

The Modern Mathematics of Deep Learning^{*}

Julius Berner[†] Philipp Grohs[‡] Gitta Kutyniok[§] Philipp Petersen[‡]

Abstract

We describe the new field of mathematical analysis of deep learning. This field emerged around a list of research questions that were not answered within the classical framework of learning theory. These questions concern: the outstanding generalization power of overparametrized neural networks, the role of depth in deep architectures, the apparent absence of the curse of dimensionality, the surprisingly successful optimization performance despite the non-convexity of the problem, understanding what features are learned, why deep architectures perform exceptionally well in physical problems, and which fine aspects of an architecture affect the behavior of a learning task in which way. We present an overview of modern approaches that yield partial answers to these questions. For selected approaches, we describe the main ideas in more detail.

Contents

1	Introduction	2
1.1	Notation	4
1.2	Foundations of learning theory	4
1.3	Do we need a new theory?	17
2	Generalization of large neural networks	22
2.1	Kernel regime	23
2.2	Norm-based bounds and margin theory	24
2.3	Optimization and implicit regularization	25
2.4	Limits of classical theory and double descent	27
3	The role of depth in the expressivity of neural networks	29
3.1	Approximation of radial functions	29
3.2	Deep ReLU networks	31
3.3	Alternative notions of expressivity	32
4	Deep neural networks overcome the curse of dimensionality	34
4.1	Manifold assumption	34
4.2	Random sampling	35
4.3	PDE assumption	36
5	Optimization of deep neural networks	39
5.1	Loss landscape analysis	39
5.2	Lazy training and provable convergence of stochastic gradient descent	41

^{*}This review paper will appear as a book chapter in the book “Theory of Deep Learning” by Cambridge University Press.

[†]Faculty of Mathematics, University of Vienna.

[‡]Faculty of Mathematics and Research Network DataScience@UniVienna, University of Vienna.

[§]Department of Mathematics, Ludwig Maximilian University of Munich, and Department of Physics and Technology, University of Tromsø.

6	Tangible effects of special architectures	44
6.1	Convolutional neural networks	45
6.2	Residual neural networks	46
6.3	Framelets and U-Nets	47
6.4	Batch normalization	49
6.5	Sparse neural networks and pruning	50
6.6	Recurrent neural networks	52
7	Describing the features a deep neural network learns	52
7.1	Invariances and the scattering transform	52
7.2	Hierarchical sparse representations	53
8	Effectiveness in natural sciences	55
8.1	Deep neural networks meet inverse problems	55
8.2	PDE-based models	56

1 Introduction

Deep learning has undoubtedly established itself as the outstanding machine learning technique of recent times. This dominant position was claimed through a series of overwhelming successes in widely different application areas.

Perhaps the most famous application of deep learning and certainly one of the first where these techniques became state-of-the-art is image classification [LBBH98, KSH12, SLJ⁺15, HZRS16]. In this area, deep learning is nowadays the only method that is seriously considered. The prowess of deep learning classifiers goes so far that they often outperform humans in image labelling tasks [HZRS15].

A second famous application area is the training of deep-learning-based agents to play board games or computer games, such as Atari games [MKS⁺13]. In this context, probably the most prominent achievement yet is the development of an algorithm that beat the best human player in the game of Go [SHM⁺16, SSS⁺17]—a feat that was previously unthinkable owing to the extreme complexity of this game. Besides, even in multiplayer, team-based games with incomplete information deep-learning-based agents nowadays outperform world-class human teams [BBC⁺19, VBC⁺19].

In addition to playing games, deep learning has also led to impressive breakthroughs in the natural sciences. For example, it is used in the development of drugs [MSL⁺15], molecular dynamics [FHH⁺17], or in high-energy physics [BSW14]. One of the most astounding recent breakthroughs in scientific applications is the development of a deep-learning-based predictor for the folding behavior of proteins [SEJ⁺20]. This predictor is the first method to match the accuracy of lab-based methods.

Finally, in the vast field of natural language processing, which includes the subtasks of understanding, summarizing, or generating text, impressive advances were made based on deep learning. Here, we refer to [YHPC18] for an overview. One technique that recently stood out is based on a so-called transformer neural network [BCB15, VSP⁺17]. This network structure gave rise to the impressive GPT-3 model [BMR⁺20] which not only creates coherent and compelling texts but can also produce code, such as, for the layout of a webpage according to some instructions that a user inputs in plain English. Transformer neural networks have also been successfully employed in the field of symbolic mathematics [SGHK18, LC19].

In this article, we present and discuss the mathematical foundations of the success story outlined above. More precisely, our goal is to outline the newly emerging field of *mathematical analysis of deep learning*. To accurately describe this field, a necessary preparatory step is to sharpen our definition of the term deep learning. For the purposes of this article, we will use the term in the following narrow sense: *Deep learning refers to techniques where deep neural networks¹ are trained with gradient-based methods*. This narrow

¹We will define the term *neural network* later but, for this definition, one can view it as a parametrized family of functions with a differentiable parametrization.

definition is helpful to make this article more concise. We would like to stress, however, that we do not claim in any way that this is the *best* or the *right* definition of deep learning.

Having fixed a definition of deep learning, three questions arise concerning the aforementioned emerging field of mathematical analysis of deep learning: To what extent is a mathematical theory necessary? Is it truly a new field? What are the questions studied in this area?

Let us start by explaining the necessity of a theoretical analysis of the tools described above. From a scientific perspective, the primary reason why deep learning should be studied mathematically is simple curiosity. As we will see throughout this article, many practically observed phenomena in this context are not explained theoretically. Moreover, theoretical insights and the development of a comprehensive theory are often the driving force underlying the development of new and improved methods. Prominent examples of mathematical theories with such an effect are the theory of fluid mechanics which is an invaluable asset to the design of aircraft or cars and the theory of information that affects and shapes all modern digital communication. In the words of Vapnik²: “Nothing is more practical than a good theory”, [Vap13, Preface]. In addition to being interesting and practical, theoretical insight may also be necessary. Indeed, in many applications of machine learning, such as medical diagnosis, self-driving cars, and robotics, a significant level of control and predictability of deep learning methods is mandatory. Also, in services, such as banking or insurance, the technology should be controllable to guarantee fair and explainable decisions.

Let us next address the claim that the field of mathematical analysis of deep learning is a newly emerging area. In fact, under the aforementioned definition of deep learning, there are two main ingredients of the technology: deep neural networks and gradient-based optimization. The first artificial neuron was already introduced in 1943 in [MP43]. This neuron was not trained but instead used to explain a biological neuron. The first multi-layered network of such artificial neurons that was also trained can be found in [Ros58]. Since then, various neural network architectures have been developed. We will discuss these architectures in detail in the following sections. The second ingredient, gradient-based optimization, is made possible by the observation that due to the graph-based structure of neural networks the gradient of an objective function with respect to the parameters of the neural network can be computed efficiently. This has been observed in various ways, see [Kel60, Dre62, Lin70, RHW86]. Again, these techniques will be discussed in the upcoming sections. Since then, techniques have been improved and extended. As the rest of the manuscript is spent reviewing these methods, we will keep the discussion of literature at this point brief. Instead, we refer to some overviews of the history of deep learning from various perspectives: [LBH15, Sch15, GBC16, HH19].

Given the fact that the two main ingredients of deep neural networks have been around for a long time, one would expect that a comprehensive mathematical theory has been developed that describes why and when deep-learning-based methods will perform well or when they will fail. Statistical learning theory [AB99, Vap99, CS02, BBL03, Vap13] describes multiple aspects of the performance of general learning methods and in particular deep learning. We will review this theory in the context of deep learning in Subsection 1.2 below. Hereby, we focus on classical, deep learning-related results that we consider well-known in the machine learning community. Nonetheless, the choice of these results is guaranteed to be subjective. We will find that the presented, classical theory is too general to explain the performance of deep learning adequately. In this context, we will identify the following questions that appear to be difficult to answer within the classical framework of learning theory: *Why do trained deep neural networks not overfit on the training data despite the enormous power of the architecture? What is the advantage of deep compared to shallow architectures? Why do these methods seemingly not suffer from the curse of dimensionality? Why does the optimization routine often succeed in finding good solutions despite the non-convexity, non-linearity, and often non-smoothness of the problem? Which aspects of an architecture affect the performance of the associated models and how? Which features of data are learned by deep architectures? Why do these methods perform as well as or better than specialized numerical tools in natural sciences?*

The new field of mathematical analysis of deep learning has emerged around questions like the ones listed above. In the remainder of this article, we will collect some of the main recent advances to answer these questions. Because this field of mathematical analysis of deep learning is incredibly active and new material is added at breathtaking speeds, a brief survey on recent advances in this area is guaranteed to miss not only

²This claim can be found earlier in a non-mathematical context in the works of Kurt Lewin [Lew43].

a couple of references but also many of the most essential ones. Therefore, we do not strive for a complete overview, but instead, showcase several fundamental ideas on a mostly intuitive level. In this way, we hope to allow the reader to familiarize themselves with some exciting concepts and provide a convenient entry-point for further studies.

1.1 Notation

We denote by \mathbb{N} the set of natural numbers, by \mathbb{Z} the set of integers and by \mathbb{R} the field of real numbers. For $N \in \mathbb{N}$, we denote by $[N]$ the set $\{1, \dots, N\}$. For two functions $f, g: \mathcal{X} \rightarrow [0, \infty)$, we write $f \lesssim g$, if there exists a universal constant c such that $f(x) \leq cg(x)$ for all $x \in \mathcal{X}$. In a pseudometric space $(\mathcal{X}, d_{\mathcal{X}})$, we define the ball of radius $r \in (0, \infty)$ around a point $x \in \mathcal{X}$ by $B_r^{d_{\mathcal{X}}}(x)$ or $B_r(x)$ if the pseudometric $d_{\mathcal{X}}$ is clear from the context. By $\|\cdot\|_p$, $p \in [1, \infty]$, we denote the ℓ^p -norm, and by $\langle \cdot, \cdot \rangle$ the Euclidean inner product of given vectors. By $\|\cdot\|_{\text{op}}$ we denote the operator norm induced by the Euclidean norm and by $\|\cdot\|_F$ the Frobenius norm of given matrices. For $p \in [1, \infty]$, $s \in [0, \infty)$, $d \in \mathbb{N}$, and $\mathcal{X} \subset \mathbb{R}^d$, we denote by $W^{s,p}(\mathcal{X})$ the Sobolev-Slobodeckij space, which for $s = 0$ is just a Lebesgue space, i.e., $W^{0,p}(\mathcal{X}) = L^p(\mathcal{X})$. For measurable spaces \mathcal{X} and \mathcal{Y} , we define $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ to be the set of measurable functions from \mathcal{X} to \mathcal{Y} . We denote by \hat{g} the Fourier transform³ of a tempered distribution g . For probabilistic statements, we will assume a suitable underlying probability space with probability measure \mathbb{P} . For an \mathcal{X} -valued random variable X , we denote by $\mathbb{E}[X]$ and $\mathbb{V}[X]$ its expectation and variance and by \mathbb{P}_X the image measure of X on \mathcal{X} , i.e., $\mathbb{P}_X(A) = \mathbb{P}(X \in A)$ for every measurable set $A \subset \mathcal{X}$. If possible, we use the corresponding lowercase letter to denote the realization $x \in \mathcal{X}$ of the random variable X for a given outcome. We write \mathbf{I}_d for the d -dimensional identity matrix and, for a set A , we write $\mathbf{1}_A$ for the indicator function of A , i.e., $\mathbf{1}_A(x) = 1$ if $x \in A$ and $\mathbf{1}_A(x) = 0$ else.

1.2 Foundations of learning theory

Before we continue to describe recent developments in the mathematical analysis of deep learning methods, we start by providing a concise overview of the classical mathematical and statistical theory underlying machine learning tasks and algorithms which, in their most general form, can be formulated as follows.

Definition 1.1 (Learning - informal). *Let \mathcal{X}, \mathcal{Y} , and \mathcal{Z} be measurable spaces. In a learning task, one is given data in \mathcal{Z} and a loss function $\mathcal{L}: \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{Z} \rightarrow \mathbb{R}$. The goal is to choose a hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$ and construct a learning algorithm, i.e., a mapping*

$$\mathcal{A}: \bigcup_{m \in \mathbb{N}} \mathcal{Z}^m \rightarrow \mathcal{F},$$

that uses training data $s = (z^{(i)})_{i=1}^m \in \mathcal{Z}^m$ to find a model $f_s = \mathcal{A}(s) \in \mathcal{F}$ that performs well on the training data s and also generalizes to unseen data $z \in \mathcal{Z}$. Here, performance is measured via the loss function \mathcal{L} and the corresponding loss $\mathcal{L}(f_s, z)$ and, informally speaking, generalization means that the out-of-sample performance of f_s at z behaves similar to the in-sample performance on s .

Definition 1.1 is deliberately vague on how to measure generalization performance. Later, we will often study the *expected* out-of-sample performance. To talk about expected performance, a data distribution needs to be specified. We will revisit this point in Assumption 1.10 and Definition 1.11.

For simplicity, we focus on one-dimensional, supervised prediction tasks with input features in Euclidean space as defined in the following.

