt{\sum_i |\mathbf{w}_i \cdot \mathbf{h}|} \leq \sqrt{IJ}\|\theta\|_\infty$ and we obtain:

$$\|W\| \leq \sqrt{IJ}\|\theta\|_\infty \tag{32}$$

[4]Recall that the Lipschitz constant of a composition of two or more functions is the product of their respective Lipschtiz constants.

Now for $W = W^{(k)}$, we have $I = d_{k-1}$ and $J = d_k$. In the product over k, every d_k except the first and the last occur in pairs, which cancels the square root. For $k = 1$, $d_{k-1} = d$ (for the d input neurons) and for $k = L + 1$, $d_k = 1$ (for a single output neuron). The final inequality now follows. □

A.3.7. The Fourier Transform of a Function Composition

Consider Equation 14. The general idea is to investigate the behaviour of $P_\gamma(\mathbf{l}, \mathbf{k})$ for large frequencies $\mathbf{l}$ on manifold but smaller frequencies $\mathbf{k}$ in the input domain. In particular, we are interested in the regime where the stationary phase approximation is applicable to P_γ, i.e. when $l^2 + k^2 \to \infty$ (cf. section 3.2. of Bergner et al.). In this regime, the integrand in $P_\gamma(\mathbf{k}, \mathbf{l})$ oscillates fast enough such that the only constructive contribution originates from where the phase term $u(\mathbf{z}) = \mathbf{k} \cdot \gamma(\mathbf{z}) - \mathbf{l} \cdot \mathbf{z}$ does not change with changing $\mathbf{z}$. This yields the condition that $\nabla_{\mathbf{z}} u(\mathbf{z}) = 0$, which translates to the condition (with Einstein summation convention implied and $\partial_\nu = {}^{\partial}/_{\partial x_\nu}$):

$$l_\nu = k_\mu \partial_\nu \gamma_\mu(\mathbf{z}) \tag{33}$$

Now, we impose periodic boundary conditions[5] on the components of γ, and without loss of generality we let the period be 2π. Further, we require that the manifold be contained in a box[6] of some size in $\mathbb{R}^d$. The μ-th component γ_μ can now be expressed as a Fourier series:

$$\begin{aligned} \gamma_\mu(\mathbf{z}) &= \textstyle\sum_{\mathbf{p} \in \mathbb{Z}^m} \tilde{\gamma}_\mu[\mathbf{p}] e^{-i p_\rho z_\rho} \\ \partial_\nu \gamma_\mu(\mathbf{z}) &= \textstyle\sum_{\mathbf{p} \in \mathbb{Z}^m} -i p_\nu \tilde{\gamma}_\mu[\mathbf{p}] e^{-i p_\rho z_\rho} \end{aligned} \tag{34}$$

Equation 34 can be substituted in equation 33 to obtain:

$$l \hat{l}_\nu = -ik \sum_{\mathbf{p} \in \mathbb{Z}^m} p_\nu \hat{k}_\mu \tilde{\gamma}_\mu[\mathbf{p}] e^{-i p_\rho z_\rho} \tag{35}$$

where we have split k_μ and l_ν in to their magnitudes k and l and directions $\hat{k}_\nu$ and $\hat{l}_\mu$ (respectively). We are now interested in the conditions on γ under which the RHS can be large in magnitude, even when k is fixed. Recall that γ is constrained to a box – consequently, we can not arbitrarily scale up $\tilde{\gamma}_\mu$. However, if $\tilde{\gamma}_\mu[\mathbf{p}]$ decays slowly enough with increasing $\mathbf{p}$, the RHS can be made arbitrarily large (for certain conditions on $\mathbf{z}$, $\hat{l}_\mu$ and $\hat{k}_\nu$).

[5] This is possible whenever γ is defined on a bounded domain, e.g. on $[0, 1]^m$.
[6] This is equivalent to assuming that the data lies in a bounded set.

A.4. Volume of *High-Frequency Parameters* in Parameter Space

For a given neural network, we now show that the volume of the parameter space containing parameters that contribute ϵ-non-negligibly to frequency components of magnitude k' above a certain cut-off k decays with increasing k. For notational simplicity and without loss of generality, we absorb the direction $\hat{\mathbf{k}}$ of $\mathbf{k}$ in the respective mappings and only deal with the magnitude k.

Definition A.1. *Given a ReLU network f_θ of fixed depth, width and weight clip K with parameter vector θ, an $\epsilon > 0$ and $\Theta = B_K^\infty(0)$ a L^∞ ball around 0, we define:*

$$\Xi_\epsilon(k) = \{\theta \in \Theta | \exists k' > k, |\tilde{f}_\theta(k')| > \epsilon\}$$

as the set of all parameters vectors $\theta \in \Xi_\epsilon(k)$ that contribute more than an ϵ in expressing one or more frequencies k' above a cut-off frequency k.

Remark 1. *If $k_2 \geq k_1$, we have $\Xi_\epsilon(k_2) \subseteq \Xi_\epsilon(k_1)$ and consequently $vol(\Xi_\epsilon(k_2)) \leq vol(\Xi_\epsilon(k_1))$, where vol is the Lebesgue measure.*

Lemma A.5. *Let $1_k^\epsilon(\theta)$ be the indicator function on $\Xi_\epsilon(k)$. Then:*

$$\exists\, \kappa > 0 : \forall k \geq \kappa, 1_k^\epsilon(\theta) = 0$$

Proof. From theorem 3.3, we know that[7] $|\tilde{f}_\theta(k)| = \mathcal{O}(k^{-\Delta-1})$ for an integer $1 \leq \Delta \leq d$. In the worse case where $\Delta = 1$, we have that $\exists M < \infty : |\tilde{f}_\theta(k)| < \frac{M}{k^2}$. Now, simply select a $\kappa > \sqrt{\frac{M}{\epsilon}}$ such that $\frac{M}{\kappa^2} < \epsilon$. This yields that $|\tilde{f}_\theta(\kappa)| < \frac{M}{\kappa^2} < \epsilon$, and given that $\frac{M}{\kappa^2} \leq \frac{M}{k^2}\, \forall\, k \geq \kappa$, we find $|\tilde{f}_\theta(k)| < \epsilon\, \forall\, k \geq \kappa$. Now by definition A.1, $\theta \notin \Xi_\epsilon(\kappa)$, and since $\Xi_\epsilon(k) \subseteq \Xi_\epsilon(\kappa)$ (see remark 1), we have $\theta \notin \Xi_\epsilon(k)$, implying $1_k^\epsilon(\theta) = 0\, \forall\, k \geq \kappa$. □

Remark 2. *We have $1_k^\epsilon(\theta) \leq |\tilde{f}_\theta(k)|$ for large enough k (i.e. for $k \geq \kappa$), since $|\tilde{f}_\theta(k)| \geq 0$.*

Proposition A.6. *The relative volume of $\Xi_\epsilon(k)$ w.r.t. Θ is $\mathcal{O}(k^{-\Delta-1})$ where $1 \leq \Delta \leq d$.*

Proof. The volume is given by the integral over the indicator function, i.e.

$$\text{vol}(\Xi_\epsilon(k)) = \int_{\theta \in \Theta} 1_k^\epsilon(\theta) d\theta$$

For a large enough k, we have from remark 2, the monotonicity of the Lebesgue integral and theorem 3.3 that:

[7] Note from Theorem 3.3 that Δ implicitly depends only on the unit vector $\hat{\mathbf{k}}$.

$$\text{vol}(\Xi_\epsilon(k)) = \int_{\theta\in\Theta} 1_k^\epsilon(\theta)d\theta \tag{36}$$

$$\leq \int_{\theta\in\Theta} |\tilde{f}_\theta(k)|d\theta = \mathcal{O}(k^{-\Delta-1})\text{vol}(\Theta) \tag{37}$$

$$\implies \frac{\text{vol}(\Xi_\epsilon(k))}{\text{vol}(\Theta)} = \mathcal{O}(k^{-\Delta-1}) \tag{38}$$

□

A.5. Kernel Machines and KNNs

In this section, in light of our findings, we want to compare DNNs with K-nearest neighbor (k-NN) classifier and kernel machines which are also popular learning algorithms, but are, in contrast to DNNs, better understood theoretically.

A.5.1. Kernel Machines vs DNNs

Given that we study why DNNs are biased towards learning smooth functions, we note that kernel machines (KM) are also highly Lipschitz smooth (Eg. for Gaussian kernels all derivatives are bounded). However there are crutial differences between the two. While kernel machines can approximate any target function in principal (Hammer & Gersmann, 2003), the number of Gaussian kernels needed scales linearly with the number of sign changes in the target function (Bengio et al., 2009). Ma & Belkin (2017) have further shown that for smooth kernels, a target function cannot be approximated within ϵ precision in any polynomial of $1/\epsilon$ steps by gradient descent.

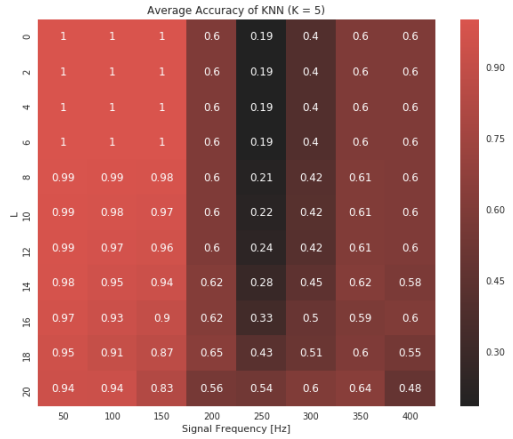
Deep networks on the other hand are also capable of approximating any target function (as shown by the universal approximation theorems Hornik et al. (1989); Cybenko (1989)), but they are also parameter efficient in contrast to KM. For instance, we have seen that deep ReLU networks separate the input space into number of linear regions that grow polynomially in width of layers and exponentially in the depth of the network (Montufar et al., 2014; Raghu et al., 2016). A similar result on the exponentially growing expressive power of networks in terms of their depth is also shown in (Poole et al., 2016). In this paper we have further shown that DNNs are inherently biased towards lower frequency (smooth) functions over a finite parameter space. This suggests that DNNs strike a good balance between function smoothness and expressibility/parameter-efficiency compared with KM.

A.5.2. K-NN Classifier vs. DNN classifier

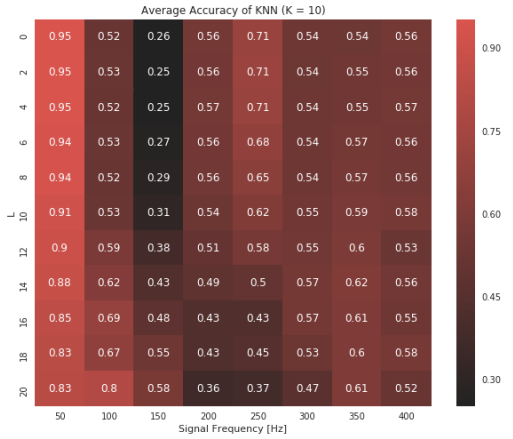
K-nearest neighbor (KNN) also has a historical importance as a classification algorithm due to its simplicity. It has been shown to be a consistent approximator Devroye et al. (1996), i.e., asymptotically its empirical risk goes to zero as $K \to \infty$ and $K/N \to 0$, where N is the number of training samples. However, because it is a memory based algorithm, it is prohibitively slow for large datasets. Since the smoothness of a KNN prediction function is not well studied, we compare the smoothness between KNN and DNN. For various values of K, we train a KNN classifier on a $k = 150$ frequency signal (which is binarized) defined on the $L = 20$ manifold (see section 4), and extract probability predictions on a box interval in $\mathbb{R}^2$. On this interval, we evaluate the 2D FFT and integrate out the angular components (where the angle is parameterized by φ) to obtain $\zeta(k)$:

$$\zeta(k) = \frac{d}{dk} \int_0^k dk' k' \int_0^{2\pi} d\varphi |\tilde{f}(k', \varphi)| \tag{39}$$

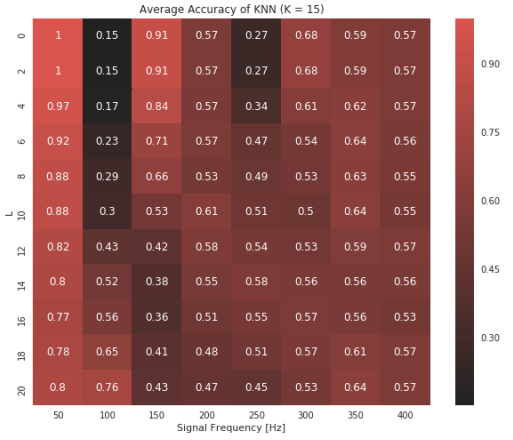
Finally, we plot $\zeta(k)$ for various K in figure A.13e. Furthermore, we train a DNN on the very same dataset and overlay the radial spectrum of the resulting probability map on the same plot. We find that while DNN's are as expressive as a $K = 1$ KNN classifier at lower (radial) frequencies, the frequency spectrum of DNNs decay faster than KNN classifier for all values of K considered, indicating that the DNN is smoother than the KNNs considered. We also repeat the experiment corresponding to Fig. 3.9 with KNNs (see Fig. A.13) for various K's, to find that unlike DNNs, KNNs do not necessarily perform better for larger L's, suggesting that KNNs do not exploit the geometry of the manifold like DNNs do.



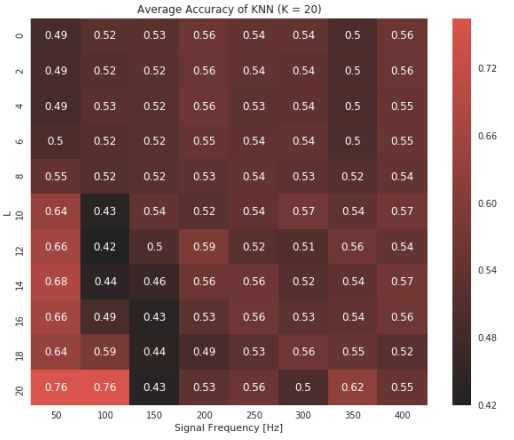
(a) $K = 5$.



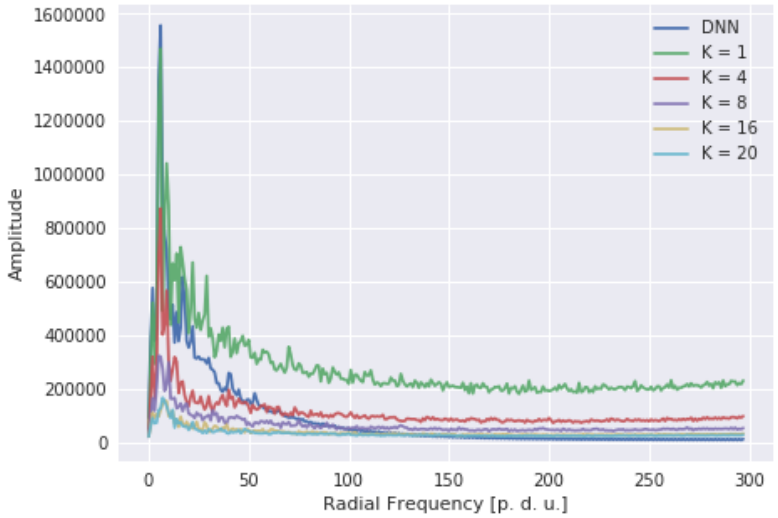
(b) $K = 10$.