Definition 1.2 (Prediction task). *In a prediction task, we have that $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$, i.e., we are given training data $s = ((x^{(i)}, y^{(i)}))_{i=1}^m$ that consist of input features $x^{(i)} \in \mathcal{X}$ and corresponding labels $y^{(i)} \in \mathcal{Y}$. For one-dimensional regression tasks with $\mathcal{Y} \subset \mathbb{R}$, we consider the quadratic loss $\mathcal{L}(f, (x, y)) = (f(x) - y)^2$ and,*

³Respecting common notation, we will also use the hat symbol to denote the minimizer of the empirical risk \hat{f}_s in Definition 1.8 but this clash of notation does not cause any ambiguity.

for binary classification tasks with $\mathcal{Y} = \{-1, 1\}$, we consider the 0-1 loss $\mathcal{L}(f, (x, y)) = \mathbb{1}_{(-\infty, 0)}(yf(x))$. We assume that our input features are in Euclidean space, i.e., $\mathcal{X} \subset \mathbb{R}^d$ with input dimension $d \in \mathbb{N}$.

In a prediction task, we aim for a model $f_s: \mathcal{X} \rightarrow \mathcal{Y}$, such that, for unseen pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $f_s(x)$ is a good prediction of the true label y . However, note that large parts of the presented theory can be applied to more general settings.

Remark 1.3 (Learning tasks). *Apart from straightforward extensions to multi-dimensional prediction tasks and other loss functions, we want to mention that unsupervised and semi-supervised learning tasks are often treated as prediction tasks. More precisely, one transforms unlabeled training data $z^{(i)}$ into features $x^{(i)} = T_1(z^{(i)}) \in \mathcal{X}$ and labels $y^{(i)} = T_2(z^{(i)}) \in \mathcal{Y}$ using suitable transformations $T_1: \mathcal{Z} \rightarrow \mathcal{X}$, $T_2: \mathcal{Z} \rightarrow \mathcal{Y}$. In doing so, one asks for a model f_s approximating the transformation $T_2 \circ T_1^{-1}: \mathcal{X} \rightarrow \mathcal{Y}$ which is, e.g., done in order to learn feature representations or invariances.*

Furthermore, one can consider density estimation tasks, where $\mathcal{X} = \mathcal{Z}$, $\mathcal{Y} := [0, \infty]$, and \mathcal{F} consists of probability densities with respect to some σ -finite reference measure μ on \mathcal{Z} . One then aims for a probability density f_s that approximates the density of the unseen data z with respect to μ . One can perform $L^2(\mu)$ -approximation based on the discretization $\mathcal{L}(f, z) = -2f(z) + \|f\|_{L^2(\mu)}^2$ or maximum likelihood estimation based on the surprisal $\mathcal{L}(f, z) = -\log(f(z))$.

In deep learning the hypothesis set \mathcal{F} consists of realizations of neural networks $\Phi_a(\cdot, \theta)$, $\theta \in \mathcal{P}$, with a given architecture a and parameter set \mathcal{P} . In practice, one uses the term neural network for a range of functions that can be represented by directed acyclic graphs, where the vertices correspond to elementary almost everywhere differentiable functions parametrizable by $\theta \in \mathcal{P}$ and the edges symbolize compositions of these functions. In Section 6, we will review some frequently used architectures, in the other sections, however, we will mostly focus on *fully connected feedforward* (FC) neural networks as defined below.

Definition 1.4 (FC neural network). *A fully connected feedforward neural network is given by its architecture $a = (N, \varrho)$, where $L \in \mathbb{N}$, $N \in \mathbb{N}^{L+1}$, and $\varrho: \mathbb{R} \rightarrow \mathbb{R}$. We refer to ϱ as the activation function, to L as the number of layers, and to N_0 , N_L , and N_ℓ , $\ell \in [L-1]$, as the number of neurons in the input, output, and ℓ -th hidden layer, respectively. We denote the number of parameters by*

$$P(N) := \sum_{\ell=1}^L N_\ell N_{\ell-1} + N_\ell$$

and define the corresponding realization function $\Phi_a: \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \rightarrow \mathbb{R}^{N_L}$ which satisfies for every input $x \in \mathbb{R}^{N_0}$ and parameters

$$\theta = (\theta^{(\ell)})_{\ell=1}^L = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^L \in \prod_{\ell=1}^L (\mathbb{R}^{N_\ell \times N_{\ell-1}} \times \mathbb{R}^{N_\ell}) \cong \mathbb{R}^{P(N)}$$

that $\Phi_a(x, \theta) = \Phi^{(L)}(x, \theta)$, where

$$\begin{aligned} \Phi^{(1)}(x, \theta) &= W^{(1)}x + b^{(1)}, \\ \bar{\Phi}^{(\ell)}(x, \theta) &= \varrho(\Phi^{(\ell)}(x, \theta)), \quad \ell \in [L-1], \quad \text{and} \\ \Phi^{(\ell+1)}(x, \theta) &= W^{(\ell+1)}\bar{\Phi}^{(\ell)}(x, \theta) + b^{(\ell+1)}, \quad \ell \in [L-1], \end{aligned} \tag{1.1}$$

and ϱ is applied componentwise. We refer to $W^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $b^{(\ell)} \in \mathbb{R}^{N_\ell}$ as the weight matrices and bias vectors, and to $\bar{\Phi}^{(\ell)}$ and $\Phi^{(\ell)}$ as the activations and pre-activations of the N_ℓ neurons in the ℓ -th layer. The width and depth of the architecture are given by $\|N\|_\infty$ and L and we call the architecture deep if $L > 2$ and shallow if $L = 2$.

The underlying directed acyclic graph of FC networks is given by compositions of the affine linear maps $x \mapsto W^{(\ell)}x + b^{(\ell)}$, $\ell \in [L]$, with the activation function ϱ intertwined, see Figure 1.1. Typical activation

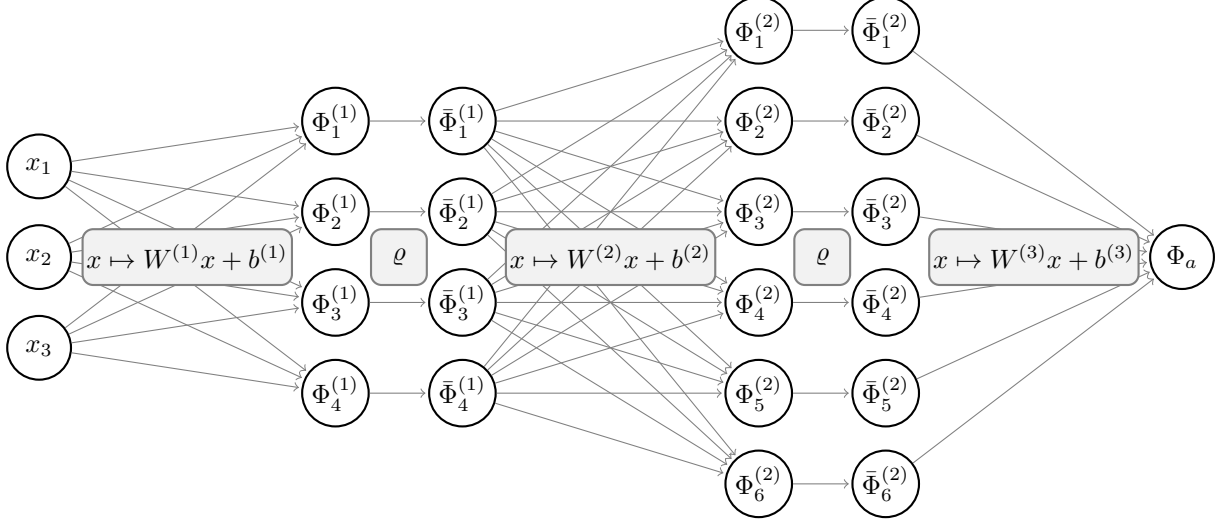


Figure 1.1: Graph (grey) and (pre-)activations of the neurons (white) of a deep fully connected feedforward neural network $\Phi_a: \mathbb{R}^3 \times \mathbb{R}^{53} \mapsto \mathbb{R}$ with architecture $a = ((3, 4, 6, 1), \varrho)$ and parameters $\theta = ((W^{(\ell)}, b^{(\ell)})_{\ell=1}^3$.

functions used in practice are variants of the *rectified linear unit* (ReLU) given by $\varrho_R(x) := \max\{0, x\}$ and *sigmoidal functions* $\varrho \in C(\mathbb{R})$ satisfying $\varrho(x) \rightarrow 1$ for $x \rightarrow \infty$ and $\varrho(x) \rightarrow 0$ for $x \rightarrow -\infty$, such as the logistic function $\varrho_\sigma(x) := 1/(1 + e^{-x})$ (often referred to as *the sigmoid function*). See also Table 1 for a comprehensive list of widely used activation functions.

Remark 1.5 (Neural networks). *If not further specified, we will use the term (neural) network, or the abbreviation NN, to refer to FC neural networks. Note that many of the architectures used in practice (see Section 6) can be written as special cases of Definition 1.4 where, e.g., specific parameters are prescribed by constants or shared with other parameters. Furthermore, note that affine linear functions are NNs with depth $L = 1$. We will also consider biasless NNs given by linear mappings without bias vector, i.e., $b^{(\ell)} = 0$, $\ell \in [L]$. In particular, any NN can always be written without bias vectors by redefining*

$$x \mapsto \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad (W^{(\ell)}, b^{(\ell)}) \mapsto \begin{bmatrix} W^{(\ell)} & b^{(\ell)} \\ 0 & 1 \end{bmatrix}, \quad \ell \in [L-1], \quad \text{and} \quad (W^{(L)}, b^{(L)}) \mapsto [W^{(L)} \quad b^{(L)}].$$

To enhance readability we will often not specify the underlying architecture $a = (N, \varrho)$ or the parameters $\theta \in \mathbb{R}^{P(N)}$ and use the term NN to refer to the architecture as well as the realization functions $\Phi_a(\cdot, \theta): \mathbb{R}^{N_0} \rightarrow \mathbb{R}^{N_L}$ or $\Phi_a: \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \rightarrow \mathbb{R}^{N_L}$. However, we want to emphasize that one cannot infer the underlying architecture or properties like magnitude of parameters solely from these functions as the mapping $(a, \theta) \mapsto \Phi_a(\cdot, \theta)$ is highly non-injective. As an example, we can set $W^{(L)} = 0$ which implies $\Phi_a(\cdot, \theta) = b^{(L)}$ for all architectures $a = (N, \varrho)$ and all values of $(W^{(\ell)}, b^{(\ell)})_{\ell=1}^{L-1}$.

In view of our considered prediction tasks in Definition 1.2, this naturally leads to the following hypothesis sets of neural networks.

Definition 1.6 (Hypothesis sets of neural networks). *Let $a = (N, \varrho)$ be a NN architecture with input dimension $N_0 = d$, output dimension $N_L = 1$, and measurable activation function ϱ . For regression tasks the corresponding hypothesis set is given by*

$$\mathcal{F}_a = \{\Phi_a(\cdot, \theta): \theta \in \mathbb{R}^{P(N)}\}$$

and for classification tasks by

$$\mathcal{F}_{a, \text{sgn}} = \{\text{sgn}(\Phi_a(\cdot, \theta)): \theta \in \mathbb{R}^{P(N)}\}, \quad \text{where} \quad \text{sgn}(x) := \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0. \end{cases}$$

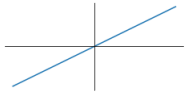

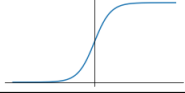
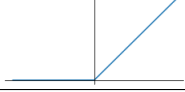
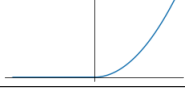
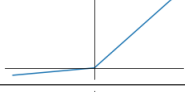
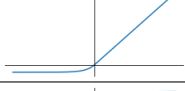
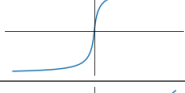
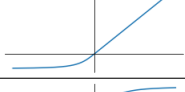
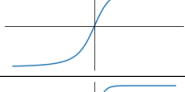
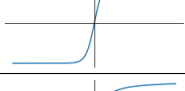
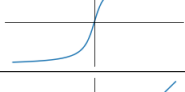
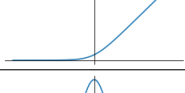
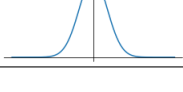
Name	Given as a function of $x \in \mathbb{R}$ by	Plot
linear	x	
Heaviside / step function	$\mathbb{1}_{(0,\infty)}(x)$	
logistic / sigmoid	$\frac{1}{1+e^{-x}}$	
rectified linear unit (ReLU)	$\max\{0, x\}$	
power rectified linear unit	$\max\{0, x\}^k$ for $k \in \mathbb{N}$	
parametric ReLU (PReLU)	$\max\{ax, x\}$ for $a \geq 0, a \neq 1$	
exponential linear unit (ELU)	$x \cdot \mathbb{1}_{[0,\infty)}(x) + (e^x - 1) \cdot \mathbb{1}_{(-\infty,0)}(x)$	
softsign	$\frac{x}{1+ x }$	
inverse square root linear unit	$x \cdot \mathbb{1}_{[0,\infty)}(x) + \frac{x}{\sqrt{1+ax^2}} \cdot \mathbb{1}_{(-\infty,0)}(x)$ for $a > 0$	
inverse square root unit	$\frac{x}{\sqrt{1+ax^2}}$ for $a > 0$	
tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	
arctan	$\arctan(x)$	
softplus	$\ln(1 + e^x)$	
Gaussian	$e^{-x^2/2}$	

Table 1: List of commonly used activation functions.

Note that we compose the output of the NN with the sign function in order to obtain functions mapping to $\mathcal{Y} = \{-1, 1\}$. This can be generalized to multi-dimensional classification tasks by replacing the sign by an argmax function. Given a hypothesis set, a popular learning algorithm is *empirical risk minimization* (ERM), which minimizes the average loss on the given training data, as described in the next definitions.

Definition 1.7 (Empirical risk). *For training data $s = (z^{(i)})_{i=1}^m \in \mathcal{Z}^m$ and a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, we define the empirical risk by*

$$\widehat{\mathcal{R}}_s(f) := \frac{1}{m} \sum_{i=1}^m \mathcal{L}(f, z^{(i)}).$$

Definition 1.8 (ERM learning algorithm). *Given a hypothesis set \mathcal{F} , an empirical risk minimization algorithm \mathcal{A}^{erm} chooses⁴ for training data $s \in \mathcal{Z}^m$ a minimizer $\widehat{f}_s \in \mathcal{F}$ of the empirical risk in \mathcal{F} , i.e.,*

$$\mathcal{A}^{\text{erm}}(s) \in \arg \min_{f \in \mathcal{F}} \widehat{\mathcal{R}}_s(f). \quad (1.2)$$

Remark 1.9 (Surrogate loss and regularization). *Note that, for classification tasks, one needs to optimize over non-differentiable functions with discrete outputs in (1.2). For NN hypothesis sets $\mathcal{F}_{a, \text{sgn}}$ one typically uses the corresponding hypothesis set for regression tasks \mathcal{F}_a to find an approximate minimizer $\widehat{f}_s^{\text{sur}} \in \mathcal{F}_a$ of*

$$\frac{1}{m} \sum_{i=1}^m \mathcal{L}^{\text{sur}}(f, z^{(i)}),$$

where $\mathcal{L}^{\text{sur}}: \mathcal{M}(\mathcal{X}, \mathbb{R}) \times \mathcal{Z} \rightarrow \mathbb{R}$ is a surrogate loss guaranteeing that $\text{sgn}(\widehat{f}_s^{\text{sur}}) \in \arg \min_{f \in \mathcal{F}_{a, \text{sgn}}} \widehat{\mathcal{R}}_s(f)$. A frequently used surrogate loss is the logistic loss⁵ given by

$$\mathcal{L}^{\text{sur}}(f, z) = \log \left(1 + e^{-yf(x)} \right).$$

In various learning tasks one also adds regularization terms to the minimization problem in (1.2), such as penalties on the norm of the parameters of the NN, i.e.,

$$\min_{\theta \in \mathbb{R}^{P(N)}} \widehat{\mathcal{R}}_s(\Phi_a(\cdot, \theta)) + \alpha \|\theta\|_2^2,$$

where $\alpha \in (0, \infty)$ is a regularization parameter. Note that in this case the minimizer depends on the chosen parameters θ and not only on the realization function $\Phi_a(\cdot, \theta)$, see also Remark 1.5.

Coming back to our initial, informal description of learning in Definition 1.1, we have now outlined potential learning tasks in Definition 1.2, NN hypothesis sets in Definition 1.6, a metric for the in-sample performance in Definition 1.7, and a corresponding learning algorithm in Definition 1.8. However, we are still lacking a mathematical concept to describe the out-of-sample (generalization) performance of our learning algorithm. This question has been intensively studied in the field of statistical learning theory, see Section 1 for various references.

In this field one usually establishes a connection between unseen data z and the training data $s = (z^{(i)})_{i=1}^m$ by imposing that z and $z^{(i)}$, $i \in [m]$, are realizations of independent samples drawn from the same distribution.

Assumption 1.10 (Independent and identically distributed data). *We assume that $z^{(1)}, \dots, z^{(m)}, z$ are realizations of i.i.d. random variables $Z^{(1)}, \dots, Z^{(m)}, Z$.*

⁴For simplicity, we assume that the minimum is attained which, for instance, is the case if \mathcal{F} is a compact topological space on which $\widehat{\mathcal{R}}_s$ is continuous. Hypothesis sets of NNs $\mathcal{F}_{(N, \varrho)}$ constitute a compact space if, e.g., one chooses a compact parameter set $\mathcal{P} \subset \mathbb{R}^{P(N)}$ and a continuous activation function ϱ . One could also work with approximate minimizers, see [AB99].

⁵This can be viewed as cross-entropy between the label y and the output of f composed with a logistic function ϱ_σ . In a multi-dimensional setting one can replace the logistic function with a softmax function.

In this formal setting, we can compute the average out-of-sample performance of a model. Recall from our notation in Section 1.1 that we denote by \mathbb{P}_Z the image measure of Z on \mathcal{Z} , which is the underlying distribution of our training data $S = (Z^{(i)})_{i=1}^m \sim \mathbb{P}_Z^m$ and unknown data $Z \sim \mathbb{P}_Z$.

Definition 1.11 (Risk). *For a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, we define⁶ the risk by*

$$\mathcal{R}(f) := \mathbb{E}[\mathcal{L}(f, Z)] = \int_{\mathcal{Z}} \mathcal{L}(f, z) d\mathbb{P}_Z(z).$$

Defining $S := (Z^{(i)})_{i=1}^m$, the risk of a model $f_S = \mathcal{A}(S)$ is thus given by $\mathcal{R}(f_S) = \mathbb{E}[\mathcal{L}(f_S, Z)|S]$.

For prediction tasks, we can write $Z = (X, Y)$, such that the input features and labels are given by an \mathcal{X} -valued random variable X and a \mathcal{Y} -valued random variable Y , respectively. Note that for classification tasks the risk equals the probability of misclassification

$$\mathcal{R}(f) = \mathbb{E}[\mathbb{1}_{(-\infty, 0)}(Yf(X))] = \mathbb{P}[f(X) \neq Y].$$

For noisy data, there might be a positive, lower bound on the risk, i.e., an irreducible error. If the lower bound on the risk is attained, one can also define the notion of an optimal solution to a learning task.

Definition 1.12 (Bayes-optimal function). *A function $f^* \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ achieving the smallest risk, the so-called Bayes risk*

$$\mathcal{R}^* := \inf_{f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})} \mathcal{R}(f),$$

is called a Bayes-optimal function.

For the prediction tasks in Definition 1.2, we can represent the risk of a function with respect to the Bayes risk and compute the Bayes-optimal function, see, e.g., [CZ07, Propositions 1.8 and 9.3].

Lemma 1.1 (Regression and classification risk). *For a regression task with $\mathbb{V}[Y] < \infty$, the risk can be decomposed into*

$$\mathcal{R}(f) = \mathbb{E}[(f(X) - \mathbb{E}[Y|X])^2] + \mathcal{R}^*, \quad f \in \mathcal{M}(\mathcal{X}, \mathcal{Y}), \quad (1.3)$$

which is minimized by the regression function $f^(x) = \mathbb{E}[Y|X = x]$. For a classification task, the risk can be decomposed into*

$$\mathcal{R}(f) = \mathbb{E}[|\mathbb{E}[Y|X]| \mathbb{1}_{(-\infty, 0)}(\mathbb{E}[Y|X]f(X))] + \mathcal{R}^*, \quad f \in \mathcal{M}(\mathcal{X}, \mathcal{Y}),$$

which is minimized by the Bayes classifier $f^(x) = \text{sgn}(\mathbb{E}[Y|X = x])$.*

As our model f_S is depending on the random training data S , the risk $\mathcal{R}(f_S)$ is a random variable and we might aim⁷ for $\mathcal{R}(f_S)$ small with high probability or in expectation over the training data. The challenge for the learning algorithm \mathcal{A} is to minimize the risk by only using training data but without knowing the underlying distribution. One can even show that for every learning algorithm there exists a distribution where convergence of the expected risk of f_S to the Bayes risk is arbitrarily slow with respect to the number of samples m [DGL96, Theorem 7.2].

Theorem 1.13 (No free lunch). *Let $a_m \in (0, \infty)$, $m \in \mathbb{N}$, be a monotonically decreasing sequence with $a_1 \leq 1/16$. Then for every learning algorithm \mathcal{A} of a classification task there exists a distribution \mathbb{P}_Z such that for every $m \in \mathbb{N}$ and training data $S \sim \mathbb{P}_Z^m$ it holds that*

$$\mathbb{E}[\mathcal{R}(\mathcal{A}(S))] \geq \mathcal{R}^* + a_m.$$

⁶Note that this requires $z \mapsto \mathcal{L}(f, z)$ to be measurable for every $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, which is the case for our considered prediction tasks.

⁷In order to make probabilistic statements on $\mathcal{R}(f_S)$ we assume that $\mathcal{R}(f_S)$ is a random variable, i.e., measurable. This is, e.g., the case if \mathcal{F} constitutes a measurable space and $s \mapsto \mathcal{A}(s)$ and $f \mapsto \mathcal{R}|_{\mathcal{F}}$ are measurable.

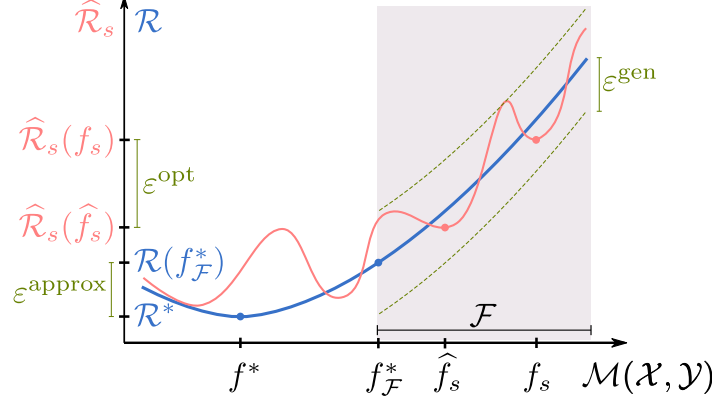


Figure 1.2: Illustration of the errors (A)–(C) in the decomposition of (1.4). It shows an exemplary risk $\widehat{\mathcal{R}}$ (blue) and empirical risk $\widehat{\mathcal{R}}_s$ (red) with respect to the projected space of measurable functions $\mathcal{M}(\mathcal{X}, \mathcal{Y})$. Note that the empirical risk and thus ε^{gen} and ε^{opt} depend on the realization $s = (z^{(i)})_{i=1}^m$ of the training data $S \sim \mathbb{P}_Z^m$.

Theorem 1.13 shows the non-existence of a universal learning algorithm for every data distribution \mathbb{P}_Z and shows that useful bounds must necessarily be accompanied by a priori regularity conditions on the underlying distribution \mathbb{P}_Z . Such prior knowledge can then be incorporated in the choice of the hypothesis set \mathcal{F} . To illustrate this, let $f_{\mathcal{F}}^* \in \arg \min_{f \in \mathcal{F}} \mathcal{R}(f)$ be a best approximation in \mathcal{F} , such that we can bound the error

$$\begin{aligned} \mathcal{R}(f_s) - \mathcal{R}^* &= \mathcal{R}(f_s) - \widehat{\mathcal{R}}_s(f_s) + \widehat{\mathcal{R}}_s(f_s) - \widehat{\mathcal{R}}_s(f_{\mathcal{F}}^*) + \widehat{\mathcal{R}}_s(f_{\mathcal{F}}^*) - \mathcal{R}(f_{\mathcal{F}}^*) + \mathcal{R}(f_{\mathcal{F}}^*) - \mathcal{R}^* \\ &\leq \varepsilon^{\text{opt}} + 2\varepsilon^{\text{gen}} + \varepsilon^{\text{approx}} \end{aligned} \quad (1.4)$$

by

- (A) an *optimization error* $\varepsilon^{\text{opt}} := \widehat{\mathcal{R}}_s(f_s) - \widehat{\mathcal{R}}_s(\widehat{f}_s) \geq \widehat{\mathcal{R}}_s(f_s) - \widehat{\mathcal{R}}_s(f_{\mathcal{F}}^*)$, with \widehat{f}_s as in Definition 1.8,
- (B) a (uniform⁸) *generalization error* $\varepsilon^{\text{gen}} := \sup_{f \in \mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_s(f)| \geq \max\{\mathcal{R}(f_s) - \widehat{\mathcal{R}}_s(f_s), \widehat{\mathcal{R}}_s(f_{\mathcal{F}}^*) - \mathcal{R}(f_{\mathcal{F}}^*)\}$, and
- (C) an *approximation error* $\varepsilon^{\text{approx}} := \mathcal{R}(f_{\mathcal{F}}^*) - \mathcal{R}^*$,

see also Figure 1.2. The approximation error is decreasing when enlarging the hypothesis set, but taking $\mathcal{F} = \mathcal{M}(\mathcal{X}, \mathcal{Y})$ prevents controlling the generalization error, see also Theorem 1.13. This suggests a sweet-spot for the complexity of our hypothesis set \mathcal{F} and is usually referred to as the *bias-variance trade-off*, see also Figure 1.4 below. In the next sections, we will sketch mathematical ideas to tackle each of the errors in (A)–(C) in the context of deep learning. Observe that we bound the generalization and optimization error with respect to the empirical risk $\widehat{\mathcal{R}}_s$ and its minimizer \widehat{f}_s which is motivated by the fact that in deep-learning-based applications one typically tries to minimize variants of $\widehat{\mathcal{R}}_s$.