(c) $K = 15$.



(d) $K = 20$.



(e) Frequency spectrum

Fig. A.13. (a,b,c,d): Heatmaps of training accuracies (L-vs-k) of KNNs for various K. When comparing with figure 3.9, note that the y-axis is flipped. (e): The frequency spectrum of KNNs with different values of K, and a DNN. The DNN learns a smoother function compared with the KNNs considered since the spectrum of the DNN decays faster compared with KNNs.

Appendix B

Tangent Feature Alignment: Supplementary Material

B.1. Tangent Features and Geometry

We describe in more formal detail some of the notions introduced in Section 2 of the paper. We will consider general classes of vector-valued predictors:

$$\mathcal{F} = \{f_{\mathbf{w}} \colon \mathcal{X} \to \mathbb{R}^c \mid \mathbf{w} \in \mathcal{W}\}, \tag{1}$$

where the parameter space $\mathcal{W}$ is a finite dimensional manifold of dimension P (typically $\mathbb{R}^P$). For multiclass classification, $f_{\mathbf{w}}$ outputs a score $f_{\mathbf{w}}(\mathbf{x})[y]$ for each class $y \in \{1 \cdots c\}$. Each function can also be viewed as a scalar function on $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y} = \{1 \cdots c\}$ is the set of classes.

B.1.1. Metric

We assume that $\mathbf{w} \to f_{\mathbf{w}}$ is a smooth mapping from $\mathcal{W}$ to $L^2(\rho, \mathbb{R}^c)$, where ρ is some input data distribution. The inclusion $\mathcal{F} \subset L^2(\rho, \mathbb{R}^c)$ equips $\mathcal{F}$ with the L^2 scalar product and corresponding norm:

$$\langle f, g \rangle_\rho := \mathbb{E}_{\mathbf{x} \sim \rho}[f(\mathbf{x})^\top g(\mathbf{x})], \qquad \|f\|_\rho := \sqrt{\langle f, f \rangle_\rho} \tag{2}$$

The parameter space $\mathcal{W}$ inherits a **metric tensor** $g_{\mathbf{w}}$ by pull-back of the scalar product $\langle f, g \rangle_\rho$ on $\mathcal{F}$. That is, given $\boldsymbol{\zeta}, \boldsymbol{\xi} \in \mathcal{T}_{\mathbf{w}}\mathcal{W} \cong \mathbb{R}^P$ on the tangent space at $\mathbf{w}$ (Lang, 2012),

$$g_{\mathbf{w}}(\boldsymbol{\zeta}, \boldsymbol{\xi}) = \langle \partial_{\boldsymbol{\zeta}} f_{\mathbf{w}}, \partial_{\boldsymbol{\xi}} f_{\mathbf{w}} \rangle_\rho \tag{3}$$

where $\partial_{\boldsymbol{\zeta}} f_{\mathbf{w}} = \langle df_{\mathbf{w}}, \boldsymbol{\zeta} \rangle$ is the directional derivative in the direction of $\boldsymbol{\zeta}$. Concretely, in a given basis of $\mathbb{R}^P$, the metric is represented by the matrix of gradient second moments:

$$(g_{\mathbf{w}})_{pq} = \mathbb{E}_{\mathbf{x}\sim\rho}\left[\left(\frac{\partial f_{\mathbf{w}}(\mathbf{x})}{\partial w_p}\right)^{\top} \frac{\partial f_{\mathbf{w}}(\mathbf{x})}{\partial w_q}\right] \tag{4}$$

where w_p, $p = 1, \cdots, P$ are the parameter coordinates. The metric shows up by spelling out the line element $ds^2 := \|df_{\mathbf{w}}\|^2_\rho$, since we have,

$$\|df_{\mathbf{w}}\|^2_\rho = \sum_{p,q=1}^{P} \langle \frac{\partial f_{\mathbf{w}}}{\partial w_p} dw_p, \frac{\partial f_{\mathbf{w}}}{\partial w_q} dw_q \rangle_\rho = \sum_{p,q=1}^{P} (g_{\mathbf{w}})_{pq}\, dw_p dw_q \tag{5}$$

B.1.2. Tangent Kernels

This geometry has a dual description in function space in terms of *kernels*. The idea is to view the differential of the mapping $\mathbf{w} \to f_{\mathbf{w}}$ at each $\mathbf{w}$ as a map $df_{\mathbf{w}} \colon \mathcal{X} \times \mathcal{Y} \to \mathcal{T}^*_{\mathbf{w}}\mathcal{W} \cong \mathbb{R}^p$ defining (joined) features in the (co)tangent space. In a given basis, this yields the **tangent features** given by the function derivatives w.r.t the parameters,

$$\Phi_{w_p}(\mathbf{x})[y] := \frac{\partial f_{\mathbf{w}}(\mathbf{x})[y]}{\partial w_p} \tag{6}$$

The tangent feature map $\Phi_{\mathbf{w}}$ can be viewed as a function mapping each pair $(\mathbf{x}, y)$ to a vector in $\mathbb{R}^P$. It defines the so-called **tangent kernel** (Jacot et al., 2018) through the Euclidean dot product $\langle \cdot, \cdot \rangle$ in $\mathbb{R}^P$:

$$k_{\mathbf{w}}(\mathbf{x}, y; \tilde{\mathbf{x}}, y') = \langle \Phi_{\mathbf{w}}(\mathbf{x})[y], \Phi_{\mathbf{w}}(\tilde{\mathbf{x}})[y'] \rangle = \sum_{p=1}^{P} \Phi_{w_p}(\mathbf{x})[y] \Phi_{w_p}(\tilde{\mathbf{x}})[y'] \tag{7}$$

It induces an integral operator on $L^2(\rho, \mathbb{R}^c)$ acting as

$$(k_{\mathbf{w}} \triangleright f)(\mathbf{x})[y] = \langle k_{\mathbf{w}}(\mathbf{x}, y; \,\cdot\,), f \rangle \tag{8}$$

The metric tensor (4) is expressed in terms of the tangent features as $(g_{\mathbf{w}})_{pq} = \langle \Phi_{w_p}, \Phi_{w_q} \rangle_\rho$.

B.1.3. Spectral Decomposition

The local metric tensor (as symmetric $P \times P$ matrix) and tangent kernel (as rank P integral operator) share the same spectrum. More generally, let

$$g_{\mathbf{w}} = \sum_{j=1}^{P} \lambda_{\mathbf{w}j} \boldsymbol{v}_{\mathbf{w}j} \boldsymbol{v}_{\mathbf{w}j}^{\top} \tag{9}$$

be the eigenvalue decomposition of the positive (semi-)definite symmetric matrix (4), where $\boldsymbol{v}_{\mathbf{w}j}^{\top} \boldsymbol{v}_{\mathbf{w}j'} = \delta_{jj'}$. Assuming non-degeneracy, i.e $\lambda_{\mathbf{w}j} > 0$, let $u_{\mathbf{w}j}, j \in \{1 \cdots P\}$ be the functions

in $L^2(\rho, \mathbb{R}^c)$ defined as:

$$u_{\mathbf{w}j}(\mathbf{x})[y] = \frac{1}{\sqrt{\lambda_{\mathbf{w}j}}} \boldsymbol{v}_{\mathbf{w}j}^\top \Phi_{\mathbf{w}}(\mathbf{x})[y] \tag{10}$$

The following result holds.

Proposition B.1 (Spectral decomposition)**.** The functions $(u_{j\mathbf{w}})_{j=1}^P$ form an orthonormal family in $L^2(\rho, \mathbb{R}^c)$. They are eigenfunctions of the tangent kernel as an integral operator, which admits the spectral decomposition:

$$k_{\mathbf{w}}(\mathbf{x}, y; \tilde{\mathbf{x}}, y') = \sum_{j=1}^{P} \lambda_{\mathbf{w}j}\, u_{\mathbf{w}j}(\mathbf{x})[y]\, u_{\mathbf{w}j}(\tilde{\mathbf{x}})[y'] \tag{11}$$

In particular metric tensor and tangent kernels share the same spectrum.

Proof. We first show orthonormality, i.e $\langle u_{\mathbf{w}j} u_{\mathbf{w}j'} \rangle_\rho = \delta_{jj'}$. We have indeed,

$$\begin{aligned}
\langle u_{\mathbf{w}j}, u_{\mathbf{w}j'} \rangle_\rho &= \tfrac{1}{\sqrt{\lambda_{\mathbf{w}j}\lambda_{\mathbf{w}j'}}} \textstyle\sum_{p,q=1}^P (\boldsymbol{v}_{\mathbf{w}j})_p (\boldsymbol{v}_{\mathbf{w}j})_q \langle \Phi_{w_p}, \Phi_{w_q} \rangle_\rho && (12)\\
&= \tfrac{1}{\sqrt{\lambda_{\mathbf{w}j}\lambda_{\mathbf{w}j'}}} \boldsymbol{v}_{\mathbf{w}j}^\top\, g_{\mathbf{w}}\, \boldsymbol{v}_{\mathbf{w}j'} && (13)\\
&= \tfrac{1}{\lambda_{\mathbf{w}j}} \lambda_{\mathbf{w}j} \delta_{jj'} && (14)\\
&= \delta_{jj'} && (15)
\end{aligned}$$

where we used the definition of the matrix $(g_{\mathbf{w}})_{pq}$ and its eigenvalue decomposition. Next, using the action (8) of the tangent kernel, we prove that the functions $u_{\mathbf{w}j}$ defined in (10) is an eigenfunction with eigenvalue $\lambda_{\mathbf{w}j}$:

$$\begin{aligned}
(k_{\mathbf{w}} \triangleright u_{\mathbf{w}j})(\mathbf{x})[y] &= \textstyle\sum_{p=1}^P \Phi_{w_p}(\mathbf{x})[y] \langle \Phi_{w_p}, u_{\mathbf{w}j} \rangle_\rho && (16)\\
&= \tfrac{1}{\sqrt{\lambda_{\mathbf{w}j}}} \textstyle\sum_{p,q=1}^P (\boldsymbol{v}_{\mathbf{w}j})_q\, \Phi_{w_p}(\mathbf{x})[y] \langle \Phi_{w_p}, \Phi_{w_q} \rangle && (17)\\
&= \tfrac{1}{\sqrt{\lambda_{\mathbf{w}j}}} \boldsymbol{v}_{\mathbf{w}j}^\top\, g_{\mathbf{w}}\, \Phi_{\mathbf{w}}(\mathbf{x})[y] && (18)\\
&= \tfrac{1}{\sqrt{\lambda_{\mathbf{w}j}}} (\lambda_{\mathbf{w}j} \boldsymbol{v}_{\mathbf{w}j}^\top)\, \Phi_{\mathbf{w}}(\mathbf{x})[y] && (19)\\
&= \lambda_{\mathbf{w}j} \tfrac{1}{\sqrt{\lambda_{\mathbf{w}j}}} \boldsymbol{v}_{\mathbf{w}j}^\top\, \Phi_{\mathbf{w}}(\mathbf{x})[y] && (20)\\
&= \lambda_{\mathbf{w}j}\, u_{\mathbf{w}j} && (21)
\end{aligned}$$

Inserting the resolution of unity $\mathrm{Id}_P = \sum_{j=1}^P \boldsymbol{v}_{\mathbf{w}j} \boldsymbol{v}_{\mathbf{w}j}^\top$ in the expression (7) of the tangent kernel directly yields the spectral decomposition (11). □

B.1.4. Sampled Versions

Given n input samples $\mathbf{x}_1, \cdots \mathbf{x}_n$, any function $f\colon \mathcal{X} \to \mathbb{R}^c$ yields a vector $\mathbf{f} \in \mathbb{R}^{nc}$ obtained by concatenating the outputs $f(\mathbf{x}_i) \in \mathbb{R}^c$ of the n input samples $\mathbf{x}_i$. The sample output scores $f_{\mathbf{w}}(\mathbf{x}_i)[y]$ thus yields $\mathbf{f}_{\mathbf{w}} \in \mathbb{R}^{nc}$; and the tangent features $\Phi_{w_p}(\mathbf{x}_i)[y]$ are represented as

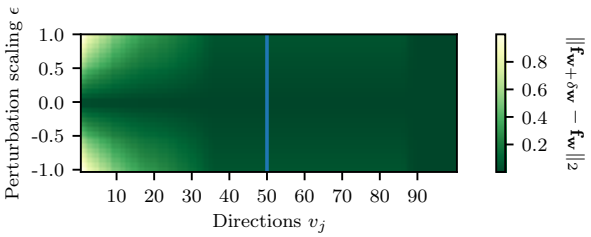


Fig. B.1. Variations of $\mathbf{f}_{\mathbf{w}}$ (evaluated on a test set) when perturbing the parameters in the directions given by the right singular vectors of the Jacobian (first 50 directions) or in randomly sampled directions (last 50 directions) on a VGG11 network trained for 10 epochs on CIFAR10. We observe that perturbations in most directions have almost no effect, except in those aligned with the top singular vectors.

a $nc \times P$ matrix $\mathbf{\Phi}_{\mathbf{w}}$. Using this notation, (4) and (7) yield the sample covariance $P \times P$ matrix and kernel (Gram) $nc \times nc$ matrix:

$$\boldsymbol{G}_{\mathbf{w}} = \mathbf{\Phi}_{\mathbf{w}}^{\top}\mathbf{\Phi}_{\mathbf{w}}, \quad \boldsymbol{K}_{\mathbf{w}} = \mathbf{\Phi}_{\mathbf{w}}\mathbf{\Phi}_{\mathbf{w}}^{\top} \tag{22}$$

The eigenvalue decompositions of $\boldsymbol{G}_{\mathbf{w}}$ and $\boldsymbol{K}_{\mathbf{w}}$ follow from the (SVD) of $\mathbf{\Phi}_{\mathbf{w}}$: assuming $P > nc$, we can write this SVD by indexing the singular values by a pair $J = (i, y)$ with $i = 1, \cdots n$ and $y = 1 \cdots c$ as

$$\mathbf{\Phi}_{\mathbf{w}} = \sum_{J=1}^{nc} \sqrt{\hat{\lambda}_{\mathbf{w}J}}\hat{\boldsymbol{u}}_{\mathbf{w}J}\hat{\boldsymbol{v}}_{\mathbf{w}J}^{\top} \tag{23}$$

Such decompositions summarize the predominant directions both in parameter and feature space, in the neighborhood of $\mathbf{w}$: a small variation $\delta\mathbf{w}$ induces the first order variation $\delta\mathbf{f}_{\mathbf{w}}$ of the function,

$$\delta\mathbf{f}_{\mathbf{w}} := \Phi_{\mathbf{w}}\delta\mathbf{w} = \sum_{J=1}^{nc} \sqrt{\hat{\lambda}_{\mathbf{w}J}}(\hat{\boldsymbol{v}}_{\mathbf{w}J}^{T}\delta\mathbf{w})\hat{\boldsymbol{u}}_{\mathbf{w}J} \tag{24}$$

Fig. B.1 illustrates this 'hierarchy' for a VGG11 network (Simonyan & Zisserman, 2014) trained for 10 epoches on CIFAR10 (Krizhevsky & Hinton, 2009). We observe that perturbations in most directions have almost no effect, except in those aligned with the top singular vectors. This is reflected by a strong anisotropy of the tangent kernel spectrum. Recent analytical results for wide random neural networks also point to such a pathological structure of the spectrum (Karakida et al., 2019a,b).