1.2.1 Optimization

The first error in the decomposition of (1.4) is the optimization error: ε^{opt} . This error is primarily influenced by the numerical algorithm \mathcal{A} that is used to find the model f_s in a hypothesis set of NNs for given training data $s \in \mathcal{Z}^m$. We will focus on the typical setting where such an algorithm tries to approximately minimize the empirical risk $\widehat{\mathcal{R}}_s$. While there are many conceivable methods to solve this minimization problem, by far the most common are gradient-based methods. The main reason for the popularity of gradient-based

⁸Although this uniform deviation can be a coarse estimate it is frequently considered to allow for the application of uniform laws of large numbers from the theory of empirical processes.

methods is that for FC networks as in Definition 1.4, the accurate and efficient computation of pointwise derivatives $\nabla_{\theta}\Phi_a(x, \theta)$ is possible by means of automatic differentiation, a specific form of which is often referred to as the *backpropagation algorithm* [Kel60, Dre62, Lin70, RHW86, GW08]. This numerical scheme is also applicable in general settings, such as, when the architecture of the NN is given by a general directed acyclic graph. Using these pointwise derivatives, one usually attempts to minimize the empirical risk $\widehat{\mathcal{R}}_s$ by updating the parameters θ according to a variant of *stochastic gradient descent* (SGD), which we shall review below in a general formulation:

Algorithm 1: Stochastic gradient descent

Input : Differentiable function $r: \mathbb{R}^p \rightarrow \mathbb{R}$, sequence of step-sizes $\eta_k \in (0, \infty)$, $k \in [K]$, \mathbb{R}^p -valued random variable $\Theta^{(0)}$.

Output : Sequence of \mathbb{R}^p -valued random variables $(\Theta^{(k)})_{k=1}^K$.

for $k = 1, \dots, K$ **do**

 Let $D^{(k)}$ be a random variable such that $\mathbb{E}[D^{(k)} | \Theta^{(k-1)}] = \nabla r(\Theta^{(k-1)})$;

 Set $\Theta^{(k)} := \Theta^{(k-1)} - \eta_k D^{(k)}$;

end

If $D^{(k)}$ is chosen deterministically in Algorithm 1, i.e., $D^{(k)} = \nabla r(\Theta^{(k-1)})$, then the algorithm is known as *gradient descent*. To minimize the empirical loss, we apply SGD with $r: \mathbb{R}^{P(N)} \rightarrow \mathbb{R}$ set to $r(\theta) = \widehat{\mathcal{R}}_s(\Phi_a(\cdot, \theta))$. More concretely, one might choose a *batch-size* $m' \in \mathbb{N}$ with $m' \leq m$ and consider the iteration

$$\Theta^{(k)} := \Theta^{(k-1)} - \frac{\eta_k}{m'} \sum_{z \in S'} \nabla_{\theta} \mathcal{L}(\Phi_a(\cdot, \Theta^{(k-1)}), z), \quad (1.5)$$

where S' is a so-called *mini-batch* of size $|S'| = m'$ chosen uniformly⁹ at random from the training data s . The sequence of step-sizes $(\eta_k)_{k \in \mathbb{N}}$ is often called *learning rate* in this context. Stopping at step K , the output of a deep learning algorithm \mathcal{A} is then given by

$$f_s = \mathcal{A}(s) = \Phi_a(\cdot, \bar{\theta}),$$

where $\bar{\theta}$ can be chosen to be the realization of the last parameter $\Theta^{(K)}$ of (1.5) or a convex combination of $(\Theta^{(k)})_{k=1}^K$ such as the mean.

Algorithm 1 was originally introduced in [RM51] in the context of finding the root of a nondecreasing function from noisy measurements. Shortly afterwards this idea was applied to find a unique minimum of a Lipschitz-regular function that has no flat regions away from the global minimum [KW52].

In some regimes, we can guarantee convergence of SGD at least in expectation, see [NY83, NJLS09, SSSS09], [SDR14, Section 5.9], [SSBD14, Chapter 14]. One prototypical convergence guarantee that is found in the aforementioned references in various forms is stated below.

Theorem 1.14 (Convergence of SGD). *Let $p, K \in \mathbb{N}$ and let $r: \mathbb{R}^p \supset B_1(0) \rightarrow \mathbb{R}$ be differentiable and convex. Further let $(\Theta^{(k)})_{k=1}^K$ be the output of Algorithm 1 with initialization $\Theta^{(0)} = 0$, step-sizes $\eta_k = K^{-1/2}$, $k \in [K]$, and random variables $(D^{(k)})_{k=1}^K$ satisfying that $\|D^{(k)}\|_2 \leq 1$ almost surely for all $k \in [K]$. Then*

$$\mathbb{E}[r(\bar{\Theta})] - r(\theta^*) \leq \frac{1}{\sqrt{K}},$$

where $\bar{\Theta} := \frac{1}{K} \sum_{k=1}^K \Theta^{(k)}$ and $\theta^* \in \arg \min_{\theta \in B_1(0)} r(\theta)$.

Theorem 1.14 can be strengthened to yield a faster convergence rate if the convexity is replaced by strict convexity. If r is not convex, then convergence to a global minimum can in general not be guaranteed. In fact, in that case, stochastic gradient descent may converge to a local, non-global minimum, see Figure 1.3 for an example.

⁹We remark that in practice one typically picks S' by selecting a subset of training data in a way to cover the full training data after one *epoch* of $\lceil m/m' \rceil$ many steps. This, however, does not necessarily yield an unbiased estimator $D^{(k)}$ of $\nabla_{\theta} r(\Theta^{(k-1)})$ given $\Theta^{(k-1)}$.

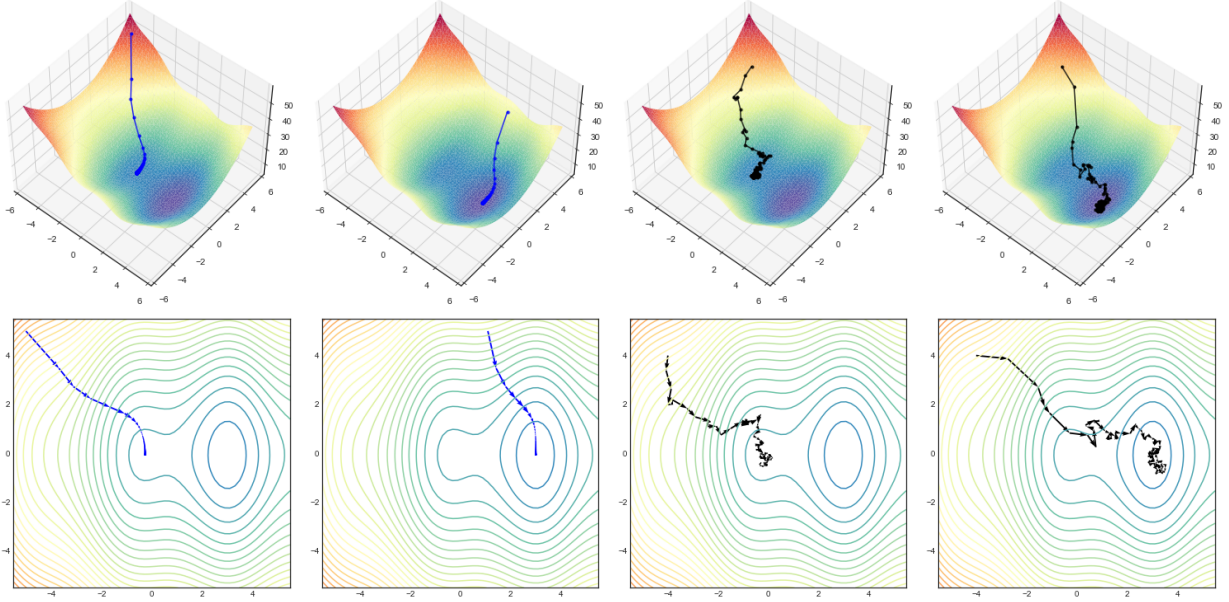


Figure 1.3: Examples of the dynamics of gradient descent (left) and stochastic gradient descent (right) for an objective function with one non-global minimum next to the global minimum. We see that depending on the initial condition and also on fluctuations in the stochastic part of SGD the algorithm can fail or succeed in finding the global minimum.

Moreover, gradient descent, i.e., the deterministic version of Algorithm 1, will stop progressing if at any point the gradient of r vanishes. This is the case in every stationary point of r . A stationary point is either a local minimum, a local maximum, or a saddle point. One would expect that if the direction of the step $D^{(k)}$ in Algorithm 1 is not deterministic, then the random fluctuations may allow the iterates to escape saddle points. Indeed, results guaranteeing convergence to local minima exist under various conditions on the type of saddle points that r admits, [NJLS09, GL13, GHJY15, LSJR16, JKNvW20].

In addition, many methods that improve the convergence by, for example, introducing more elaborate step-size rules or a momentum term have been established. We shall not review these methods here, but instead refer to [GBC16, Chapter 8] for an overview.

1.2.2 Approximation

Generally speaking, NNs, even FC NNs (see Definition 1.4) with only $L = 2$ layers, are universal approximators, meaning that under weak conditions on the activation function ϱ they can approximate any continuous function on a compact set up to arbitrary precision [Cyb89, Fun89, HSW89, LLPS93].

Theorem 1.15 (Universal approximation theorem). *Let $d \in \mathbb{N}$, let $K \subset \mathbb{R}^d$ be compact, and let $\varrho \in L_{\text{loc}}^\infty(\mathbb{R})$ be an activation function such that the closure of the points of discontinuity of ϱ is a Lebesgue null set. Further let*

$$\tilde{\mathcal{F}} := \bigcup_{n \in \mathbb{N}} \mathcal{F}_{((d,n,1), \varrho)}$$

be the corresponding set of two-layer NN realizations. Then it holds that $C(K) \subset \text{cl}(\tilde{\mathcal{F}})$ (where the closure is taken with respect to the topology induced by the $L^\infty(K)$ -norm) if and only if there does not exist a polynomial $p: \mathbb{R} \rightarrow \mathbb{R}$ with $p = \varrho$ almost everywhere.

The theorem can be proven by the theorem of Hahn–Banach, which implies that $\tilde{\mathcal{F}}$ being dense in some

real normed vector space \mathcal{S} is equivalent to the following condition: For all non-trivial functionals $F \in \mathcal{S}' \setminus \{0\}$ from the topological dual space of \mathcal{S} there exist parameters $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that

$$F(\varrho(\langle w, \cdot \rangle + b)) \neq 0.$$

In case of $\mathcal{S} = C(K)$ we have by the Riesz–Markov–Kakutani representation theorem that \mathcal{S}' is the space of signed Borel measures on K , see [Rud06]. Therefore, Theorem 1.15 holds, if ϱ is such that, for a signed Borel measure μ ,

$$\int_K \varrho(\langle w, x \rangle + b) d\mu(x) = 0 \quad (1.6)$$

for all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ implies that $\mu = 0$. An activation function ϱ satisfying this condition is called *discriminatory*. It is not hard to see that any sigmoidal ϱ is discriminatory. Indeed, assume that ϱ satisfies (1.6) for all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Since for every $x \in \mathbb{R}^d$ it holds that $\varrho(ax + b) \rightarrow \mathbb{1}_{(0, \infty)}(x) + \varrho(b)\mathbb{1}_{\{0\}}(x)$ for $a \rightarrow \infty$, we conclude by superposition and passing to the limit that for all $c_1, c_2 \in \mathbb{R}$ and $w \in \mathbb{R}^d$, $b \in \mathbb{R}$

$$\int_K \mathbb{1}_{[c_1, c_2]}(\langle w, x \rangle + b) d\mu(x) = 0.$$

Representing the exponential function $x \mapsto e^{-2\pi i x}$ as the limit of sums of elementary functions yields that $\int_K e^{-2\pi i(\langle w, x \rangle + b)} d\mu(x) = 0$ for all $w \in \mathbb{R}^d$, $b \in \mathbb{R}$. Hence, the Fourier transform of μ vanishes which implies that $\mu = 0$.

Theorem 1.15 addresses a uniform approximation problem on a general compact set. If we are given a finite number of points and only care about good approximation at these points, then one can ask if this approximation problem is potentially simpler. Below we see that, if the number of neurons is larger or equal to the number of data points, then one can always interpolate, i.e., exactly fit the data on a given finite number of points.

Proposition 1.1 (Interpolation). *Let $d, m \in \mathbb{N}$, let $x^{(i)} \in \mathbb{R}^d$, $i \in [m]$, with $x^{(i)} \neq x^{(j)}$ for $i \neq j$, let $\varrho \in C(\mathbb{R})$, and assume that ϱ is not a polynomial. Then, there exist parameters $\theta^{(1)} \in \mathbb{R}^{m \times d} \times \mathbb{R}^m$ with the following property: For every $k \in \mathbb{N}$ and every sequence of labels $y^{(i)} \in \mathbb{R}^k$, $i \in [m]$, there exist parameters $\theta^{(2)} = (W^{(2)}, 0) \in \mathbb{R}^{k \times m} \times \mathbb{R}^k$ for the second layer of the NN architecture $a = ((d, m, k), \varrho)$ such that*

$$\Phi_a(x^{(i)}, (\theta^{(1)}, \theta^{(2)})) = y^{(i)}, \quad i \in [m].$$

Let us sketch the proof in the following. First, note that Theorem 1.15 also holds for functions $g \in C(K, \mathbb{R}^m)$ with multi-dimensional output by approximating each one-dimensional component $x \mapsto (g(x))_i$ and stacking the resulting networks. Second, one can add an additional row containing only zeros to the weight matrix $W^{(1)}$ of the approximating neural network as well as an additional entry to the vector $b^{(1)}$. The effect of this is that we obtain an additional neuron with constant output. Since $\varrho \neq 0$, we can choose $b^{(1)}$ such that the output of this neuron is not zero. Therefore, we can include the bias vector $b^{(2)}$ of the second layer into the weight matrix $W^{(2)}$, see also Remark 1.5. Now choose $g \in C(\mathbb{R}^m, \mathbb{R}^m)$ to be a function satisfying $g(x^{(i)}) = e^{(i)}$, $i \in [m]$, where $e^{(i)} \in \mathbb{R}^m$ denotes the i -th standard basis vector. By the discussion before there exists a neural network architecture $\tilde{a} = ((d, n, m), \varrho)$ and parameters $\tilde{\theta} = ((\tilde{W}^{(1)}, \tilde{b}^{(1)}), (\tilde{W}^{(2)}, 0))$ such that

$$\|\Phi_{\tilde{a}}(\cdot, \tilde{\theta}) - g\|_{L^\infty(K)} < \frac{1}{m}, \quad (1.7)$$

where K is a compact set with $x^{(i)} \in K$, $i \in [m]$. Let us abbreviate the output of the activations in the first layer evaluated at the input features by

$$\tilde{A} := \left[\varrho(\tilde{W}^{(1)}(x^{(1)} + \tilde{b}^{(1)})) \dots \varrho(\tilde{W}^{(1)}(x^{(m)} + \tilde{b}^{(1)})) \right] \in \mathbb{R}^{n \times m}.$$

The equivalence of the max and operator norm and (1.7) establish that

$$\|\tilde{W}^{(2)} \tilde{A} - \mathbf{I}_m\|_{\text{op}} \leq m \max_{i,j \in [m]} |(\tilde{W}^{(2)} \tilde{A} - \mathbf{I}_m)_{i,j}| = m \max_{j \in [m]} \|\Phi_{\tilde{a}}(x^{(j)}, \tilde{\theta}) - g(x^{(j)})\|_\infty < 1,$$

where I_m denotes the $m \times m$ identity matrix. Thus, the matrix $\widetilde{W}^{(2)} \widetilde{A} \in \mathbb{R}^{m \times m}$ needs to have full rank and we can extract m linearly independent rows from \widetilde{A} resulting in an invertible matrix $A \in \mathbb{R}^{m \times m}$. Now, we define the desired parameters $\theta^{(1)}$ for the first layer by extracting the corresponding rows from $\widetilde{W}^{(1)}$ and $\widetilde{b}^{(1)}$ and the parameters $\theta^{(2)}$ of the second layer by

$$W^{(2)} := [y^{(1)} \dots y^{(m)}] A^{-1} \in \mathbb{R}^{k \times m}.$$

This proves that with any discriminatory activation function we can interpolate arbitrary training data $(x^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \mathbb{R}^k$, $i \in [m]$, using a two-layer NN with m hidden neurons, i.e., $\mathcal{O}(m(d+k))$ parameters.

One can also first project the input features to a one-dimensional line where they are separated and then apply Proposition 1.1 with $d = 1$. For nearly all activation functions, this can be represented by a three-layer NN using only $\mathcal{O}(d + mk)$ parameters¹⁰.

Beyond interpolation results, one can obtain a quantitative version of Theorem 1.15 if one knows additional regularity properties of the Bayes optimal function f^* , such as smoothness, compositionality, and symmetries. For surveys on such results, we refer the reader to [DHP20, GRK20]. For instructive purposes, we review one such result, which can be found in [Mha96, Theorem 2.1], below:

Theorem 1.16 (Approximation of smooth functions). *Let $d, k \in \mathbb{N}$ and $p \in [1, \infty]$. Further let $\varrho \in C^\infty(\mathbb{R})$ and assume that ϱ is not a polynomial. Then there exists a constant $c \in (0, \infty)$ with the following property: For every $n \in \mathbb{N}$ there exist parameters $\theta^{(1)} \in \mathbb{R}^{n \times d} \times \mathbb{R}^n$ for the first layer of the NN architecture $a = ((d, n, 1), \varrho)$ such that for every $g \in W^{k,p}((0,1)^d)$ it holds that*

$$\inf_{\theta^{(2)} \in \mathbb{R}^{1 \times n} \times \mathbb{R}} \|\Phi_a(\cdot, (\theta^{(1)}, \theta^{(2)})) - g\|_{L^p((0,1)^d)} \leq cn^{-\frac{d}{k}} \|g\|_{W^{k,p}((0,1)^d)}.$$

Theorem 1.16 shows that NNs achieve the same optimal approximation rates that, for example, spline-based approximation yields for smooth functions. The idea behind this theorem is based on a strategy that is employed repeatedly throughout the literature. This is the idea of re-approximating classical approximation methods by NNs and thereby transferring the approximation rates of these methods to NNs. In the example of Theorem 1.16, approximation by polynomials is used. The idea is that due to the non-vanishing derivatives of the activation function¹¹, one can approximate every univariate polynomial via divided differences of the activation function. Specifically, accepting unbounded parameter magnitudes, for any activation function $\varrho: \mathbb{R} \rightarrow \mathbb{R}$ which is p -times differentiable at some point $\lambda \in \mathbb{R}$ with $\varrho^{(p)}(\lambda) \neq 0$, one can approximate the monomial $x \mapsto x^p$ on a compact set $K \subset \mathbb{R}$ up to arbitrary precision by a fixed-size NN via rescaled p -th order difference quotients as

$$\lim_{h \rightarrow 0} \sup_{x \in K} \left| \sum_{i=0}^p \frac{(-1)^i \binom{p}{i}}{h^p \varrho^{(p)}(\lambda)} \varrho((p/2 - i)hx + \lambda) - x^p \right| = 0. \quad (1.8)$$

Let us end this subsection by clarifying the connection of the approximation results above to the error decomposition of (1.4). Consider, for simplicity, a regression task with quadratic loss. Then, the approximation error $\varepsilon^{\text{approx}}$ equals a common L^2 -error

$$\begin{aligned} \varepsilon^{\text{approx}} &= \mathcal{R}(f_{\mathcal{F}}^*) - \mathcal{R}^* \stackrel{(*)}{=} \int_{\mathcal{X}} (f_{\mathcal{F}}^*(x) - f^*(x))^2 d\mathbb{P}_X(x) \\ &\stackrel{(*)}{=} \min_{f \in \mathcal{F}} \|f - f^*\|_{L^2(\mathbb{P}_X)}^2 \\ &\leq \min_{f \in \mathcal{F}} \|f - f^*\|_{L^\infty(\mathcal{X})}^2, \end{aligned}$$

where the identities marked by $(*)$ follow from Lemma 1.1. Hence, Theorem 1.15 postulates that $\varepsilon^{\text{approx}} \rightarrow 0$ for increasing NN sizes, whereas Theorem 1.16 additionally explains how fast $\varepsilon^{\text{approx}}$ converges to 0.

¹⁰To avoid the $m \times d$ weight matrix (without using shared parameters as in [ZBH⁺17]) one interjects an approximate one-dimensional identity [PV18, Definition 2.5], which can be arbitrarily well approximated by a NN with architecture $a = ((1, 2, 1), \varrho)$ given that $\varrho'(\lambda) \neq 0$ for some $\lambda \in \mathbb{R}$, see (1.8) below.

¹¹The Baire category theorem ensures that for a non-polynomial $\varrho \in C^\infty(\mathbb{R})$ there exists $\lambda \in \mathbb{R}$ with $\varrho^{(p)}(\lambda) \neq 0$ for all $p \in \mathbb{N}$, see, e.g., [Don69, Chapter 10].

1.2.3 Generalization

Towards bounding the generalization error $\varepsilon^{\text{gen}} = \sup_{f \in \mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)|$, one observes that, for every $f \in \mathcal{F}$, Assumption 1.10 ensures that $\mathcal{L}(f, Z^{(i)})$, $i \in [m]$, are i.i.d. random variables. Thus, one can make use of concentration inequalities to bound the deviation of the empirical risk $\widehat{\mathcal{R}}_S(f) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(f, Z^{(i)})$ from its expectation $\mathcal{R}(f)$. For instance, assuming boundedness¹² of the loss, Hoeffding's inequality [Hoe63] and a union bound directly imply the following generalization guarantee for countable, weighted hypothesis sets \mathcal{F} , see, e.g., [BBL03].

Theorem 1.17 (Generalization bound for countable, weighted hypothesis sets). *Let $m \in \mathbb{N}$, $\delta \in (0, 1)$ and assume that \mathcal{F} is countable. Further let p be a probability distribution on \mathcal{F} and assume that $\mathcal{L}(f, Z) \in [0, 1]$ almost surely for every $f \in \mathcal{F}$. Then with probability $1 - \delta$ (with respect to repeated sampling of \mathbb{P}_Z^m -distributed training data S) it holds for every $f \in \mathcal{F}$ that*

$$|\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)| \leq \sqrt{\frac{\ln(1/p(f)) + \ln(2/\delta)}{2m}}.$$

While the weighting p needs to be chosen before seeing the training data, one could incorporate prior information on the learning algorithm \mathcal{A} . For finite hypothesis sets without prior information, setting $p(f) = 1/|\mathcal{F}|$ for every $f \in \mathcal{F}$, Theorem 1.17 implies that, with high probability, it holds that

$$\varepsilon^{\text{gen}} \lesssim \sqrt{\frac{\ln(|\mathcal{F}|)}{m}}. \quad (1.9)$$

Again, one notices that, in line with the bias-variance trade-off, the generalization bound is increasing with the size of the hypothesis set $|\mathcal{F}|$. Although in practice the parameters $\theta \in \mathbb{R}^{P(N)}$ of a NN are discretized according to floating-point arithmetic, the corresponding quantities $|\mathcal{F}_a|$ or $|\mathcal{F}_{a,\text{sgn}}|$ would be huge and we need to find a replacement for the finiteness condition.

We will focus on binary classification tasks and present a main result of VC theory which is to a great extent derived from the work of Vladimir Vapnik and Alexey Chervonenkis [VC71]. While in (1.9) we counted the number of functions in \mathcal{F} , we now refine this analysis to the number of functions restricted to a finite subset of \mathcal{X} , given by the *growth function*

$$\text{growth}(m, \mathcal{F}) := \max_{(x^{(i)})_{i=1}^m \in \mathcal{X}^m} |\{f|_{(x^{(i)})_{i=1}^m} : f \in \mathcal{F}\}|.$$

The growth function can be interpreted as the maximal number of classification patterns in $\{-1, 1\}^m$ which functions in \mathcal{F} can realize on m points and thus $\text{growth}(m, \mathcal{F}) \leq 2^m$. The asymptotic behavior of the growth function is determined by a single intrinsic dimension of our hypothesis set \mathcal{F} , the so-called *VC-dimension*

$$\text{VCdim}(\mathcal{F}) := \sup \{m \in \mathbb{N} \cup \{0\} : \text{growth}(m, \mathcal{F}) = 2^m\},$$

which defines the largest number of points such that \mathcal{F} can realize any classification pattern, see, e.g., [AB99, BBL03]. There exist various results on VC-dimensions of NNs with different activation functions, see, for instance, [BH89, KM97, BMM98, Sak99]. We present the result of [BMM98] for piecewise polynomial activation functions ϱ . It establishes a bound on the VC-dimension of hypothesis sets of NNs for classification tasks $\mathcal{F}_{(N, \varrho), \text{sgn}}$ that scales, up to logarithmic factors, linear in the number of parameters $P(N)$ and quadratic in the number of layers L .