B.1.5. Spectral Bias

B.1.5.1. Proof of Lemma 4.1. We consider parameter updates $\delta\mathbf{w}_{\mathrm{GD}} := -\eta\nabla_{\mathbf{w}}L$ for gradient descent w.r.t a loss $L := L(\mathbf{f}_{\mathbf{w}})$, which is a function of the vector $\mathbf{f}_{\mathbf{w}} \in \mathbb{R}^{nc}$ of sample output scores. We reformulate Lemma 4.1, extended to the multiclass setting.

Proposition B.2 (Lemma 4.1 restated)**.** The gradient descent function updates in first order Taylor approximation, $\delta f_{\text{GD}}(\mathbf{x})[y] := \langle \delta \mathbf{w}_{\text{GD}}, \Phi_{\mathbf{w}}(\mathbf{x})[y] \rangle$, decompose as,

$$\delta f_{\text{GD}}(\mathbf{x})[y] = \sum_{j=1}^{P} \delta f_j \, u_{\mathbf{w}j}(\mathbf{x})[y], \qquad \delta f_j = -\eta \lambda_{\mathbf{w}j} (\boldsymbol{u}_{\mathbf{w}j}^\top \nabla_{\mathbf{f}_\mathbf{w}} L) \tag{25}$$

where $u_{\mathbf{w}j}$ are the eigenfunctions (10) of the tangent kernel and $\boldsymbol{u}_{\mathbf{w}j} \in \mathbb{R}^{nc}$ are their corresponding sample vector.

Proof. Inserting the resolution of unity $\text{Id}_P = \sum_{j=1}^P \boldsymbol{v}_{\mathbf{w}j} \boldsymbol{v}_{\mathbf{w}j}^\top$ in the expression for δf_{GD} yields

$$\begin{aligned} \delta f_{\text{GD}}(\mathbf{x})[y] &= \textstyle\sum_{j=1}^P (\boldsymbol{v}_{\mathbf{w}j}^\top \delta \mathbf{w}_{\text{GD}}) \, \boldsymbol{v}_{\mathbf{w}j}^\top \Phi_{\mathbf{w}}(\mathbf{x})[y] && (26) \\ &= \textstyle\sum_{j=1}^P \sqrt{\lambda_{\mathbf{w}j}} (\boldsymbol{v}_{\mathbf{w}j}^\top \delta \mathbf{w}_{\text{GD}}) \, u_{\mathbf{w}j}(\mathbf{x})[y] && (27) \end{aligned}$$

Next, by the chain rule $\nabla_{\mathbf{w}} L = \boldsymbol{\Phi}_{\mathbf{w}}^\top \nabla_{\mathbf{f}_\mathbf{w}} L$, so we can spell out:

$$\delta \mathbf{w}_{\text{GD}} = -\eta \sum_{j=1}^{P} \sqrt{\lambda_{\mathbf{w}j}} (\boldsymbol{u}_{\mathbf{w}j}^\top \nabla_{\mathbf{f}_\mathbf{w}} L) \, \boldsymbol{v}_{\mathbf{w}j}, \tag{28}$$

which implies that $(\boldsymbol{v}_{\mathbf{w}j}^\top \delta \mathbf{w}_{\text{GD}}) = \sqrt{\lambda_{\mathbf{w}j}} (\boldsymbol{u}_{\mathbf{w}j}^\top \nabla_{\mathbf{f}_\mathbf{w}} L)$. Substituting in (26) gives the desired result. □

The decomposition (30) has a *sampled* version in terms of tangent feature and kernel matrices. Using the notation of SVD (23), let $\hat{\lambda}_{\mathbf{w}J}, \hat{\boldsymbol{u}}_{\mathbf{w}j}$ and $\hat{\boldsymbol{v}}_{\mathbf{w}j}$ be correspond to the (non-zero) eigenvalues and eigenvectors of the sample covariance and kernel (22). We consider the tangent kernel **principal components**, defined as the functions

$$\hat{u}_{\mathbf{w}J}(\mathbf{x})[y] = \frac{1}{\sqrt{\lambda_{\mathbf{w}J}}} \langle \hat{\boldsymbol{v}}_{\mathbf{w}J}, \Phi_{\mathbf{w}}(\mathbf{x})[y] \rangle, \tag{29}$$

which form an orthonormal family for the in-sample scalar product $\langle f, g \rangle_{\text{in}} = \sum_{i=1}^n f(\mathbf{x}_i) g(\mathbf{x}_i)$ and approximate the true kernel eigenfunctions (10) (e.g., Bengio et al., 2004; Braun, 2005). One can easily check from (23) that the vector $\hat{\boldsymbol{u}}_{\mathbf{w}J} \in \mathbb{R}^{nc}$ of sample outputs $\hat{u}(\mathbf{x}_i)[y]$ coincides with the J-th eigenvector of the tangent kernel matrix.

Proposition B.3 (Sampled version of Prop B.2)**.** The gradient descent function updates in first order Taylor approximation, $\delta f_{\text{GD}}(\mathbf{x})[y] := \langle \delta \mathbf{w}_{\text{GD}}, \Phi_{\mathbf{w}}(\mathbf{x})[y] \rangle$ decompose as,

$$\delta f_{\text{GD}}(\mathbf{x})[y] = \sum_{j=1}^{nc} \delta f_J \, \hat{u}_{\mathbf{w}J}(\mathbf{x})[y], \qquad \delta f_J = -\eta \hat{\lambda}_{\mathbf{w}J} (\hat{\boldsymbol{u}}_{\mathbf{w}J}^\top \nabla_{\mathbf{f}_\mathbf{w}} L) \tag{30}$$

in terms of the principal components (29) of the tangent kernel.

PROOF. Same proof as for the previous Proposition, using the resolution of unity $\mathrm{Id}_{nc} = \sum_{J=1}^{nc} \hat{\boldsymbol{v}}_{\mathbf{w}J} \hat{\boldsymbol{v}}_{\mathbf{w}J}^\top$. □

B.1.5.2. The Case of Linear Regression. The previous Proposition gives a 'local' version of a classic decomposition of the training dynamics in linear regression (e.g., Advani & Saxe, 2017)). In such a setting, $f_{\mathbf{w}} = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$ are linearly parametrized scalar functions ($c = 1$) and $L = \frac{1}{2}\|\mathbf{f}_{\mathbf{w}} - \boldsymbol{y}\|^2$. We denote by $\boldsymbol{\Phi} = \sum_{j=1}^{n} \hat{\lambda}_j \hat{\boldsymbol{u}}_j \hat{\boldsymbol{v}}_j^\top$ the $n \times P$ feature matrix and its SVD.

Proposition B.4. Gradient descent of the squared loss yields the function iterates,

$$f_{\mathbf{w}_t} = f_{\mathbf{w}^*} + (\mathrm{Id} - \eta K)^t (f_{\mathbf{w}_0} - f_{\mathbf{w}^*}) \tag{31}$$

where Id is the identity map and K is the operator acting on functions as $(K \triangleright f)(\mathbf{x}) = \sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i) f(\mathbf{x}_i)$ in terms of the kernel $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \Phi(\mathbf{x}), \Phi(\tilde{\mathbf{x}}) \rangle$.

PROOF. The updates $\delta \mathbf{w}_{\mathrm{GD}} := -\eta \nabla_{\mathbf{w}} L$ induce the (exact) functional updates $\delta f_{\mathrm{GD}} = f_{\mathbf{w}_{t+1}} - f_{\mathbf{w}_t}$ given by

$$\delta f_{\mathrm{GD}}(\mathbf{x}) = -\eta \sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i)(f_{\mathbf{w}_t}(\mathbf{x}_i) - \boldsymbol{y}_i) \tag{32}$$

Substituting $\boldsymbol{y}_i = f_{\mathbf{w}^*}(\mathbf{x}_i)$ gives $f_{\mathbf{w}_{t+1}} - f_{\mathbf{w}^*} = (\mathrm{id} - \eta K)(f_{\mathbf{w}_t} - f_{\mathbf{w}^*})$. Equ. 31 follows by induction. □

Lemma B.1. The kernel principal components $\hat{u}_j(\mathbf{x}) = \frac{1}{\sqrt{\hat{\lambda}_j}} \langle \hat{\boldsymbol{v}}_j, \Phi_{\mathbf{w}}(\mathbf{x}) \rangle$ are eigenfunctions of the operator K with corresponding eigenvalues $\hat{\lambda}_j$.

PROOF. By inserting $Id_n = \sum_j \hat{\boldsymbol{v}}_j \hat{\boldsymbol{v}}_j^\top$ in the expression of the kernel, one can write $k(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^{n} \hat{u}_j(\mathbf{x}) \hat{u}_j(\mathbf{x}_i)$. Substituting in the definition of K and using the orthonormality of $\hat{u}_j$ for the in-sample scalar product yield $K \triangleright \hat{u}_j = \hat{\lambda}_j \hat{u}_j$. □

Together with(31), this directly leads to the decoupling of the training dynamics in the basis of kernel principal components.

Proposition B.5 (Spectral Bias for Linear Regression)**.** By initializing $\mathbf{w}_0 = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}_0$ in the span of the features, the function iterates in (31) uniquely decompose as,

$$f_{\mathbf{w}_t}(\mathbf{x}) = \sum_{j=1}^{n} f_{jt} \hat{u}_j(\mathbf{x}), \quad f_{jt} = f_j^* + (1 - \eta \hat{\lambda}_j)^t (f_{j0} - f_j^*) \tag{33}$$

where f_j^* are the coefficients of the (mininum ℓ_2-norm) interpolating solution.

This standard result shows how each independent mode labelled by j has its own linear convergence rate. For example setting $\eta = 1/\hat{\lambda}_1$, this gives $f_{jt} - f_j^* \propto e^{-t/\tau_j}$, where $\tau_j = -\log(1 - \frac{\hat{\lambda}_j}{\hat{\lambda}_1})$ is the time constant (number of iterations) for the mode j. Top modes f_j^* of the target function are learned faster than low modes.

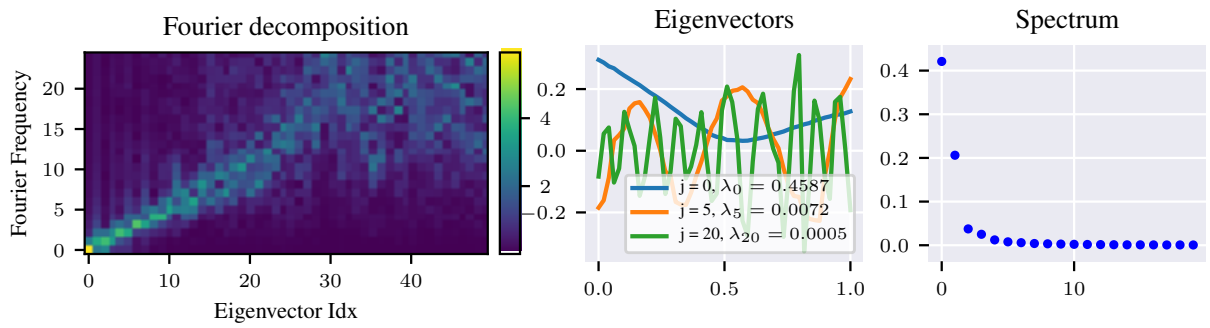


Fig. B.2. Eigendecomposition of the tangent kernel matrix of a random 6-layer deep 256-unit wide MLP on 1D uniform data (50 equally spaced points in [0,1]). **(left)** Fourier decomposition (y-axis for frequency, colorbar for magnitude) of each eigenvector (x-axis), ranked in nonincreasing order of the eigenvalues. We observe that eigenvectors with increasing index j (hence decreasing eigenvalues) correspond to modes with increasing Fourier frequency. **(middle)** Plot of the j-th eigenvectors with $j \in \{0, 5, 20\}$ and **(right)** distribution of eigenvalues. We note the fast decay (e.g $\lambda_{10}/\lambda_1 \approx 4‰$).

In linearized regimes where deep learning reduces to kernel regression (Jacot et al., 2018; Du et al., 2019b; Allen-Zhu et al., 2019), one can dwell further the nature of such a bias by analyzing the eigenfunctions of the neural tangent kernel (e.g., Yang & Salman, 2019). As a simple example, for a randomly initialized MLP on 1D uniform data, Fig. B.2 shows the Fourier decomposition of such eigenfunctions, ranked in nonincreasing order of the eigenvalues. We observe that eigenfunctions with increasing index j (hence decreasing eigenvalues) correspond to modes with increasing Fourier frequency, with a remarkable alignment with Fourier modes for the first half of the spectrum. This in line with observations (e.g., Rahaman et al., 2019) that deep networks tend to prioritize learning low frequency modes during training.

B.2. Complexity Bounds

In this section, we spell out details and proofs for the content of Section 4.

B.2.1. Rademacher Complexity

Given a family $\mathcal{G} \subset \mathbb{R}^{\mathcal{Z}}$ of real-valued functions on a probability space $(\mathcal{Z}, \rho)$, the empirical Rademacher complexity of $\mathcal{G}$ with respect to a sample $\mathcal{S} = \{\mathbf{z}_1, \cdots \mathbf{z}_n\} \sim \rho^n$ is defined as (Mohri et al., 2012):

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}) = \mathbb{E}_{\boldsymbol{\sigma} \in \{\pm 1\}^n} \left[\sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g(\mathbf{z}_i) \right], \tag{34}$$

where the expectation is over n i.i.d uniform random variables $\sigma_1, \cdots \sigma_n \in \{\pm 1\}$. For any $n \geq 1$, the Rademacher complexity with respect to samples of size n is then $\mathcal{R}_n(\mathcal{G}) = \mathbb{E}_{\mathcal{S} \sim \rho^n} \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G})$.

B.2.2. Generalization Bounds

Generalization bounds based on Rademacher complexity are standard (Bartlett et al., 2017; Mohri et al., 2012). We give here one instance of such a bound, relevant for classification tasks.