Theorem 1.18 (VC-dimension of neural network hypothesis sets). *Let ϱ be a piecewise polynomial activation function. Then there exists a constant $c \in (0, \infty)$ such that for every $L \in \mathbb{N}$ and $N \in \mathbb{N}^{L+1}$ it holds that*

$$\text{VCdim}(\mathcal{F}_{(N, \varrho), \text{sgn}}) \leq c(P(N)L \log(P(N)) + P(N)L^2).$$

¹²Note that for our classification tasks in Definition 1.2 it holds that $\mathcal{L}(f, Z) \in \{0, 1\}$ for every $f \in \mathcal{F}$. For the regression tasks, one typically assumes boundedness conditions, such as $|Y| \leq c$ and $\sup_{f \in \mathcal{F}} |f(X)| \leq c$ almost surely for some $c \in (0, \infty)$, which yields that $\sup_{f \in \mathcal{F}} |\mathcal{L}(f, Z)| \leq 4c^2$.

Given $(x^{(i)})_{i=1}^m \in \mathcal{X}^m$, there exists a partition of $\mathbb{R}^{P(N)}$ such that $\Phi(x^{(i)}, \cdot)$, $i \in [m]$, are polynomials on each region of the partition. The proof of Theorem 1.18 is based on bounding the number of such regions and the number of classification patterns of a set of polynomials.

A finite VC-dimension ensures the following generalization bound [Tal94, AB99]:

Theorem 1.19 (VC-dimension generalization bound). *There exists a constant $c \in (0, \infty)$ with the following property: For every classification task as in Definition 1.2, every \mathcal{Z} -valued random variable Z , and every $m \in \mathbb{N}$, $\delta \in (0, 1)$ it holds with probability $1 - \delta$ (with respect to repeated sampling of \mathbb{P}_Z^m -distributed training data S) that*

$$\sup_{f \in \mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)| \leq c \sqrt{\frac{\text{VCdim}(\mathcal{F}) + \log(1/\delta)}{m}}.$$

In summary, using NN hypothesis sets $\mathcal{F}_{(N, \varrho), \text{sgn}}$ with a fixed depth and piecewise polynomial activation ϱ for a classification task, with high probability it holds that

$$\varepsilon^{\text{gen}} \lesssim \sqrt{\frac{P(N) \log(P(N))}{m}}. \quad (1.10)$$

In the remainder of this section we will sketch a proof of Theorem 1.19 and, in doing so, present further concepts and complexity measures connected to generalization bounds. We start by observing that McDiarmid's inequality [McD89] ensures that ε^{gen} is sharply concentrated around its expectation, i.e., with probability $1 - \delta$ it holds that¹³

$$|\varepsilon^{\text{gen}} - \mathbb{E}[\varepsilon^{\text{gen}}]| \lesssim \sqrt{\frac{\log(1/\delta)}{m}}. \quad (1.11)$$

To estimate the expectation of the uniform generalization error we employ a *symmetrization argument* [GZ84]. Define $\mathcal{G} := \mathcal{L} \circ \mathcal{F} := \{\mathcal{L}(f, \cdot) : f \in \mathcal{F}\}$, let $\tilde{S} = (\tilde{Z}^{(i)})_{i=1}^m \sim \mathbb{P}_Z^m$ be a test data set independent of S , and note that $\mathcal{R}(f) = \mathbb{E}[\widehat{\mathcal{R}}_{\tilde{S}}(f)]$. By properties of the conditional expectation and Jensen's inequality it holds that

$$\begin{aligned} \mathbb{E}[\varepsilon^{\text{gen}}] &= \mathbb{E}\left[\sup_{f \in \mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)|\right] = \mathbb{E}\left[\sup_{g \in \mathcal{G}} \frac{1}{m} \left| \sum_{i=1}^m \mathbb{E}[g(\tilde{Z}^{(i)}) - g(Z^{(i)}) | S] \right|\right] \\ &\leq \mathbb{E}\left[\sup_{g \in \mathcal{G}} \frac{1}{m} \left| \sum_{i=1}^m g(\tilde{Z}^{(i)}) - g(Z^{(i)}) \right|\right] \\ &= \mathbb{E}\left[\sup_{g \in \mathcal{G}} \frac{1}{m} \left| \sum_{i=1}^m \tau_i (g(\tilde{Z}^{(i)}) - g(Z^{(i)})) \right|\right] \\ &\leq 2\mathbb{E}\left[\sup_{g \in \mathcal{G}} \frac{1}{m} \left| \sum_{i=1}^m \tau_i g(Z^{(i)}) \right|\right], \end{aligned}$$

where we used that multiplications with Rademacher variables $(\tau_1, \dots, \tau_m) \sim \mathcal{U}(\{-1, 1\}^m)$ only amount to interchanging $Z^{(i)}$ with $\tilde{Z}^{(i)}$ which has no effect on the expectation, since $Z^{(i)}$ and $\tilde{Z}^{(i)}$ have the same distribution. The quantity

$$\mathfrak{R}_m(\mathcal{G}) := \mathbb{E}\left[\sup_{g \in \mathcal{G}} \left| \frac{1}{m} \sum_{i=1}^m \tau_i g(Z^{(i)}) \right|\right]$$

is called the *Rademacher complexity*¹⁴ of \mathcal{G} . One can also prove a corresponding lower bound [vdVW97], i.e.,

$$\mathfrak{R}_m(\mathcal{G}) - \frac{1}{\sqrt{m}} \lesssim \mathbb{E}[\varepsilon^{\text{gen}}] \lesssim \mathfrak{R}_m(\mathcal{G}). \quad (1.12)$$

¹³For precise conditions to ensure that the expectation of ε^{gen} is well-defined, we refer the reader to [vdVW97, Dud14].

¹⁴Due to our decomposition in (1.4), we want to uniformly bound the absolute value of the difference between the risk and the empirical risk. It is also common to just bound $\sup_{f \in \mathcal{F}} \mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)$ leading to a definition of the Rademacher complexity without the absolute values which can be easier to deal with.

Now we use a *chaining method* to bound the Rademacher complexity of \mathcal{F} by covering numbers on different scales. Specifically, Dudley’s entropy integral [Dud67, LT91] implies that

$$\mathfrak{R}_m(\mathcal{G}) \lesssim \mathbb{E} \left[\int_0^\infty \sqrt{\frac{\log N_\alpha(\mathcal{G}, d_S)}{m}} d\alpha \right], \quad (1.13)$$

where

$$N_\alpha(\mathcal{G}, d_S) := \inf \left\{ |G| : G \subset \mathcal{G}, \mathcal{G} \subset \bigcup_{g \in G} B_\alpha^{d_S}(g) \right\}$$

denotes the covering number with respect to the (random) pseudometric given by

$$d_S(f, g) = d_{(Z^{(i)})_{i=1}^m}(f, g) := \sqrt{\frac{1}{m} \sum_{i=1}^m (f(Z^{(i)}) - g(Z^{(i)}))^2}.$$

For the 0-1 loss $\mathcal{L}(f, z) = \mathbb{1}_{(-\infty, 0)}(yf(x)) = (1 - f(x)y)/2$, we can get rid of the loss function by the fact that

$$N_\alpha(\mathcal{G}, d_S) = N_{2\alpha}(\mathcal{F}, d_{(X^{(i)})_{i=1}^m}). \quad (1.14)$$

The proof is completed by combining the inequalities in (1.11), (1.12), (1.13) and (1.14) with a result of David Haussler [Hau95] which shows that for $\alpha \in (0, 1)$ we have

$$\log(N_\alpha(\mathcal{F}, d_{(X^{(i)})_{i=1}^m})) \lesssim \text{VCdim}(\mathcal{F}) \log(1/\alpha). \quad (1.15)$$

We remark that this resembles a typical behavior of covering numbers. For instance, the logarithm of the covering number $\log(N_\alpha(\mathcal{M}))$ of a compact d -dimensional Riemannian manifold \mathcal{M} essentially scales like $d \log(1/\alpha)$. Finally, note that there exists a similar bound to the one in (1.15) for bounded regression tasks making use of the so-called *fat-shattering dimension* [MV03, Theorem 1].

1.3 Do we need a new theory?

Despite the already substantial insight that the classical theories provide, a lot of open questions remain. We will outline these questions below. The remainder of this article then collects modern approaches to explain the following issues:

Why do large neural networks not overfit? In Subsection 1.2.2, we have observed that three-layer NNs with commonly used activation functions and only $\mathcal{O}(d + m)$ parameters can interpolate any training data $(x^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \mathbb{R}$, $i \in [m]$. While this specific representation might not be found in practice, [ZBH⁺17] indeed trained convolutional¹⁵ NNs with ReLU activation function and about 1.6 million parameters to achieve zero empirical risk on $m = 50000$ training images of the CIFAR10 dataset [KH09] with 32×32 pixels per image, i.e., $d = 1024$. For such large NNs, generalization bounds scaling with the number of parameters $P(N)$ as the VC-dimension bound in (1.10) are vacuous. However, they observed close to state-of-the-art generalization performance¹⁶.

Generally speaking, NNs in practice are observed to generalize well despite having more parameters than training samples (usually referred to as *overparametrization*) and approximately interpolating the training data (usually referred to as *overfitting*). As we cannot perform any better on the training data, there is no trade-off between fit to training data and complexity of the hypothesis set \mathcal{F} happening, seemingly contradicting the classical bias-variance trade-off of statistical learning theory. This is quite surprising, especially given the following additional empirical observations in this regime, see [NTS14, ZBH⁺17, NBMS17, BHMM19, NKB⁺20]:

¹⁵The basic definition of a convolutional NN will be given in Section 6. In [ZBH⁺17] more elaborate versions such as an *Inception* architecture [SLJ⁺15] are employed.

¹⁶In practice one usually cannot measure the risk $\mathcal{R}(f_s)$ and instead evaluates the performance of a trained model f_s by $\widehat{\mathcal{R}}_{\tilde{s}}(f_s)$ using test data \tilde{s} , i.e., realizations of i.i.d. random variables distributed according to \mathbb{P}_Z and drawn independently of the training data. In this context one often calls $\mathcal{R}_s(f_s)$ the *training error* and $\mathcal{R}_{\tilde{s}}(f_s)$ the *test error*.

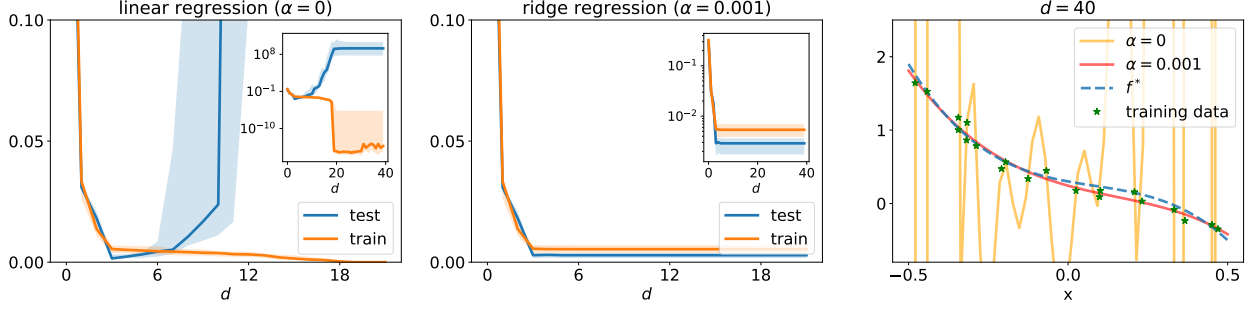


Figure 1.4: The first plot (and its semi-log inset) shows median and interquartile range of the test and training errors of ten independent linear regressions with $m = 20$ samples, polynomial input features $X = (1, Z, \dots, Z^d)$ of degree $d \in [40]$, and labels $Y = f^*(Z) + \nu$, where $Z \sim \mathcal{U}([-0.5, 0.5])$, f^* is a polynomial of degree three, and $\nu \sim \mathcal{N}(0, 0.01)$. This clearly reflects the classical u-shaped bias-variance curve with a sweet-spot at $d = 3$ and drastic overfitting beyond the interpolation threshold at $d = 20$. However, the second plot shows that we can control the complexity of our hypothesis set of linear models by restricting the Euclidean norm of their parameters using ridge regression with a small regularization parameter $\alpha = 10^{-3}$, i.e., minimizing the regularized empirical risk $\frac{1}{m} \sum_{i=1}^m (\Phi(X^{(i)}, \theta) - Y^{(i)})^2 + \alpha \|\theta\|_2^2$, where $\Phi(\cdot, \theta) = \langle \theta, \cdot \rangle$. Corresponding examples of \hat{f}_s are depicted in the last plot.

1. *Zero training error on random labels:* Zero empirical risk can also be achieved for random labels using the same architecture and training scheme with only slightly increased training time: This suggests that the considered hypothesis set of NNs \mathcal{F} can fit arbitrary binary labels, which would imply that $\text{VCdim}(\mathcal{F}) \approx m$ or $\mathfrak{R}_m(\mathcal{F}) \approx 1$ rendering our uniform generalization bounds in Theorem 1.19 and in (1.12) vacuous.
2. *Lack of explicit regularization:* The test error depends only mildly on explicit regularization like norm-based penalty terms or dropout (see [Gér17] for an explanation of different regularization methods): As such regularization methods are typically used to decrease the complexity of \mathcal{F} , one might ask if there is any *implicit* regularization (see Figure 1.4), constraining the range of our learning algorithm \mathcal{A} to some smaller, potentially data-dependent subset, i.e., $\mathcal{A}(s) \in \tilde{\mathcal{F}}_s \subsetneq \mathcal{F}$.
3. *Dependence on the initialization:* The same NN trained to zero empirical risk starting from different initializations can exhibit different test errors: This indicates that properties of the local minimum at f_s to which gradient descent converges might be correlated with its generalization.
4. *Interpolation of noisy training data:* One still observes low test error when training up to approximately zero empirical risk using a regression (or surrogate) loss on noisy training data. This is particularly interesting, as the noise is captured by the model but seems not to hurt generalization performance.
5. *Further overparametrization improves generalization performance:* Further increasing the NN size can lead to even lower test error: Together with the previous item, this might ask for a different treatment of models complex enough to fit the training data. According to the traditional lore “The training error tends to decrease whenever we increase the model complexity, that is, whenever we fit the data harder. However with too much fitting, the model adapts itself too closely to the training data, and will not generalize well (i.e., have large test error)”, [HTF01]. While this flawlessly describes the situation for certain machine learning tasks (see Figure 1.4), it seems not to be directly applicable here.

In summary, this suggests that the generalization performance of NNs depends on an interplay of the data distribution \mathbb{P}_Z combined with properties of the learning algorithm \mathcal{A} , such as the optimization procedure and its range. In particular, classical uniform bounds as in Item (B) of our error decomposition might only deliver insufficient explanation, see also [NK19]. The mismatch between predictions of classical theory and

the practical generalization performance of deep NNs is often referred to as *generalization puzzle*. In Section 2 we will present possible explanations for this phenomenon.

What is the role of depth? We have seen in Subsection 1.2.2 that NNs can closely approximate every function if they are sufficiently wide [Cyb89, Fun89, HSW89]. There are additional classical results that even provide a trade-off between the width and the approximation accuracy [CLM94, Mha96, MP99]. In these results, the central concept is the width of a NN. In modern applications, however at least as much focus if not more lies on the depth of the underlying architectures, which can have more than 1000 layers [HZRS16]. After all, the depth of NNs is responsible for the name of deep learning.

This consideration begs the question of whether there is a concrete mathematically quantifiable benefit of deep architectures over shallow NNs. Indeed, we will see effects of depth at many places throughout this manuscript. However, one of the aspects of deep learning that is most clearly affected by deep architectures is the approximation theoretical aspect. In this framework, we will discuss in Section 3 multiple approaches that describe the effect of depth.

Why do neural networks perform well in very high-dimensional environments? We have seen in Subsection 1.2.2 and will see in Section 3 that from the perspective of approximation theory deep NNs match the performance of the best classical approximation tool in virtually every task. In practice, we observe something that is even more astounding. In fact, NNs seem to perform incredibly well on tasks that no classical, non-specialized approximation method can even remotely handle. The approximation problem that we are talking about here is that of approximation of high-dimensional functions. Indeed, the classical *curse of dimensionality* [Bel52, NW09] postulates that essentially every approximation method deteriorates exponentially fast with increasing dimension.

For example, for the uniform approximation error of 1-Lipschitz continuous functions on a d -dimensional unit cube in the uniform norm, we have a lower bound of $\Omega(p^{-1/d})$, for $p \rightarrow \infty$, when approximating with a continuous scheme¹⁷ of p free parameters [DeV98].

On the other hand, in most applications, the input dimensions are massive. For example, the following datasets are typically used as benchmarks in image classification problems: MNIST [LBBH98] with 28×28 pixels per image, CIFAR-10/CIFAR-100 [KH09] with 32×32 pixels per image and ImageNet [DDS⁺09, KSH12] which contains high-resolution images that are typically down-sampled to 256×256 pixels. Naturally, in real-world applications, the input dimensions may well exceed those of these test problems. However, already for the simplest of the test cases above, the input dimension is $d = 784$. If we use $d = 784$ in the aforementioned lower bound for the approximation of 1-Lipschitz functions, then we require $\mathcal{O}(\varepsilon^{-784})$ parameters to achieve a uniform error of $\varepsilon \in (0, 1)$. Already for moderate ε this value will quickly exceed the storage capacity of any conceivable machine in this universe. Considering the aforementioned curse of dimensionality, it is puzzling to see that NNs perform adequately in this regime. In Section 4, we describe three approaches that offer explanations as to why deep NN-based approximation is not rendered meaningless in the context of high-dimensional input dimensions.

Why does stochastic gradient descent converge to good local minima despite the non-convexity of the problem? As mentioned in Subsection 1.2.1, a convergence guarantee of stochastic gradient descent to a global minimum is typically only given if the underlying objective function admits some form of convexity. However, the empirical risk of a NN, i.e., $\widehat{\mathcal{R}}_s(\Phi(\cdot, \theta))$, is typically not a convex function with respect to the parameters θ . For a simple intuitive reason why this function fails to be convex, it is instructive to consider the following example.

Example 1.20. *Consider the NN*

$$\Phi(x, \theta) = \theta_1 \varrho_R(\theta_3 x + \theta_5) + \theta_2 \varrho_R(\theta_4 x + \theta_6), \quad \theta \in \mathbb{R}^6, \quad x \in \mathbb{R},$$

¹⁷One can achieve better rates at the cost of discontinuous (with respect to the function to be approximated) parameter assignment. This can be motivated by the use of space-filling curves. In the context of NNs with piecewise polynomial activation functions, a rate of $p^{-2/d}$ can be achieved by very deep architectures [Yar18a, YZ20].

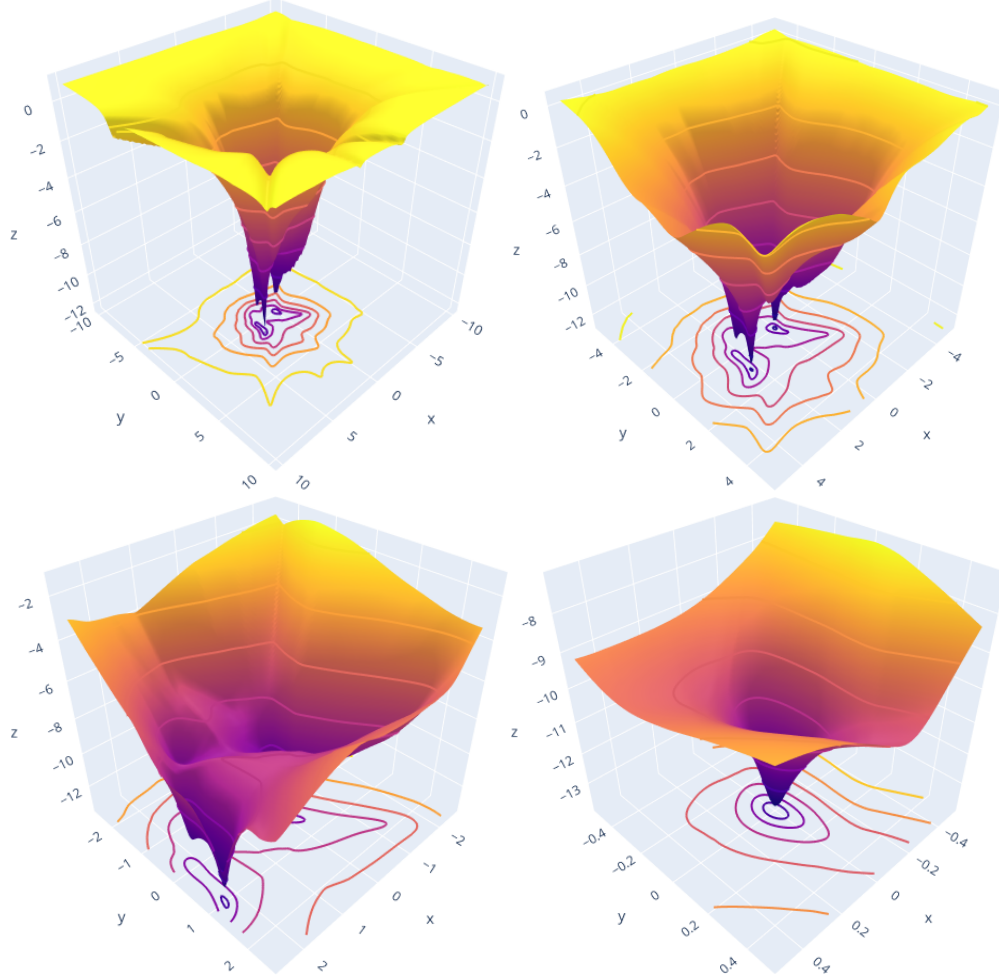


Figure 1.5: Two-dimensional projection of the loss landscape of a neural network with four layers and ReLU activation function on four different scales. From top-left to bottom-right, we zoom into the global minimum of the landscape.

with the ReLU activation function $\varrho_R(x) = \max\{0, x\}$. It is not hard to see that the two parameter values $\theta = (1, -1, 1, 1, 1, 0)$ and $\bar{\theta} = (-1, 1, 1, 1, 0, 1)$ produce the same realization function¹⁸, i.e., $\Phi(\cdot, \theta) = \Phi(\cdot, \bar{\theta})$. However, since $(\theta + \bar{\theta})/2 = (0, 0, 1, 1, 1/2, 1/2)$, we conclude that $\Phi(\cdot, (\theta + \bar{\theta})/2) = 0$. Clearly, for the data $s = ((-1, 0), (1, 1))$, we now have that

$$\widehat{\mathcal{R}}_s(\Phi(\cdot, \theta)) = \widehat{\mathcal{R}}_s(\Phi(\cdot, \bar{\theta})) = 0 \quad \text{and} \quad \widehat{\mathcal{R}}_s(\Phi(\cdot, (\theta + \bar{\theta})/2)) = \frac{1}{2},$$

showing the non-convexity of $\widehat{\mathcal{R}}_s$.

Given this non-convexity, Algorithm 1 faces serious challenges. Firstly, there may exist multiple suboptimal local minima. Secondly, the objective may exhibit saddle points, some of which may be of higher order, i.e., the Hessian vanishes. Finally, even if no suboptimal local minima exist, there may be extensive areas of the parameter space where the gradient is very small, so that escaping these regions can take a very long time.

¹⁸This corresponds to interchanging the two neurons in the hidden layer. In general it holds that the realization function of a FC NN is invariant under permutations of the neurons in a given hidden layer.

These issues are not mere theoretical possibilities, but will almost certainly arise. For example, [AHW96, SS18] show the existence of many suboptimal local minima in typical learning tasks. Moreover, for fixed-sized NNs, it has been shown in [BEG19, PRV20], that with respect to L^p -norms the set of NNs is generally a very non-convex and non-closed set. Also, the map $\theta \mapsto \Phi_a(\cdot, \theta)$ is not a quotient map, i.e., not continuously invertible when accounting for its non-injectivity. In addition, in various situations finding the global optimum of the minimization problem is shown to be NP-hard in general [BR89, Jud90, Ším02]. In Figure 1.5 we show the two-dimensional projection of a loss landscape, i.e., the projection of the graph of the function $\theta \mapsto \widehat{\mathcal{R}}_s(\Phi(\cdot, \theta))$. It is apparent from the visualization that the problem exhibits more than one minimum. We also want to add that in practice one neglects that the loss is only almost everywhere differentiable in case of piecewise smooth activation functions, such as the ReLU, although one could resort to subgradient methods [KL18].

In view of these considerations, the classical framework presented in Subsection 1.2.1 offers no explanation as to why deep learning works in practice. Indeed, in the survey [OM98, Section 1.4] the state of the art in 1998 was summarized by the following assessment: “There is no formula to guarantee that (1) the NN will converge to a good solution, (2) convergence is swift, or (3) convergence even occurs at all.”

Nonetheless, in applications, not only would an explanation of when and why SGD converges be extremely desirable, convergence is also quite often observed even though there is little theoretical explanation for it in the classical set-up. In Section 5, we collect modern approaches explaining why and when convergence occurs and can be guaranteed.

Which aspects of a neural network architecture affect the performance of deep learning? In the introduction to classical approaches to deep learning above, we have seen that in classical results, such as in Theorem 1.16, only the effect of few aspects of the NN architectures are considered. In Theorem 1.16 only the impact of the width of the NN was studied. In further approximation theorems below, e.g., in Theorems 2.1 and 3.2, we will additionally have a variable depth of NNs. However, for deeper architectures, there are many additional aspects of the architecture that could potentially affect the performance of the model for the associated learning task. For example, even for a standard FC NN with L layers as in Definition 1.4, there is a lot of flexibility in choosing the number of neurons $(N_1, \dots, N_{L-1}) \in \mathbb{N}^{L-1}$ in the hidden layers. One would expect that certain choices affect the capabilities of the NNs considerably and some choices are preferable over others. Note that, one aspect of the neural network architecture that can have a profound effect on the performance, especially regarding approximation theoretical aspects of the performance, is the choice of the activation function. For example, in [MP99, Yar21] activation functions were found that allow uniform approximation of continuous functions to arbitrary accuracy with fixed-size neural networks. In the sequel we will, however, focus on architectural aspects other than the activation function.