Setup. We consider a family $\mathcal{F}$ of functions $f_{\mathbf{w}} \colon \mathcal{X} \to \mathbb{R}^c$ that output a score or probability $f_{\mathbf{w}}(\mathbf{x})[y]$ for each class $y \in \{1 \cdots c\}$ (we take $c = 1$ for binary classification). The task is to find a predictor $f_{\mathbf{w}} \in \mathcal{F}$ with small expected classification error, which can be expressed e.g. as

$$L_0(f_{\mathbf{w}}) = \mathbb{P}_{(\mathbf{x},y)\sim\rho} \{\mu(f_{\mathbf{w}}(\mathbf{x}), y) < 0\} \tag{35}$$

where $\mu(f(\mathbf{x}), y)$ denotes the **margin**,

$$\mu(f(\mathbf{x}), y) = \begin{cases} f(\mathbf{x})y & \text{binary case} \\ f(\mathbf{x})[y] - \max_{y' \neq y} f(\mathbf{x})[y'] & \text{multiclass case} \end{cases} \tag{36}$$

Margin Bound. We consider the **margin loss**,

$$\ell_\gamma(f_{\mathbf{w}}(\mathbf{x}), y)) = \phi_\gamma(\mu(f_{\mathbf{w}}(\mathbf{x}), y)) \tag{37}$$

where $\gamma > 0$, and ϕ_γ is the **ramp** function: $\phi_\gamma(u) = 1$ if $u \leq 0$, $\phi(u) = 0$ if $u > \gamma$ and $\phi(u) = 1 - u/\gamma$ otherwise. We have the following bound for the expected error (35). With probability at least $1 - \delta$ over the draw $\mathcal{S} = \{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^n$ of size n, the following holds for all $f_{\mathbf{w}} \in \mathcal{F}$ (Mohri et al., 2012, Theorems 4.4.and 8.1):

$$L_0(f_{\mathbf{w}}) \leq \widehat{L}_\gamma(f_{\mathbf{w}}) + 2\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F}, \cdot)) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2n}} \tag{38}$$

where $\widehat{L}_\gamma(f_{\mathbf{w}}) = \frac{1}{n} \sum_{i=1}^n \ell_\gamma(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$ is the empirical margin error and $\ell_\gamma(\mathcal{F}, \cdot)$ is the **loss class**,

$$\ell_\gamma(\mathcal{F}, \cdot) = \{(\mathbf{x}, y) \mapsto \ell_\gamma(f_{\mathbf{w}}(\mathbf{x}), y) \mid f_{\mathbf{w}} \in \mathcal{F}\} \tag{39}$$

For binary classifiers, because ϕ_γ is $1/\gamma$-Lipschitz, we have in addition

$$\mathcal{R}_{\mathcal{S}}(\ell_\gamma(\mathcal{F}, \cdot)) \leq \frac{1}{\gamma} \mathcal{R}_{\mathcal{S}}(\mathcal{F}) \tag{40}$$

by Talagrand's contraction lemma (Ledoux & Talagrand, 2013) (see e.g. Mohri et al., 2012, lemma 4.2 for a detailed proof).

B.2.3. Complexity Bounds: Proofs

We first derive standard bounds for the linear classes of scalar functions,

$$\mathcal{F}^A_{M_A} = \{f_{\mathbf{w}} \colon \mathbf{x} \mapsto \langle \mathbf{w}, \Phi(\mathbf{x})\rangle \mid \|\mathbf{w}\|_A \leq M_A\} \tag{41}$$

Proposition B.6. The empirical Rademacher complexity of $\mathcal{F}^A_{M_A}$ is bounded as,

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}^A_{M_A}) \leq (M_A/n)\sqrt{\mathrm{Tr}\boldsymbol{K}_A} \tag{42}$$

where $(\boldsymbol{K}_A)_{ij} = k_A(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel matrix associated to the rescaled features $A^{-1}\Phi$.

Proof. We use the notation of Section 4. For given Rademacher variables $\boldsymbol{\sigma} \in \{\pm 1\}^n$, we have,

$$\begin{aligned}
\sup_{f\in\mathcal{F}^A_{M_A}} \sum_{i=1}^n \sigma_i f(\mathbf{x}_i) &= \sup_{\|\mathbf{w}\|_A \leq M_A} \sum_{i=1}^n \sigma_i \langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle \\
&= \sup_{\|A^\top \mathbf{w}\|_2 \leq M_A} \sum_{i=1}^n \sigma_i \langle A^\top \mathbf{w}, A^{-1}\Phi(\mathbf{x}_i)\rangle \\
&= \sup_{\|\tilde{\mathbf{w}}\|_2 \leq M_A} \langle \tilde{\mathbf{w}}, \sum_{i=1}^n \sigma_i A^{-1}\Phi(\mathbf{x}_i)\rangle \\
&= M_A \left\| \sum_{i=1}^n \sigma_i A^{-1}\Phi(\mathbf{x}_i) \right\|_2 \\
&= M_A \sqrt{\boldsymbol{\sigma}^\top \boldsymbol{K}_A \boldsymbol{\sigma}}
\end{aligned} \tag{43}$$

From (43) and the definition (34) we obtain:

$$\begin{aligned}
\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}^A_{M_A}) &= \frac{M_A}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sqrt{\boldsymbol{\sigma}^\top \boldsymbol{K}_A \boldsymbol{\sigma}}\right] \\
&\leq \frac{M_A}{n}\sqrt{\mathbb{E}_{\boldsymbol{\sigma}}\left[\boldsymbol{\sigma}^\top \boldsymbol{K}_A \boldsymbol{\sigma}\right]} \\
&\leq \frac{M_A}{n}\sqrt{\mathrm{Tr}\boldsymbol{K}_A}
\end{aligned} \tag{44}$$

where we used Jensen's inequality to pass $\mathbb{E}_\sigma$ under the root, and that $\mathbb{E}[\sigma_i] = 0$ and $\sigma_i^2 = 1$ for all i. □

We now extend the result to the families (11) of learning flows:

$$\mathcal{F}^{\boldsymbol{A}}_{\boldsymbol{m}} = \{f_{\mathbf{w}} \colon \mathbf{x} \mapsto \textstyle\sum_t \langle \delta\mathbf{w}_t, \Phi(\mathbf{x})\rangle \mid \|\delta\mathbf{w}_t\|_{A_t} \leq m_t\} \tag{45}$$

Theorem B.1 (Theorem 4.2 restated)**.** The empirical Rademacher complexity of $\mathcal{F}^{\boldsymbol{A}}_{\boldsymbol{m}}$ is bounded as,

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}^{\boldsymbol{A}}_{\boldsymbol{m}}) \leq \textstyle\sum_t (m_t/n)\sqrt{\mathrm{Tr}\boldsymbol{K}_{A_t}} \tag{46}$$

where $(\boldsymbol{K}_{A_t})_{ij} = k_{A_t}(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel matrix associated to the rescaled features $A_t^{-1}\Phi$.

PROOF. This is simple extension of the previous proof:

$$\begin{aligned}\sup_{f \in \mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{A}}} \sum_{i=1}^{n} \sigma_i f(\mathbf{x}_i) &= \sup_{\|\delta\mathbf{w}_t\|_{A_t} \le m_t} \sum_{i=1}^{n} \sigma_i \sum_t \langle \delta\mathbf{w}_t, \Phi(\mathbf{x}_i)\rangle \\ &= \sum_t \sup_{\|\tilde{\delta}\mathbf{w}_t\|_2 \le m_t} \langle \tilde{\delta}\mathbf{w}_t, \sum_{i=1}^{n} \sigma_i A_t^{-1}\Phi(\mathbf{x}_i)\rangle \\ &= \sum_t m_t \sqrt{\boldsymbol{\sigma}^\top \boldsymbol{K}_{A_t} \boldsymbol{\sigma}} \end{aligned} \tag{47}$$

and we conclude as in (44). □

Finally, we note that the same result can be formulated in terms of an evolving feature map $\Phi_t = A_t^{-1}\Phi$ with kernel $k_t(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \Phi_t(\mathbf{x}), \Phi_t(\tilde{\mathbf{x}})\rangle$ In fact by reparametrization invariance, the function updates can also be written as $\delta f_{\mathbf{w}_t}(\mathbf{x}) = \langle \tilde{\delta}\mathbf{w}_t, \Phi_t(\mathbf{x})\rangle$ where $\tilde{\delta}\mathbf{w}_t = A_t^\top \delta\mathbf{w}_t$. The function class (11) can equivalently be written as $\mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{A}} = \mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{\Phi}}$ where $\boldsymbol{\Phi}$ denotes a fixed sequence of feature maps, $\boldsymbol{\Phi} = \{\Phi_t\}_t$ and

$$\mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{\Phi}} = \{f_{\mathbf{w}} \colon \mathbf{x} \mapsto \textstyle\sum_t \langle \tilde{\delta}\mathbf{w}_t, \Phi_t(\mathbf{x})\rangle \mid \|\tilde{\delta}\mathbf{w}_t\|_2 \le m_t\} \tag{48}$$

In this formulation, the result (46) is expressed as,

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_{\boldsymbol{m}}^{\boldsymbol{\Phi}}) \le \textstyle\sum_t (m_t/n)\sqrt{\mathrm{Tr}\boldsymbol{K}_t} \tag{49}$$

where $(\boldsymbol{K}_t)_{ij} = k_t(\mathbf{x}_i, \tilde{\mathbf{x}_j})$ is the kernel matrix associated to the feature map Φ_t.

B.2.4. Bounds for Multiclass Classification

The generalization bound (38) is based on the **margin loss class** (39). In this section, we show how to bound $\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F}, \cdot))$ in terms of tangent kernels for the original class $\mathcal{F}$ of functions $f_{\mathbf{w}} \colon \mathcal{X} \to \mathbb{R}^c$ instead. Although the proof is adapted from standard techniques, to our knowledge Lemma B.2 and Theorem B.2 below are new results. In what follows, we denote by $\mu_{\mathcal{F}}$ the margin class,

$$\mu_{\mathcal{F}} = \{(\mathbf{x}, y) \to \mu(f_{\mathbf{w}}(\mathbf{x}), y) \mid f_{\mathbf{w}} \in \mathcal{F}\} \tag{50}$$

where $\mu(f_{\mathbf{w}}(\mathbf{x}), y))$ is the margin (36). We also define, for each $y \in \{1 \cdots c\}$,

$$\mathcal{F}_y = \{\mathbf{x} \mapsto f_{\mathbf{w}}(\mathbf{x})[y] \mid f_{\mathbf{w}} \in \mathcal{F}\}, \quad \mu_{\mathcal{F},y} = \{\mathbf{x} \mapsto \mu(f_{\mathbf{w}}(\mathbf{x}), y) \mid f_{\mathbf{w}} \in \mathcal{F}\} \tag{51}$$

Lemma B.2. The following inequality holds:

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F},\cdot)) \leq \frac{c}{\gamma}\sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) \tag{52}$$

PROOF. We first follow the first steps of the proof of (Mohri et al., 2012, Theorem 8.1) to show that

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F},\cdot)) \leq \frac{1}{\gamma}\sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mu_{\mathcal{F},y}) \tag{53}$$

We reproduce these steps here for completeness: first, it follows from the $1/\gamma$-Lipschitzness of the ramp loss ϕ_γ in (37) and Talagrand's contraction lemma (Mohri et al., 2012, lemma 4.2) that

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F},\cdot)) \leq \frac{1}{\gamma}\widehat{\mathcal{R}}_{\mathcal{S}}(\mu_{\mathcal{F}}) \tag{54}$$

Next, we write

$$\begin{aligned}
\widehat{\mathcal{R}}_{\mathcal{S}}(\mu_{\mathcal{F}}) &:= \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i\mu(f_{\mathbf{w}}(\mathbf{x}_i),y_i)\right] \\
&= \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i\sum_{y=1}^{c}\mu(f_{\mathbf{w}}(\mathbf{x}_i),y)\,\delta_{y,y_i}\right] \\
&= \frac{1}{n}\sum_{y=1}^{c}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i\mu(f_{\mathbf{w}}(\mathbf{x}_i),y)\,\delta_{y,y_i}\right]
\end{aligned} \tag{55}$$

where $\delta_{y,y_i} = 1$ if $y = y_i$ and 0 otherwise; the second inequality follows from the sub-additivity of sup. Substituting $\delta_{y,y_i} = \frac{1}{2}(\epsilon_i + \frac{1}{2})$ where $\epsilon_i = 2\delta_{y,y_i} - 1 \in \{\pm 1\}$, we obtain

$$\begin{aligned}
\widehat{\mathcal{R}}_{\mathcal{S}}(\mu_{\mathcal{F}}) &\leq \frac{1}{2n}\sum_{y=1}^{c}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}(\epsilon_i\sigma_i)\mu(f_{\mathbf{w}}(\mathbf{x}_i),y)\right] + \frac{1}{2n}\sum_{y=1}^{c}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i\mu(f_{\mathbf{w}}(\mathbf{x}_i),y)\right] \\
&= \sum_{y=1}^{c}\frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i\mu(f_{\mathbf{w}}(\mathbf{x}_i),y)\right] \\
&= \sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mu_{\mathcal{F},y})
\end{aligned} \tag{56}$$

Together with (54), this leads to (53).

Now, spelling out $\mu(f_{\mathbf{w}}(\mathbf{x}_i, y))$ gives

$$\begin{aligned}
\widehat{\mathcal{R}}_{\mathcal{S}}(\mu_{\mathcal{F},y}) &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i(f_{\mathbf{w}}(\mathbf{x}_i)[y] - \max_{y'\neq y} f_{\mathbf{w}}(\mathbf{x}_i)[y'])\right] \\
&= \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) + \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}(-\sigma_i)\max_{y'\neq y} f_{\mathbf{w}}(\mathbf{x}_i)[y']\right] \\
&= \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) + \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathbf{w}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i\max_{y'\neq y} f_{\mathbf{w}}(\mathbf{x}_i)[y']\right] \\
&\leq \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) + \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}_y) \qquad (57)
\end{aligned}$$

where $\mathcal{G}_y = \{\max\{f_{y'} : y' \neq y\} \,|\, f_{y'} \in \mathcal{F}_{y'}\}$. Now (Mohri et al., 2012, lemma 8.1) show that $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}_y) \leq \sum_{y'\neq y}\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_{y'})$. This leads to

$$\begin{aligned}
\sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mu_{\mathcal{F},y}) &\leq \sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) + \sum_{y=1}^{c}\sum_{\substack{y'=1\\y'\neq y}}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_{y'}) \\
&= \sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) + (c-1)\sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) \\
&= c\sum_{y=1}^{c}\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}_y) \qquad (58)
\end{aligned}$$

Substituting in (53) finishes the proof. □

In the linear case, this results leads to analogous theorems as in B.2.3 in the multiclass setting. For example, considering the linear families of functions $\mathcal{X} \to \mathbb{R}^c$,

$$\mathcal{F}^A_{M_A} = \{\mathbf{x} \mapsto f_{\mathbf{w}}(\mathbf{x})[y] := \langle \mathbf{w}, \Phi(\mathbf{x})[y]\rangle \mid \|\mathbf{w}\|_A \leq M_A\} \qquad (59)$$

where $(\mathbf{x}, y) \mapsto \Phi(\mathbf{x})[y]$ is some joint feature map, we have the following

Theorem B.2. The emp. Rademacher complexity of the margin loss class $\ell_\gamma(\mathcal{F}^A_{M_A}, \cdot)$ is bounded as,

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F}^A_{M_A}, \cdot)) \leq (c^{3/2}M_A/\gamma n)\sqrt{\mathrm{Tr}\boldsymbol{K}_A} \qquad (60)$$

where $(\boldsymbol{K}_A)^{yy'}_{ij}$ is the kernel $nc \times nc$ matrix associated to the rescaled features $A^{-1}\Phi(\mathbf{x})[y]$.

Proof. Eq.52, and Theorem B.2 applied to each linear family $\mathcal{F}_y$ of (scalar) functions leads to

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F}^A_{M_A}, \cdot)) \leq \frac{c}{\gamma}\sum_{y=1}^{c}\frac{M_A}{n}\sqrt{\mathrm{Tr}\boldsymbol{K}^{yy}_A} \qquad (61)$$

where $\mathrm{Tr}\boldsymbol{K}_A^{yy} := \sum_{i=1}^n (\boldsymbol{K}_A)_{ii}^{yy}$ is computed w.r.t to the indices $i = 1,...,n$ for fixed y. Passing the average $\frac{1}{c}\sum_{y=1}^c$ under the root using Jensen inequality, we conclude:

$$\begin{aligned}\widehat{\mathcal{R}}_{\mathcal{S}}(\ell_\gamma(\mathcal{F}_{M_A}^A, \cdot)) &\leq \frac{c^2 M_A}{\gamma n}\sqrt{\frac{1}{c}\sum_{y=1}^c \mathrm{Tr}\boldsymbol{K}_A^{yy}} \\ &= \frac{c^{3/2} M_A}{\gamma n}\sqrt{\mathrm{Tr}\boldsymbol{K}_A} \end{aligned} \tag{62}$$

□

The proof of the extension of these bounds to families learning flows follows the same line as in B.2.3.

B.2.5. Which Norm for Measuring Capacity?

Implicit biases of gradient descent are relatively well understood in linear models (e.g Gunasekar et al. (2018)). For example when using square loss, it is well-known that gradient descent (initialized in the span of the data) converges to minimum ℓ^2 norm (resp. RKHS norm) solutions in parameter space (resp. function space). Yet, as pointed out by Belkin et al. (2018); Muthukumar et al. (2020), measuring capacity in terms of such norms is not coherently linked with generalization in practice. Here we discuss this issue by highlighting the critical dependence of meaningful norm-based capacity on the geometry defined by the features. We use the notation of Section 4.1: $\boldsymbol{\Phi} = \sum_{j=1}^n \sqrt{\lambda_j}\boldsymbol{u}_j\boldsymbol{v}_j^\top$ denote the $n \times P$ feature matrix and its SVD decomposition.

A standard approach is to measure capacity in terms of the ℓ^2 norm the weight vector, e.g using bounds (10) with $A = \mathrm{Id}$. If the distribution of solutions $\mathbf{w}_{\mathcal{S}}^*$, where $\mathcal{S} \sim \rho^n$ is sampled from the input distribution, is reasonably isotropic, taking the smallest ℓ^2 ball containing them (with high probability) gives an accurate description of the class of trained models. However for very anisotropic distributions, the solutions do not fill any such ball so describing trained models in terms of ℓ^2 balls is wasteful (Schölkopf et al., 1999a).

Now, for minimum ℓ^2 norm interpolators (Hastie et al., 2009),

$$\mathbf{w}^* = \boldsymbol{\Phi}^\top \boldsymbol{K}^{-1}\boldsymbol{y} = \sum_{j=1}^n \frac{\boldsymbol{u}_j^\top \boldsymbol{y}}{\sqrt{\lambda_j}}\,\boldsymbol{v}_j, \tag{63}$$

where $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ is the kernel matrix, the solution distribution typically inherits the anisotropy of the features. For example, if $y_i = \bar{y}(\mathbf{x}_i) + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, the covariance of the solutions with respect to noise is $\mathrm{cov}_{\boldsymbol{\varepsilon}}[\mathbf{w}^*, \mathbf{w}^*] = \sum_j \frac{\sigma^2}{\lambda_j}\boldsymbol{v}_j\boldsymbol{v}_j^\top$, which scales as $1/\lambda_j$ along $\boldsymbol{v}_j$.

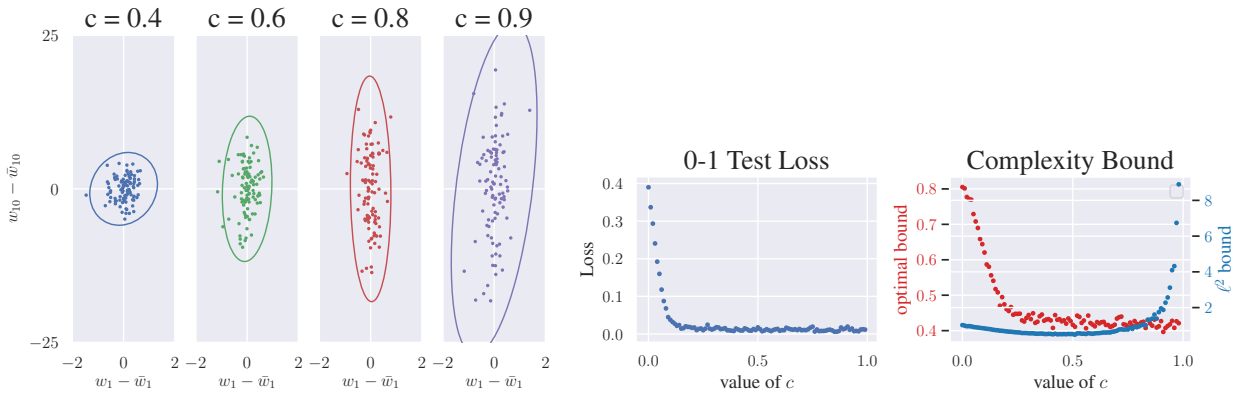


Fig. B.3. Left: 2D projection of the minimum ℓ^2-norm interpolators $\mathbf{w}^*_{\mathcal{S}}$, $\mathcal{S} \sim \rho^n$, for linear models $f_{\mathbf{w}} = \langle \mathbf{w}, \Phi_c \rangle$, as the feature scaling factor varies from 0 (white features) to 1 (original, anisotropic features). For larger c, the solutions scatter in a very anisotropic way. **Right:** Average test classification loss and complexity bounds (60) with $A = \mathrm{Id}$ (blue plot) for the solution vectors $\mathbf{w}^*_{\mathcal{S}}$, as we increase the scaling factor c. As feature anisotropy increases, the bound becomes increasingly loose and fails to reflect the shape of the test error. By contrast, the bound (10) with A optimized as in Proposition B.7 (red plot) does not suffer from this problem.

To visualize this on a simple setting, we consider P random features of a RBF kernel[1], fit on 1D data $\mathbf{x}$ modelled by N equally spaced points in $[-a, a]$. In this setting, the (true) feature map is represented by a $N \times P$ matrix with SVD $\Phi = \sum_j \sqrt{l_j} \boldsymbol{\psi}_j \boldsymbol{\varphi}_j^\top$. We assume the (true) labels are defined by the deterministic function $y(\mathbf{x}) = \mathrm{sign}(\boldsymbol{\psi}_1(\mathbf{x}))$. To highlight the effect of feature anisotropy, we further rescale the singular values as $l_j^c = 1 + c(l_j - 1)$ so as to interpolate between whitened features ($c=0$) and the original ones ($c=1$). We set $P = N = 1000$. Fig B.3 (left) shows 2D projections in the plane $(\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_{10})$ of the (centered) minimum ℓ_2 norm solutions $\mathbf{w}^*_{\mathcal{S}} - \mathbb{E}_{\mathcal{S}} \mathbf{w}^*_{\mathcal{S}}$, for a pool of 100 training (sub)samples $\mathcal{S}$ of size $n = 50$, for increasing values of the scaling factor c. As c approaches 1, the solutions begin to scatter in a very anisotropic way in parameter space; as shown in Fig B.3 (right), the complexity bound (42) based on the ℓ_2 norm, i.e $A = \mathrm{Id}$ (blue plot), becomes increasingly loose and fails to reflect the shape of the test error.

To find a more meaningful capacity measure, Prop B.2 suggests optimizing the bound (10) with $M_A = \|\mathbf{w}^*\|_A$, over a given class of rescaling matrices A. We give an example of this in the following Proposition.

Proposition B.7. Consider the class of matrices $A_\nu = \sum_{j=1}^n \sqrt{\nu_j} \boldsymbol{v}_j \boldsymbol{v}_j^\top + \mathrm{Id}_{\mathrm{span}\{\boldsymbol{v}\}^\perp}$, which act as mere rescaling of the singular values of the feature matrix. Any minimizer of the upper bound (42) for the mininum ℓ^2-norm interpolator takes the form

$$\nu_j^* = \kappa \frac{\sqrt{\lambda_j}}{|\boldsymbol{v}_j^\top \mathbf{w}^*|} = \kappa \frac{\lambda_j}{|\boldsymbol{u}_j^\top \boldsymbol{y}|} \tag{64}$$

[1]We used RBFsampler of scikit-learn, which implements a variant of Random Kitchen Sinks (Rahimi & Recht, 2007) to approximate the feature map of a RBF kernel with parameter $\gamma = 1$.

where $\kappa > 0$ is a constant independent of j.

PROOF. From (63) and the definition of A_ν, we first write

$$\|\mathbf{w}^*\|^2_{A_\nu} = \sum_{j=1}^{n} \frac{\nu_j}{\lambda_j}(\boldsymbol{u}_j^\top \boldsymbol{y})^2, \quad \mathrm{Tr}\boldsymbol{K}_{A_\nu} = \sum_{j=1}^{n} \frac{\lambda_j}{\nu_j} \tag{65}$$

The product of the above two terms has the critical points ν_j^*, $j = 1 \cdots n$ which satisfy

$$\frac{(\boldsymbol{u}_j^\top \boldsymbol{y})^2}{\lambda_j}\mathrm{Tr}\boldsymbol{K}_{A_\nu} - \frac{\lambda_j}{\nu_j^{*2}}\|\mathbf{w}^*\|^2_{A_\nu} = 0 \tag{66}$$

giving the desired result $\nu_j^* \propto \lambda_j/|\boldsymbol{u}_j^\top \boldsymbol{y}|$. □

In the context of Proposition B.7, we see that the optimal norm $\|\cdot\|_{A_{\nu^*}}$ depends both on the feature geometry – through the singular values – and on the task – through the labels –. As shown in Fig 1 (right, red plot), in the above RBF feature setting, the resulting optimal bound on the Radecher complexity has a much nicer behaviour than the standard bound based on the ℓ^2 norm.[2]

B.2.6. SuperNat: Proof of Prop 4.3

Prop. 4.3 is a *local* version of Prop B.7, where the feature rescaling factors are applied at each step of the training algorithm. The procedure is described in Fig 4.5 (left); the term to be optimized shows up in Step 2. With the chosen class of matrices described in Prop 4.3, the action $\boldsymbol{\Phi}_t \to A_\nu^{-1}\boldsymbol{\Phi}_t$ merely rescale its singular values $\lambda_{jt} \to \lambda_{jt}/\nu_j$, leaving its singular vectors $\boldsymbol{u}_j, \boldsymbol{v}_j$ unchanged.

Proposition B.8 (Prop 4.3 restated)**.** For the class of rescaling matrices A_ν defined in Prop B.7, any minimizer in Step 2 in Fig 4.5, where $\delta\mathbf{w}_{\mathrm{GD}} = -\eta\nabla_{\mathbf{w}}L$, takes the form

$$\nu_{jt}^* = \kappa\frac{1}{|\boldsymbol{u}_j^\top \nabla_{\mathbf{f_w}} L|} \tag{67}$$

where $\kappa > 0$ is a constant independent of j.

PROOF. Using the chain rule and the SVD of the feature map Φ_t we write the gradient descent updates at iteration t of SuperNat as

$$\delta\mathbf{w}_{\mathrm{GD}} = -\eta\boldsymbol{\Phi}_t^\top \nabla_{\mathbf{f_w}} L \tag{68}$$

$$= -\eta \textstyle\sum_{j=1}^{n} \sqrt{\lambda_{jt}}(\boldsymbol{u}_j^\top \nabla_{\mathbf{f_w}} L)\, \boldsymbol{v}_j, \tag{69}$$

[2]Note however that, since the optimal norm depends on the sample set $\mathcal{S}$, the resulting complexity bound does not directly yield a high probability bound on the generalization error as in (38). The more thorough analysis, which requires promoting (38) to uniform bounds over the choice of matrix A, is left for future work.

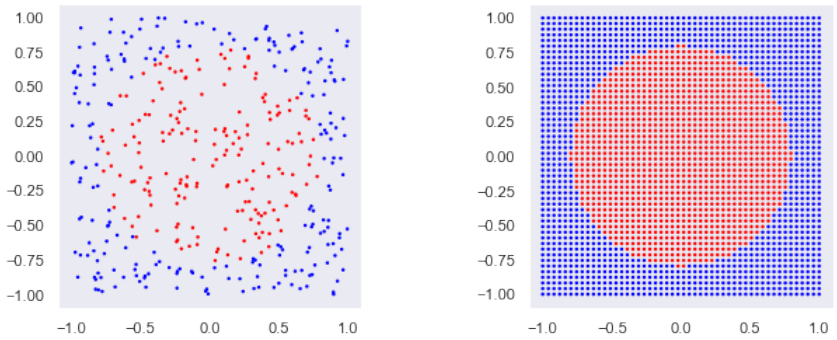


Fig. B.4. Disk dataset. **Left**: Training set of $n = 500$ points $(\mathbf{x}_i, y_i)$ where $\mathbf{x} \sim \text{Unif}[-1,1]^2$, $y_i = 1$ if $\|x_i\|_2 \leq r = \sqrt{2/\pi}$ and -1 otherwise. **Right**: Large test sample (2500 points forming a 50×50 grid) used to evaluate the tangent kernel.

From the definition of A_ν, we then spell out

$$\|\delta \mathbf{w}_{\text{GD}}\|_{A_\nu}^2 = \eta^2 \sum_{j=1}^{n} (\nu_j \lambda_j)(\boldsymbol{u}_j^\top \nabla_{\mathbf{f_w}} L)^2, \quad \|A_\nu^{-1} \boldsymbol{\Phi}_t\|_F := \text{Tr} \boldsymbol{K}_{tA_\nu} = \sum_{j=1}^{n} \frac{\lambda_j}{\nu_j} \tag{70}$$

The product of the above two terms has the critical points ν_j^*, $j = 1 \cdots n$ which satisfy

$$\lambda_j (\boldsymbol{u}_j^\top \nabla_{\mathbf{f_w}} L)^2 \text{Tr} \boldsymbol{K}_{A_\nu} - \frac{\lambda_j}{\nu_j^{*2}} \|\delta \mathbf{w}_{\text{GD}}\|_{A_\nu}^2 = 0 \tag{71}$$

giving the desired result $\nu_j^* \propto 1/|\boldsymbol{u}_j^\top \nabla_{\mathbf{f_w}} L|$. □

B.3. Additional experiments

B.3.1. Synthetic Experiment: Fig. 4.1

To visualize the adaptation of the tangent kernel to the task during training, we perform the following synthetic experiment. We train a 6-layer deep 256-unit wide MLP on $n = 500$ points of the Disc dataset $(\mathbf{x}, y)$ where $\mathbf{x} \sim \text{Unif}[-1,1]^2$ and $y(\mathbf{x}) = \pm 1$ depending on whether is within the disk of center 0 and radius $\sqrt{2/\pi}$, see Fig B.4. Fig. 4.1 in the main text shows visualizations of eigenfunctions sampled using a grid of $N = 2500$ points on the square, and ranked in non-increasing order of the spectrum $\lambda_1 \geq \cdots \geq \lambda_N$. After a number of iterations, we begin to see the class structure (e.g. boundary circle) emerge in the top eigenfunctions. We note also an increasingly fast spectrum decay (e.g $\lambda_{20}/\lambda_1 = 1.5\%$ at iteration 0 and 0.2% at iteration 2000). The interpretation is that the kernel stretches in directions of high correlation with the labels.

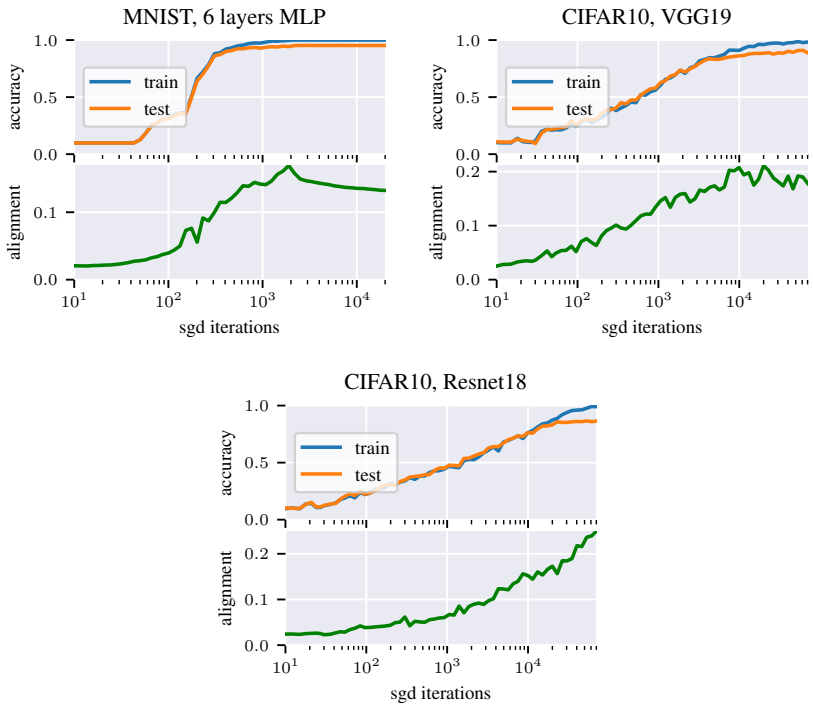


Fig. B.5. Evolution of the CKA between the tangent kernel and the class label kernel $K_Y = YY^T$ measured on a held-out test set for different architectures: **(left)** 6 layers of 80 hidden units MLP on MNIST **(middle)** VGG19 on CIFAR10 **(right)** Resnet18 on CIFAR10. We observe an increase of the alignment to the target function.

B.3.2. More Alignment Plots

Varying datasets and architectures: Fig B.5.

Uncentered kernel Experiments: Fig B.6. The evolution of the alignment to the *uncentered* kernel, in order to assess whether this effect is consistent when removing centering. The experimental details are the same as in the main text; we also observe a similar increase of the alignment as training progresses.

B.3.3. Effect of depth on alignment

In order to study the influence of the architecture on the alignment effect, we measure the CKA for different networks and different initialization as we increase the depth. The results in Fig B.7 suggest that the alignment effect is magnified as depth increases. We also observe that the ratio of the maximum alignment between easy and difficult examples is increased with depth, but stays high for a smaller number of iterations.

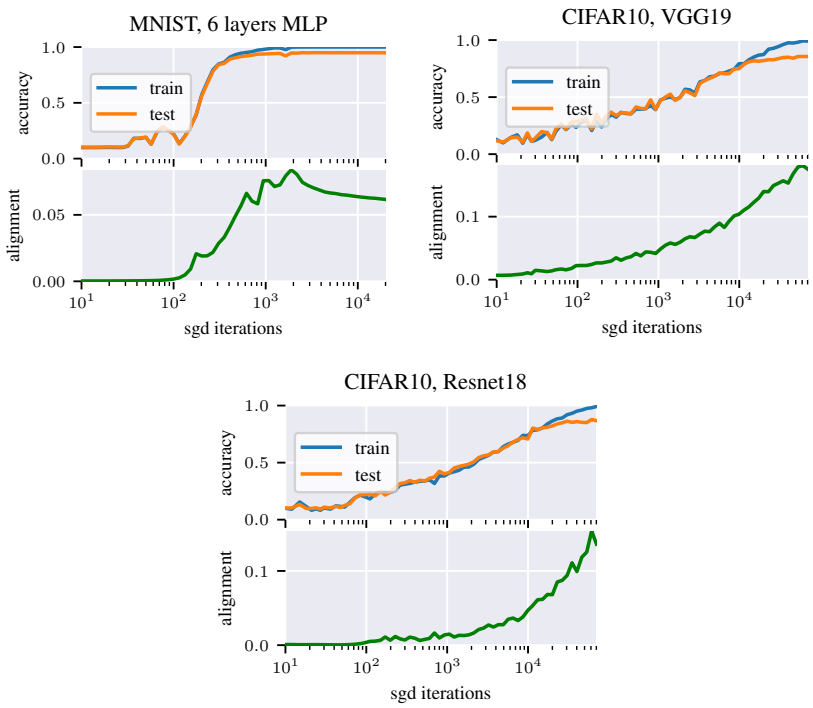


Fig. B.6. Same as figure B.5 but without centering the kernel. Evolution of the uncentered kernel alignment between the tangent kernel and the class label kernel $K_Y = YY^T$ measured on a held-out test set for different architectures: **(left)** 6 layers of 80 hidden units MLP on MNIST **(middle)** VGG19 on CIFAR10 **(right)** Resnet18 on CIFAR10. We observe an increase of the alignment to the target function.

B.3.4. Spectrum Plots with lower learning rate : Fig. B.8

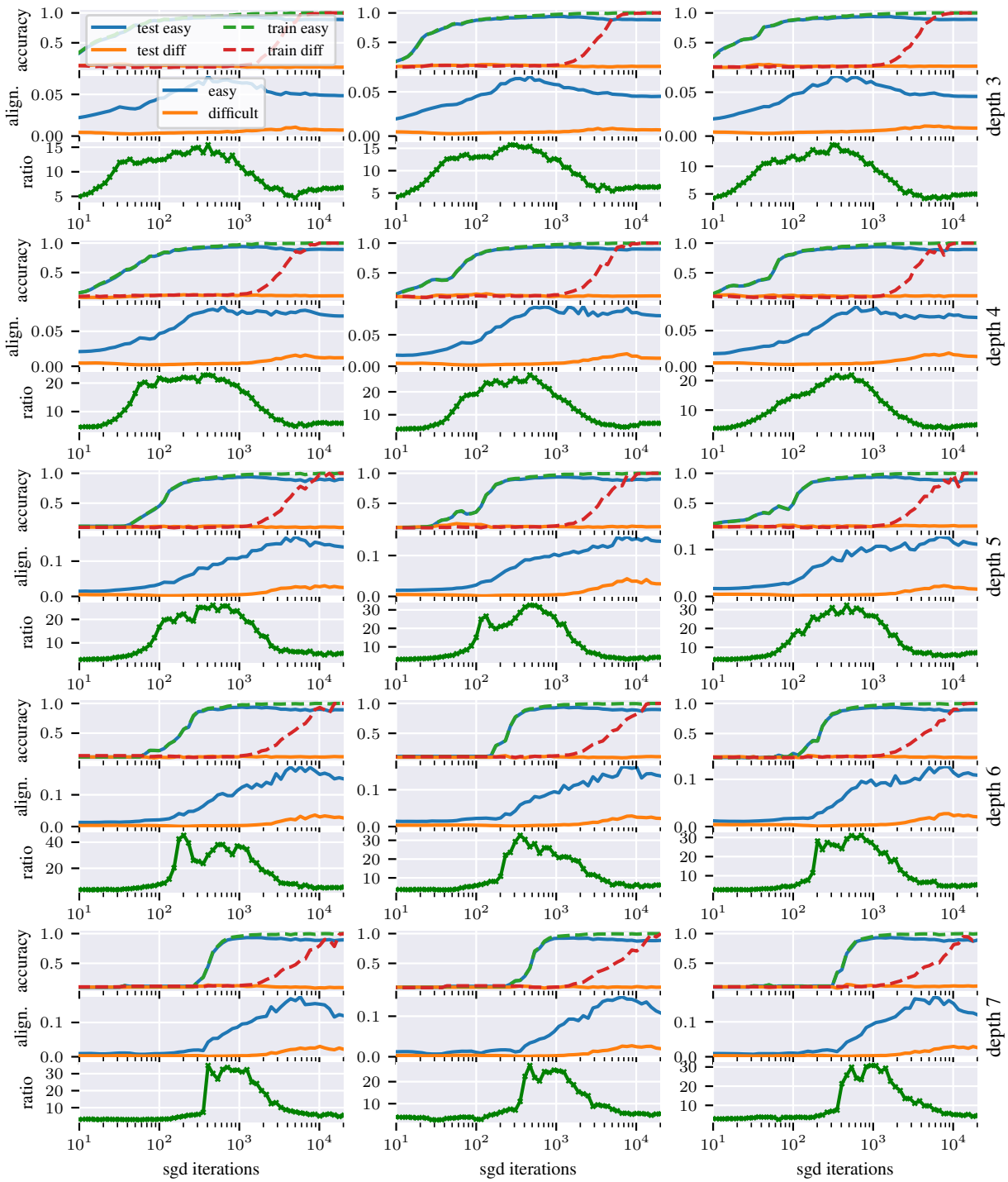


Fig. B.7. Effect of depth on alignment. 10.000 MNIST examples with 1000 random labels MNIST examples trained with learning rate=0.01, momentum=0.9 and batch size=100 for MLP with hidden layers size 60 and **(in rows)** varying depths **(in columns)** varying random initialization/minibatch sampling. As we increase the depth, the alignment starts increasing later in training and increases faster; and the ratio between easy and difficult alignments reaches a higher value.

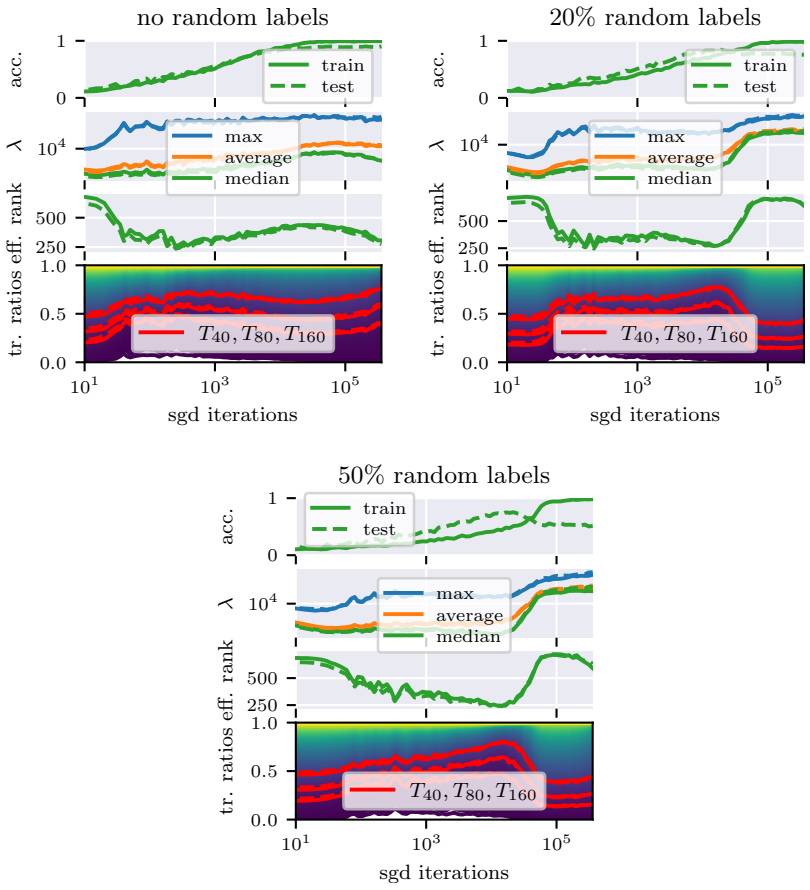


Fig. B.8. Evolution of tangent kernel spectrum, effective rank and trace ratios of a VGG19 trained by SGD with batch size 100, learning rate 0.003 and momentum 0.9 on dataset **(left)** CIFAR10 and **(right)** CIFAR10 with 50% random labels. We highlight the top 40, 80 and 160 trace ratios in **red**.

Appendix C

Mutual Information Neural Estimation: Supplementary Material

C.1. Experimental Details

C.1.1. Adaptive Clipping

Here we assume we are in the context of GANs described in Sections 5.1 and 5.2, where the mutual information shows up as a regularizer in the generator objective.

Notice that the generator is updated by two gradients. The first gradient is that of the generator's loss, $\mathcal{L}_g$ with respect to the generator's parameters θ, $g_u := \frac{\partial \mathcal{L}_g}{\partial \theta}$. The second flows from the mutual information estimate to the generator, $g_m := -\frac{\partial \widehat{I(X;Z)}}{\partial \theta}$. If left unchecked, because mutual information is unbounded, the latter can overwhelm the former, leading to a failure mode of the algorithm where the generator puts all of its attention on maximizing the mutual information and ignores the adversarial game with the discriminator. We propose to adaptively clip the gradient from the mutual information so that its Frobenius norm is at most that of the gradient from the discriminator. Defining g_a to be the adapted gradient following from the statistics network to the generator, we have,

$$g_a = \min(\|g_u\|, \|g_m\|) \frac{g_m}{\|g_m\|}. \tag{1}$$

Note that adaptive clipping can be considered in any situation where MINE is to be maximized.

C.1.2. GAN+MINE: Spiral and 25-gaussians

In this section we state the details of experiments supporting mode dropping experiments on the spiral and 25-Gaussians dataset. For both the datasets we use 100,000 examples

sampled from the target distributions, using a standard deviation of 0.05 in the case of 25-gaussians, and using additive noise for the spiral. The generator for the GAN consists of two fully connected layers with 500 units in each layer with batch-normalization (Ioffe & Szegedy, 2015) and Leaky-ReLU as activation function as in Dumoulin et al. (2016). The discriminator and statistics networks have three fully connected layers with 400 units each. We use the Adam (Kingma & Ba, 2014) optimizer with a learning rate of 0.0001. Both GAN baseline and GAN+MINE were trained for 5,000 iterations with a mini batch-size of 100.

C.1.3. GAN+MINE: Stacked-MNIST

Here we describe the experimental setup and architectural details of stacked-MNIST task with GAN+MINE. We compare to the exact same experimental setup followed and reported in PacGANLin et al. (2017) and VEEGANSrivastava et al. (2017). We use a pre-trained classifier to classify generated samples on each of the three stacked channels. Evaluation is done on 26,000 test samples as followed in the baselines. We train GAN+MINE for 50 epochs on 128,000 samples. Details for generator and discriminator networks are given below in the tableC.1 and tableC.2. Specifically the statistics network has the same architecture as discriminator in DCGAN with ELU (Clevert et al., 2015) as activation function for the individual layers and without batch-normalization as highlighted in Table C.3. In order to condition the statistics network on the z variable, we use linear MLPs at each layer, whose output are reshaped to the number of feature maps. The linear MLPs output is then added as a dynamic bias.

Generator				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input $\boldsymbol{z} \sim \mathcal{U}(-1,1)^{100}$	100			
Fully-connected	2*2*512			ReLU
Transposed convolution	4*4*256	$5*5$	2	ReLU
Transposed convolution	7*7*128	$5*5$	2	ReLU
Transposed convolution	14*14*64	$5*5$	2	ReLU
Transposed convolution	28*28*3	$5*5$	2	Tanh

Table C.1. Generator network for Stacked-MNIST experiment using GAN+MINE.

Discriminator				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input x	$28 * 28 * 3$			
Convolution	14*14*64	$5 * 5$	2	ReLU
Convolution	7*7*128	$5 * 5$	2	ReLU
Convolution	4*4*256	$5 * 5$	2	ReLU
Convolution	2*2*512	$5 * 5$	2	ReLU
Fully-connected	1	1	Valid	Sigmoid

Table C.2. Discriminator network for Stacked-MNIST experiment.

Statistics Network				
Layer	number of outputs	kernel size	stride	activation function
Input x, z				
Convolution	14*14*16	$5 * 5$	2	ELU
Convolution	7*7*32	$5 * 5$	2	ELU
Convolution	4*4*64	$5 * 5$	2	ELU
Flatten	-	-	-	-
Fully-Connected	1024	1	Valid	None
Fully-Connected	1	1	Valid	None

Table C.3. Statistics network for Stacked-MNIST experiment.

C.1.4. ALI+MINE: MNIST and CelebA

In this section we state the details of experimental setup and the network architectures used for the task of improving reconstructions and representations in bidirectional adversarial models with MINE. The generator and discriminator network architectures along with the hyper parameter setup used in these tasks are similar to the ones used in DCGAN (Radford et al., 2015).

Statistics network conditioning on the latent code was done as in the Stacked-MNIST experiments. We used Adam as the optimizer with a learning rate of 0.0001. We trained the model for a total of 35,000 iterations on CelebA and 50,000 iterations on MNIST, both with a mini batch-size of 100.

Encoder				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input $[\boldsymbol{x}, \boldsymbol{\epsilon}]$	28*28*129			
Convolution	14*14*64	$5 * 5$	2	ReLU
Convolution	7*7*128	$5 * 5$	2	ReLU
Convolution	4*4*256	$5 * 5$	2	ReLU
Convolution	256	$4 * 4$	Valid	ReLU
Fully-connected	128	-	-	None

Table C.4. Encoder network for bi-directional models on MNIST. $\boldsymbol{\epsilon} \sim \mathcal{N}_{128}(0, I)$.

Decoder				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input $\boldsymbol{z}$	128			
Fully-connected	4*4*256			ReLU
Transposed convolution	7*7*128	$5 * 5$	2	ReLU
Transposed convolution	14*14*64	$5 * 5$	2	ReLU
Transposed convolution	28*28*1	$5 * 5$	2	Tanh

Table C.5. Decoder network for bi-directional models on MNIST. $\boldsymbol{z} \sim \mathcal{N}_{256}(0, I)$

Discriminator				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input x	$28 * 28 * 3$			
Convolution	14*14*64	$5 * 5$	2	LearkyReLU
Convolution	7*7*128	$5 * 5$	2	LeakyReLU
Convolution	4*4*256	$5 * 5$	2	LeakyReLU
Flatten	-	-	-	
Concatenate $\boldsymbol{z}$	-	-	-	
Fully-connected	1024	-	-	LeakyReLU
Fully-connected	1	-	-	Sigmoid

Table C.6. Discriminator network for bi-directional models experiments MINE on MNIST.

Statistics Network				
Layer	number of outputs	kernel size	stride	activation function
Input $\boldsymbol{x}, \boldsymbol{z}$				
Convolution	14*14*64	$5 * 5$	2	LeakyReLU
Convolution	7*7*128	$5 * 5$	2	LeakyReLU
Convolution	4*4*256	$5 * 5$	2	LeakyReLU
Flatten	-	-	-	-
Fully-connected	1	-	-	None

Table C.7. Statistics network for bi-directional models using MINE on MNIST.

Encoder				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input $[\boldsymbol{x}, \boldsymbol{\epsilon}]$	64*64*259			
Convolution	32*32*64	$5 * 5$	2	ReLU
Convolution	16*16*128	$5 * 5$	2	ReLU
Convolution	8*8*256	$5 * 5$	2	ReLU
Convolution	4*4*512	$5 * 5$	2	ReLU
Convolution	512	$4 * 4$	Valid	ReLU
Fully-connected	256	-	-	None

Table C.8. Encoder network for bi-directional models on CelebA. $\boldsymbol{\epsilon} \sim \mathcal{N}_{256}(0, I)$.

Decoder				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input $\boldsymbol{z} \sim \mathcal{N}_{256}(0, I)$	256			
Fully-Connected	4*4*512	-	-	ReLU
Transposed convolution	8*8*256	$5 * 5$	2	ReLU
Transposed convolution	16*16*128	$5 * 5$	2	ReLU
Transposed convolution	32*32*64	$5 * 5$	2	ReLU
Transposed convolution	64*64*3	$5 * 5$	2	Tanh

Table C.9. Decoder network for bi-directional model(ALI, ALICE) experiments using MINE on CelebA.

Discriminator				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input x	$64 * 64 * 3$			
Convolution	32*32*64	$5 * 5$	2	LearkyReLU
Convolution	16*16*128	$5 * 5$	2	LeakyReLU
Convolution	8*8*256	$5 * 5$	2	LeakyReLU
Convolution	4*4*512	$5 * 5$	2	LeakyReLU
Flatten	-	-	-	
Concatenate $\boldsymbol{z}$	-	-	-	
Fully-connected	1024	-	-	LeakyReLU
Fully-connected	1	-	-	Sigmoid

Table C.10. Discriminator network for bi-directional models on CelebA.

Statistics Network				
Layer	number of outputs	kernel size	stride	activation function
Input $\boldsymbol{x}, \boldsymbol{z}$				
Convolution	32*32*16	$5 * 5$	2	ELU
Convolution	16*16*32	$5 * 5$	2	ELU
Convolution	8*8*64	$5 * 5$	2	ELU
Convolution	4*4*128	$5 * 5$	2	ELU
Flatten	-	-	-	-
Fully-connected	1	-	-	None

Table C.11. Statistics network for bi-directional models on CelebA.

C.1.5. Information bottleneck with MINE

In this section we outline the network details and hyper-parameters used for the information bottleneck task using MINE. To keep comparison fair all hyperparameters and architectures are those outlined in Alemi et al. (2016). The statistics network is shown, a two layer MLP with additive noise at each layer and 512 ELUs (Clevert et al., 2015) activations, is outlined in tableC.12.

Statistics Network		
Layer	number of outputs	activation function
input $[\boldsymbol{x}, \boldsymbol{z}]$		
Gaussian noise(std=0.3)	-	-
dense layer	512	ELU
Gaussian noise(std=0.5)	-	-
dense layer	512	ELU
Gaussian noise(std=0.5)	-	-
dense layer	1	None

Table C.12. Statistics network for Information-bottleneck experiments on MNIST.

C.2. Proofs

C.2.1. Donsker-Varadhan Representation

Theorem C.1 (Theorem 5.1 restated)**.** The KL divergence admits the following dual representation:

$$D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T]), \tag{2}$$

where the supremum is taken over all functions T such that the two expectations are finite.

Proof. A simple proof goes as follows. For a given function T, consider the Gibbs distribution $\mathbb{G}$ defined by $d\mathbb{G} = \frac{1}{Z} e^T d\mathbb{Q}$, where $Z = \mathbb{E}_{\mathbb{Q}}[e^T]$. By construction,

$$\mathbb{E}_{\mathbb{P}}[T] - \log Z = \mathbb{E}_{\mathbb{P}}\left[\log \frac{d\mathbb{G}}{d\mathbb{Q}}\right] \tag{3}$$

Let Δ be the gap,

$$\Delta := D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) - \big(\mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T])\big) \tag{4}$$

Using Eqn 3, we can write Δ as a KL-divergence:

$$\Delta = \mathbb{E}_{\mathbb{P}}\left[\log \frac{d\mathbb{P}}{d\mathbb{Q}} - \log \frac{d\mathbb{G}}{d\mathbb{Q}}\right] = \mathbb{E}_{\mathbb{P}} \log \frac{d\mathbb{P}}{d\mathbb{G}} = D_{KL}(\mathbb{P} \,||\, \mathbb{G}) \tag{5}$$

The positivity of the KL-divergence gives $\Delta \geq 0$. We have thus shown that for any T,

$$D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) \geq \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T]) \tag{6}$$

and the inequality is preserved upon taking the supremum over the right-hand side. Finally, the identity (5) also shows that this bound is *tight* whenever $\mathbb{G} = \mathbb{P}$, namely for optimal functions T^* taking the form $T^* = \log \frac{d\mathbb{P}}{d\mathbb{Q}} + C$ for some constant $C \in \mathbb{R}$. □

C.2.2. Consistency Proofs

This section presents the proofs of the Lemma and consistency theorem stated in the consistency in Section 3.3.1.

In what follows, we assume that the input space $\Omega = \mathcal{X} \times \mathcal{Z}$ is a compact domain of $\mathbb{R}^d$, and all measures are absolutely continuous with respect to the Lebesgue measure. We will restrict to families of feedforward functions with continuous activations, with a single output neuron, so that a given architecture defines a continuous mapping $(\omega, \theta) \to T_\theta(\omega)$ from $\Omega \times \Theta$ to $\mathbb{R}$.

To avoid unnecessary heavy notation, we denote $\mathbb{P} = \mathbb{P}_{XZ}$ and $\mathbb{Q} = \mathbb{P}_X \otimes \mathbb{P}_Z$ for the joint distribution and product of marginals, and $\mathbb{P}_n, \mathbb{Q}_n$ for their empirical versions. We will use the notation $\hat{I}(T)$ for the quantity:

$$\hat{I}(T) = \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T]) \tag{7}$$

so that $I_\Theta(X, Z) = \sup_{\theta \in \Theta} \hat{I}(T_\theta)$.

Lemma C.1 (Lemma 5.2 restated)**.** *Let $\eta > 0$. There exists a family of neural network functions T_θ with parameters θ in some compact domain $\Theta \subset \mathbb{R}^k$, such that*

$$|I(X, Z) - I_\Theta(X,Z)| \leq \eta \tag{8}$$

where

$$I_\Theta(X, Z) = \sup_{\theta \in \Theta} \mathbb{E}_{\mathbb{P}_{XZ}}[T_\theta] - \log(\mathbb{E}_{\mathbb{P}_X \otimes \mathbb{P}_Z}[e^{T_\theta}]) \tag{9}$$

Proof. Let $T^* = \log \frac{d\mathbb{P}}{d\mathbb{Q}}$. By construction, T^* satisfies:

$$\mathbb{E}_{\mathbb{P}}[T^*] = I(X, Z), \qquad \mathbb{E}_{\mathbb{Q}}[e^{T^*}] = 1 \tag{10}$$

For a function T, the (positive) gap $I(X, Z) - \hat{I}(T)$ can be written as

$$I(X, Z) - \hat{I}(T) = \mathbb{E}_{\mathbb{P}}[T^* - T] + \log \mathbb{E}_{\mathbb{Q}}[e^T] \leq \mathbb{E}_{\mathbb{P}}[T^* - T] + \mathbb{E}_{\mathbb{Q}}[e^T - e^{T^*}] \tag{11}$$

where we used the inequality $\log x \leq x - 1$.

Fix $\eta > 0$. We first consider the case where T^* is *bounded* from above by a constant M. By the universal approximation theorem (see corollary 2.2 of Hornik (1989)[1]), we may choose a feedforward network function $T_{\hat{\theta}} \leq M$ such that

$$\mathbb{E}_{\mathbb{P}}|T^* - T_{\hat{\theta}}| \leq \frac{\eta}{2} \quad \text{and} \quad \mathbb{E}_{\mathbb{Q}}|T^* - T_{\hat{\theta}}| \leq \frac{\eta}{2} e^{-M} \tag{12}$$

[1]Specifically, the argument relies on the density of feedforward network functions in the space $L^1(\Omega, \mu)$ of integrable functions with respect the measure $\mu = \mathbb{P} + \mathbb{Q}$.

Since exp is Lipschitz continuous with constant e^M on $(-\infty, M]$, we have

$$\mathbb{E}_{\mathbb{Q}}|e^{T^*} - e^{T_{\hat{\theta}}}| \leq e^M \, \mathbb{E}_{\mathbb{Q}}|T^* - T_{\hat{\theta}}| \leq \frac{\eta}{2} \tag{13}$$

From Equ 11 and the triangular inequality, we then obtain:

$$|I(X,Z) - \hat{I}(T_{\hat{\theta}})| \leq \mathbb{E}_{\mathbb{P}}|T^* - T_{\hat{\theta}}| + \mathbb{E}_{\mathbb{Q}}|e^{T^*} - e^{T_{\hat{\theta}}}| \leq \frac{\eta}{2} + \frac{\eta}{2} \leq \eta \tag{14}$$

In the general case, the idea is to partition Ω in two subset $\{T^* > M\}$ and $\{T^* \leq M\}$ for a suitably chosen large value of M. For a given subset $S \subset \Omega$, we will denote by $\mathbb{1}_S$ its characteristic function, $\mathbb{1}_S(\omega) = 1$ if $\omega \in S$ and 0 otherwise. T^* is integrable with respect to $\mathbb{P}$[2], and e^{T^*} is integrable with respect to $\mathbb{Q}$, so by the dominated convergence theorem, we may choose M so that the expectations $\mathbb{E}_{\mathbb{P}}[T^* \mathbb{1}_{T^*>M}]$ and $\mathbb{E}_{\mathbb{Q}}[e^{T^*} \mathbb{1}_{T^*>M}]$ are lower than $\eta/4$. Just like above, we then use the universal approximation theorem to find a feed forward network function $T_{\hat{\theta}}$, which we can assume without loss of generality to be upper-bounded by M, such that

$$\mathbb{E}_{\mathbb{P}}|T^* - T_{\hat{\theta}}| \leq \frac{\eta}{2} \quad \text{and} \quad \mathbb{E}_{\mathbb{Q}}|T^* - T_{\hat{\theta}}|\mathbb{1}_{T^* \leq M} \leq \frac{\eta}{4} e^{-M} \tag{15}$$

We then write

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}}[e^{T^*} - e^{T_{\hat{\theta}}}] &= \mathbb{E}_{\mathbb{Q}}[(e^{T^*} - e^{T_{\hat{\theta}}})\mathbb{1}_{T^* \leq M}] + \mathbb{E}_{\mathbb{Q}}[(e^{T^*} - e^{T_{\hat{\theta}}})\mathbb{1}_{T^*>M}] \\
&\leq e^M \mathbb{E}_{\mathbb{Q}}[|T^* - T_{\hat{\theta}}|\mathbb{1}_{T^* \leq M}] + \mathbb{E}_{\mathbb{Q}}[e^{T^*}\mathbb{1}_{T^*>M}] \\
&\leq \frac{\eta}{4} + \frac{\eta}{4} \qquad (16) \\
&\leq \frac{\eta}{2} \qquad (17)
\end{aligned}$$

where the inequality in the second line arises from the convexity and positivity of exp. Eqns. 15 and 16, together with the triangular inequality, lead to Eqn. 14, which proves the Lemma.

□

Lemma C.2 (Lemma 5.3 restated)**.** *Let $\eta > 0$. Given a family $\mathcal{F}$ of neural network functions T_θ with parameters θ in some compact domain $\Theta \subset \mathbb{R}^k$, there exists $N \in \mathbb{N}$ such that*

$$\forall n \geq N, \quad \Pr\left(|\widehat{I(X;Z)}_n - I_{\mathcal{F}}(X,Z)| \leq \eta\right) = 1 \tag{18}$$

Proof. We start by using the triangular inequality to write,

$$|\widehat{I(X;Z)}_n - \sup_{T_\theta \in \mathcal{F}} \hat{I}(T_\theta)| \leq \sup_{T_\theta \in \mathcal{F}} |\mathbb{E}_{\mathbb{P}}[T_\theta] - \mathbb{E}_{\mathbb{P}_n}[T_\theta]| + \sup_{T_\theta \in \mathcal{F}} |\log \mathbb{E}_{\mathbb{Q}}[e^{T_\theta}] - \log \mathbb{E}_{\mathbb{Q}_n}[e^{T_\theta}]| \tag{19}$$

[2]This can be seen from the identity (Györfi & van der Meulen, 1987)

$$\mathbb{E}_{\mathbb{P}}\left|\log \frac{d\mathbb{P}}{d\mathbb{Q}}\right| \leq D_{KL}(\mathbb{P} \,||\, \mathbb{Q}) + 4\sqrt{D_{KL}(\mathbb{P} \,||\, \mathbb{Q})}$$

The continuous function $(\theta, \omega) \to T_\theta(\omega)$, defined on the compact domain $\Theta \times \Omega$, is bounded. So the functions T_θ are uniformly bounded by a constant M, i.e $|T_\theta| \leq M$ for all $\theta \in \Theta$. Since log is Lipschitz continuous with constant e^M in the interval $[e^{-M}, e^M]$, we have

$$|\log \mathbb{E}_{\mathbb{Q}}[e^{T_\theta}] - \log \mathbb{E}_{\mathbb{Q}_n}[e^{T_\theta}]| \leq e^M |\mathbb{E}_{\mathbb{Q}}[e^{T_\theta}] - \mathbb{E}_{\mathbb{Q}_n}[e^{T_\theta}]| \tag{20}$$

Since Θ is compact and the feedforward network functions are continuous, the families of functions T_θ and e^{T_θ} satisfy the uniform law of large numbers (Van de Geer, 2000). Given $\eta > 0$ we can thus choose $N \in \mathbb{N}$ such that $\forall n \geq N$ and with probability one,

$$\sup_{T_\theta \in \mathcal{F}} |\mathbb{E}_{\mathbb{P}}[T_\theta] - \mathbb{E}_{\mathbb{P}_n}[T_\theta] \leq \frac{\eta}{2} \quad \text{and} \quad \sup_{T_\theta \in \mathcal{F}} |\mathbb{E}_{\mathbb{Q}}[e^{T_\theta}] - \mathbb{E}_{\mathbb{Q}_n}[e^{T_\theta}]| \leq \frac{\eta}{2} e^{-M} \tag{21}$$

Together with Eqns. 19 and 20, this leads to

$$|\widehat{I(X;Z)}_n - \sup_{T_\theta \in \mathcal{F}} \hat{I}(T_\theta)| \leq \frac{\eta}{2} + \frac{\eta}{2} = \eta \tag{22}$$

□

Theorem C.2 (Theorem 5.4 restated)**.** MINE is strongly consistent.

Proof. Let $\epsilon > 0$. We apply the two Lemmas to find a a family of neural network function $\mathcal{F}$ and $N \in \mathbb{N}$ such that (8) and (18) hold with $\eta = \epsilon/2$. By the triangular inequality, for all $n \geq N$ and with probability one, we have:

$$|I(X,Z) - \widehat{I(X;Z)}_n| \quad \leq \quad |I(X,Z) - \sup_{T_\theta \in \mathcal{F}} \hat{I}(T_\theta)| + |\widehat{I(X;Z)}_n - I_{\mathcal{F}}(X,Z)| \leq \quad \epsilon \tag{23}$$

which proves consistency. □

C.2.3. Sample complexity proof

Theorem C.3 (Theorem 5.5 restated)**.** Assume that the functions T_θ in $\mathcal{F}$ are L-Lipschitz with respect to the parameters θ; and that both T_θ and e^{T_θ} are M-bounded (i.e., $|T_\theta|, e^{T_\theta} \leq M$). The domain $\Theta \subset \mathbb{R}^d$ is bounded, so that $\|\theta\| \leq K$ for some constant K. Given any values ϵ,δ of the desired accuracy and confidence parameters, we have,

$$\Pr\left(|\widehat{I(X;Z)}_n - I_{\mathcal{F}}(X,Z)| \leq \epsilon\right) \geq 1 - \delta \tag{24}$$

whenever the number n of samples satisfies

$$n \geq \frac{2M^2(d \log(16KL\sqrt{d}/\epsilon) + 2dM + \log(2/\delta))}{\epsilon^2} \tag{25}$$

PROOF. The assumptions of Lemma 5.3 apply, so let us begin with Eqns. 19 and 20. By the Hoeffding inequality, for all function f,

$$\Pr\left(|\mathbb{E}_{\mathbb{Q}}[f] - \mathbb{E}_{\mathbb{Q}_n}[f]| \ > \frac{\epsilon}{4}\right) \leq 2\exp\left(-\frac{\epsilon^2 n}{2M^2}\right) \tag{26}$$

To extend this inequality to a uniform inequality over *all* functions T_θ and e^{T_θ}, the standard technique is to choose a minimal cover of the domain $\Theta \subset \mathbb{R}^d$ by a finite set of small balls of radius η, $\Theta \subset \cup_j B_\eta(\theta_j)$, and to use the union bound. The minimal cardinality of such covering is bounded by the *covering number* $N_\eta(\Theta)$ of Θ, known to satisfy(Shalev-Schwartz & Ben-David, 2014)

$$N_\eta(\Theta) \leq \left(\frac{2K\sqrt{d}}{\eta}\right)^d \tag{27}$$

Successively applying a union bound in Eqn 26 with the set of functions $\{T_{\theta_j}\}_j$ and $\{e^{T_{\theta_j}}\}_j$ gives

$$\Pr\left(\max_j |\mathbb{E}_{\mathbb{Q}}[T_{\theta_j}] - \mathbb{E}_{\mathbb{Q}_n}[T_{\theta_j}]| > \frac{\epsilon}{4}\right) \leq 2N_\eta(\Theta)\exp\left(-\frac{\epsilon^2 n}{2M^2}\right) \tag{28}$$

and

$$\Pr\left(\max_j |\mathbb{E}_{\mathbb{Q}}[e^{T_{\theta_j}}] - \mathbb{E}_{\mathbb{Q}_n}[e^{T_{\theta_j}}]| > \frac{\epsilon}{4}\right) \leq 2N_\eta(\Theta)\exp\left(-\frac{\epsilon^2 n}{2M^2}\right) \tag{29}$$

We now choose the ball radius to be $\eta = \frac{\epsilon}{8L}e^{-2M}$. Solving for n the inequation,

$$2N_\eta(\Theta)\exp\left(-\frac{\epsilon^2 n}{2M^2}\right) \leq \delta \tag{30}$$

we deduce from Eqn 28 that, whenever Eqn 25 holds, with probability at least $1-\delta$, for all $\theta \in \Theta$,

$$\begin{aligned} |\mathbb{E}_{\mathbb{Q}}[T_\theta] - \mathbb{E}_{\mathbb{Q}_n}[T_\theta]| \ &\leq |\mathbb{E}_{\mathbb{Q}}[T_\theta] - \mathbb{E}_{\mathbb{Q}}[T_{\theta_j}]| + |\mathbb{E}_{\mathbb{Q}}[T_{\theta_j}] - \mathbb{E}_{\mathbb{Q}_n}[T_{\theta_j}]| + |\mathbb{E}_{\mathbb{Q}_n}[T_{\theta_j}] - \mathbb{E}_{\mathbb{Q}_n}[T_\theta]| \\ &\leq \tfrac{\epsilon}{8}e^{-2M} + \tfrac{\epsilon}{4} + \tfrac{\epsilon}{8}e^{-2M} \\ &\leq \tfrac{\epsilon}{2} \end{aligned} \tag{31}$$

Similarly, using Eqn 20 and 29, we obtain that with probability at least $1-\delta$,

$$|\log \mathbb{E}_{\mathbb{Q}}[e^{T_\theta}] - \log \mathbb{E}_{\mathbb{Q}_n}[e^{T_\theta}]| \leq \frac{\epsilon}{2} \tag{32}$$

and hence using the triangular inequality,

$$|\widehat{I(X;Z)}_n - I_{\mathcal{F}}(X,Z)| \leq \epsilon \tag{33}$$

□

C.2.4. Bound on the reconstruction error

Here we clarify relationship between reconstruction error and mutual information, by proving the bound in Equ 18. We begin with a definition:

Definition C.1 (Reconstruction Error)**.** *We consider encoder and decoder models giving conditional distributions $q(z|x)$ and $p(x|z)$ over the data and latent variables. If $q(x)$ denotes the marginal data distribution, the reconstruction error is defined as*

$$\mathcal{R} = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[-\log p(\boldsymbol{x}|\boldsymbol{z})] \tag{34}$$

We can rewrite the reconstruction error in terms of the joints $q(\boldsymbol{x},\boldsymbol{z}) = q(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})$ and $p(\boldsymbol{x},\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$. Elementary manipulations give:

$$\mathcal{R} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{z})\sim q(\boldsymbol{x},\boldsymbol{z})} \log \frac{q(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{x},\boldsymbol{z})} - \mathbb{E}_{(\boldsymbol{x},\boldsymbol{z})\sim q(\boldsymbol{x},\boldsymbol{z})} \log q(\boldsymbol{x},\boldsymbol{z}) + \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})} \log p(\boldsymbol{z}) \tag{35}$$

where $q(\boldsymbol{z})$ is the aggregated posterior. The first term is the KL-divergence $D_{KL}(q \,||\, p)$; the second term is the joint entropy $H_q(\boldsymbol{x},\boldsymbol{z})$. The third term can be written as

$$\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})} \log p(\boldsymbol{z}) = -D_{KL}(q(\boldsymbol{z}) \,||\, p(\boldsymbol{z})) - H_q(\boldsymbol{z})$$

Finally, the identity

$$H_q(\boldsymbol{x},\boldsymbol{z}) - H_q(\boldsymbol{z}) := H_q(\boldsymbol{z}|\boldsymbol{x}) = H_q(\boldsymbol{z}) - I_q(\boldsymbol{x},\boldsymbol{z}) \tag{36}$$

yields the following expression for the reconstruction error:

$$\mathcal{R} = D_{KL}(q(\boldsymbol{x},\boldsymbol{z}) \,||\, p(\boldsymbol{x},\boldsymbol{z})) - D_{KL}(q(\boldsymbol{z}) \,||\, p(\boldsymbol{z})) - I_q(\boldsymbol{x},\boldsymbol{z}) + H_q(\boldsymbol{z}) \tag{37}$$

Since the KL-divergence is positive, we obtain the bound:

$$\mathcal{R} \leq D_{KL}(q(\boldsymbol{x},\boldsymbol{z}) \,||\, p(\boldsymbol{x},\boldsymbol{z})) - I_q(\boldsymbol{x},\boldsymbol{z}) + H_q(\boldsymbol{z}) \tag{38}$$

which is tight whenever the induced marginal $q(\boldsymbol{z})$ matches the prior distribution $p(\boldsymbol{z})$.

C.2.5. Embeddings for bi-direction 25 Gaussians experiments

Here (Fig. C.1) we present the embeddings for the experiments corresponding to Fig. 5.6.

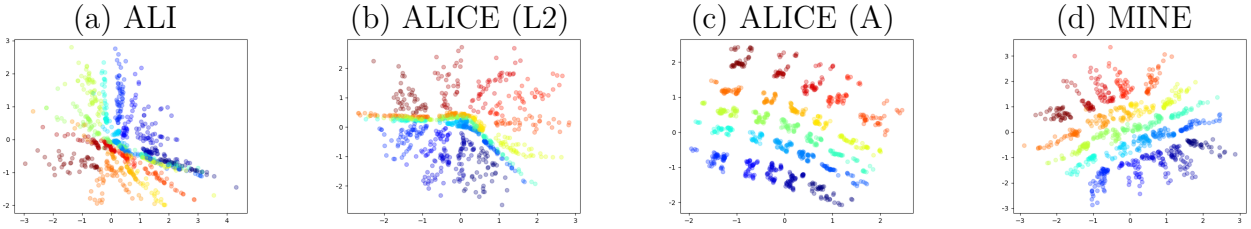


Fig. C.1. Embeddings from adversarially learned inference (ALI) and variations intended to increase the mutual information. Shown left to right are the baseline (ALI), ALICE with the L2 loss to minimize the reconstruction error, ALI with an additional adversarial loss, and MINE.