In addition, practitioners have invented an immense variety of NN architectures for specific problems. These include NNs with convolutional blocks [LBBH98], with skip connections [HZRS16], sparse connections [ZAP16, BBC17], batch normalization blocks [IS15], and many more. In addition, for sequential data, recurrent connections are used [RHW86] and these often have forget mechanisms [HS97] or other gates [CvMG⁺14] included in their architectures.

The choice of an appropriate NN architecture is essential to the success of many deep learning tasks. This goes so far, that frequently an architecture search is applied to find the most suitable one [ZL17, PGZ⁺18]. In most cases, though, the design and choice of the architecture is based on the intuition of the practitioner.

Naturally, from a theoretical point of view, this situation is not satisfactory. Instead, it would be highly desirable to have a mathematical theory guiding the choice of NN architectures. More concretely, one would wish for mathematical theorems that identify those architectures that work for a specific problem and those that will yield suboptimal results. In Section 6, we discuss various results that explain theoretically quantifiable effects of certain aspects or building blocks of NN architectures.

Which features of data are learned by deep architectures? It is commonly believed that the neurons of NNs constitute feature extractors in different levels of abstraction that correspond to the layers. This belief is partially grounded in experimental evidence as well as in drawing connections to the human visual

cortex, see [GBC16, Chapter 9.10].

Understanding the features that are learned can, in a way, be linked to understanding the reasoning with which a NN-based model ended up with its result. Therefore, analyzing the features that a NN learns constitutes a data-aware approach to understanding deep learning. Naturally, this falls outside of the scope of the classical theory, which is formulated in terms of optimization, generalization, and approximation errors.

One central obstacle towards understanding these features theoretically is that, at least for practical problems, the data distribution is unknown. However, one often has partial knowledge. One example is that in image classification it appears reasonable to assume that any classifier is translation and rotation invariant as well as invariant under small deformations. In this context, it is interesting to understand under which conditions trained NNs admit the same invariances.

Biological NNs such as the visual cortex are believed to be evolved in a way that is based on sparse multiscale representations of visual information [OF96]. Again, a fascinating question is whether NNs trained in practice can be shown to favor such multiscale representations based on sparsity or if the architecture is theoretically linked to sparse representations. We will discuss various approaches studying the features learned by neural networks in Section 7.

Are neural networks capable of replacing highly specialized numerical algorithms in natural sciences? Shortly after their successes in various data-driven tasks in data science and AI applications, NNs have been used also as a numerical ansatz for solving highly complex models from the natural sciences which may be combined with data driven methods. This per se is not very surprising as many such models can be formulated as optimization problems where the common deep learning paradigm can be directly applied. What might be considered surprising is that this approach seems to be applicable to a wide range of problems which have previously been tackled by highly specialized numerical methods.

Particular successes include the data-driven solution of ill-posed *inverse problems* [AMÖS19] which have, for example, led to a fourfold speedup in MRI scantimes [ZKS⁺18] igniting the research project fastmri.org. Deep-learning-based approaches have also been very successful in solving a vast array of different *partial differential equation* (PDE) models, especially in the high-dimensional regime [EY18, RPK19, HSN20, PSMF20] where most other methods would suffer from the curse of dimensionality.

Despite these encouraging applications, the foundational mechanisms governing their workings and limitations are still not well understood. In Subsection 4.3 and Section 8 we discuss some theoretical and practical aspects of deep learning methods applied to the solution of inverse problems and PDEs.

2 Generalization of large neural networks

In the following, we will shed light on the generalization puzzle of NNs as described in Subsection 1.3. We focus on four different lines of research which, of course, do not cover the wide range of available results. In fact, we had to omit a discussion of a multitude of important works, some of which we reference in the following paragraph.

First, let us mention extensions of the generalization bounds presented in Subsection 1.2.3 making use of *local* Rademacher complexities [BBM05] or dropping assumptions on boundedness or rapidly decaying tails [Men14]. Furthermore, there are approaches to generalization which do not focus on the hypothesis set \mathcal{F} , i.e., the range of the learning algorithm \mathcal{A} , but the way \mathcal{A} chooses its model f_s . For instance, one can assume that f_s does not depend too strongly on each individual sample (*algorithmic stability* [BE02, PRMN04]), only on a subset of the samples (*compression bounds* [AGNZ18]), or satisfies local properties (*algorithmic robustness* [XM12]). Finally, we refer the reader to [JNM⁺20] and the references mentioned therein for an empirical study of various measures related to generalization.

Note that many results on generalization capabilities of NNs can still only be proven in simplified settings, e.g., for deep linear NNs, i.e., $\varrho(x) = x$, or basic linear models, i.e., one-layer NNs. Thus, we start by emphasizing the connection of deep, nonlinear NNs to linear models (operating on features given by a suitable kernel) in the *infinite width limit*.

2.1 Kernel regime

We consider a one-dimensional prediction setting where the loss $\mathcal{L}(f, (x, y))$ depends on $x \in \mathcal{X}$ only through $f(x) \in \mathcal{Y}$, i.e., there exists a function $\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ such that

$$\mathcal{L}(f, (x, y)) = \ell(f(x), y).$$

For instance, in case of the quadratic loss we have that $\ell(\hat{y}, y) = (\hat{y} - y)^2$. Further, let Φ be a NN with architecture $(N, \varrho) = ((d, N_1, \dots, N_{L-1}, 1), \varrho)$ and let Θ_0 be a $\mathbb{R}^{P(N)}$ -valued random variable. For simplicity, we evolve the parameters of Φ according to the continuous version of gradient descent, so-called *gradient flow*, given by

$$\frac{d\Theta(t)}{dt} = -\nabla_{\theta} \widehat{\mathcal{R}}_s(\Phi(\cdot, \Theta(t))) = -\frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \Phi(x^{(i)}, \Theta(t)) D_i(t), \quad \Theta(0) = \Theta_0, \quad (2.1)$$

where $D_i(t) := \frac{\partial \ell(\hat{y}, y^{(i)})}{\partial \hat{y}}|_{\hat{y}=\Phi(x^{(i)}, \Theta(t))}$ is the derivative of the loss with respect to the prediction at input feature $x^{(i)}$ at time $t \in [0, \infty)$. The chain rule implies the following dynamics of the NN realization

$$\frac{d\Phi(\cdot, \Theta(t))}{dt} = -\frac{1}{m} \sum_{i=1}^m K_{\Theta(t)}(\cdot, x^{(i)}) D_i(t) \quad (2.2)$$

and its empirical risk

$$\frac{d\widehat{\mathcal{R}}_s(\Phi(\cdot, \Theta(t)))}{dt} = -\frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m D_i(t) K_{\Theta(t)}(x^{(i)}, x^{(j)}) D_j(t), \quad (2.3)$$

where $K_{\theta}, \theta \in \mathbb{R}^{P(N)}$, is the so-called *neural tangent kernel* (NTK)

$$K_{\theta}: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}, \quad K_{\theta}(x_1, x_2) = (\nabla_{\theta} \Phi(x_1, \theta))^T \nabla_{\theta} \Phi(x_2, \theta). \quad (2.4)$$

Now let $\sigma_w, \sigma_b \in (0, \infty)$ and assume that the initialization Θ_0 consists of independent entries, where entries corresponding to the weight matrix and bias vector in the ℓ -th layer follow a normal distribution with zero mean and variances σ_w^2/N_{ℓ} and σ_b^2 , respectively. Under weak assumptions on the activation function, the central limit theorem implies that the pre-activations converge to i.i.d. centered Gaussian processes in the infinite width limit $N_1, \dots, N_{L-1} \rightarrow \infty$, see [LBN⁺18, MHR⁺18]. Similarly, also K_{Θ_0} converges to a deterministic kernel K^{∞} which stays constant in time and only depends on the activation function ϱ , the depth L , and the initialization parameters σ_w and σ_b [JGH18, ADH⁺19, Yan19, LXS⁺20]. Thus, within the infinite width limit, gradient flow on the NN parameters as in (2.1) is equivalent to functional gradient flow in the *reproducing kernel Hilbert space* $(\mathcal{H}_{K^{\infty}}, \|\cdot\|_{K^{\infty}})$ corresponding to K^{∞} , see (2.2).

By (2.3), the empirical risk converges to a global minimum as long as the kernel evaluated at the input features, $\bar{K}^{\infty} := (K^{\infty}(x^{(i)}, x^{(j)}))_{i,j=1}^m \in \mathbb{R}^{m \times m}$, is positive definite (see, e.g., [JGH18, DLL⁺19] for suitable conditions) and the $\ell(\cdot, y^{(i)})$ are convex and lower bounded. For instance, in case of the quadratic loss the solution of (2.2) is then given by

$$\Phi(\cdot, \Theta(t)) = C(t)(y^{(i)})_{i=1}^m + (\Phi(\cdot, \Theta_0) - C(t)(\Phi(x^{(i)}, \Theta_0))_{i=1}^m), \quad (2.5)$$

where $C(t) := ((K^{\infty}(\cdot, x^{(i)}))_{i=1}^m)^T (\bar{K}^{\infty})^{-1} (\mathbf{I}_m - e^{-\frac{2\bar{K}^{\infty} t}{m}})$. As the initial realization $\Phi(\cdot, \Theta_0)$ constitutes a centered Gaussian process, the second term in (2.5) follows a normal distribution with zero mean at each input. In the limit $t \rightarrow \infty$, its variance vanishes on the input features $x^{(i)}$, $i \in [m]$, and the first term converges to the minimum kernel-norm interpolator, i.e., to the solution of

$$\min_{f \in \mathcal{H}_{K^{\infty}}} \|f\|_{K^{\infty}} \quad \text{s.t.} \quad f(x^{(i)}) = y^{(i)}.$$

Therefore, within the infinite width limit, the generalization properties of the NN could be described by the generalization properties of the minimizer in the reproducing kernel Hilbert space corresponding to the kernel K^∞ [BMM18, LR20, LRZ20, GMMM21, Li21].

This so-called *lazy training*, where a NN essentially behaves like a linear model with respect to the nonlinear features $x \mapsto \nabla_\theta \Phi(x, \theta)$, can already be observed in the non-asymptotic regime, see also Subsection 5.2. For sufficiently overparametrized ($P(N) \gg m$) and suitably initialized models, one can show that $K_{\theta(0)}$ is close to K^∞ at initialization and $K_{\theta(t)}$ stays close to $K_{\theta(0)}$ throughout training, see [DZPS18, ADH⁺19, COB19, DLL⁺19]. The dynamics of the NN under gradient flow in (2.2) and (2.3) can thus be approximated by the dynamics of the linearization of Φ at initialization Θ_0 , given by

$$\Phi^{\text{lin}}(\cdot, \theta) := \Phi(\cdot, \Theta_0) + \langle \nabla_\theta \Phi(\cdot, \Theta_0), \theta - \Theta_0 \rangle,$$

which motivates to study the behavior of linear models in the overparametrized regime.

2.2 Norm-based bounds and margin theory

For piecewise linear activation functions, one can improve upon the VC-dimension bounds in Theorem 1.18 and show that, up to logarithmic factors, the VC-dimension is asymptotically bounded both above and below by $P(N)L$, see [BHL19]. The lower bound shows that the generalization bound in Theorem 1.19 can only be non-vacuous if the number of samples m scales at least linearly with the number of NN parameters $P(N)$. However, heavily overparametrized NNs used in practice seem to generalize well outside of this regime.

One solution is to bound other complexity measures of NNs taking into account various norms on the parameters and avoid the direct dependence on the number of parameters [Bar98]. For instance, we can compute bounds on the Rademacher complexity of NNs with positively homogeneous activation function, where the Frobenius norm of the weight matrices is bounded, see also [NTS15]. Note that, for instance, the ReLU activation is positively homogeneous, i.e., it satisfies that $\varrho_R(\lambda x) = \lambda \varrho_R(x)$ for all $x \in \mathbb{R}$ and $\lambda \in (0, \infty)$.

Theorem 2.1 (Rademacher complexity of neural networks). *Let $d \in \mathbb{N}$, assume that $\mathcal{X} = B_1(0) \subset \mathbb{R}^d$, and let ϱ be a positively homogeneous activation function with Lipschitz constant 1. We define the set of all biasless NN realizations with depth $L \in \mathbb{N}$, output dimension 1, and Frobenius norm of the weight matrices bounded by $C \in (0, \infty)$ as*

$$\tilde{\mathcal{F}}_{L,C} := \{ \Phi_{(N,\varrho)}(\cdot, \theta) : N \in \mathbb{N}^{L+1}, N_0 = d, N_L = 1, \theta = ((W^{(\ell)}, 0))_{\ell=1}^L \in \mathbb{R}^{P(N)}, \|W^{(\ell)}\|_F \leq C \}.$$

Then for every $m \in \mathbb{N}$ it holds that

$$\mathfrak{R}_m(\tilde{\mathcal{F}}_{L,C}) \leq \frac{C(2C)^{L-1}}{\sqrt{m}}.$$

The term 2^{L-1} depending exponentially on the depth can be reduced to \sqrt{L} or completely omitted by invoking also the spectral norm of the weight matrices [GRS18]. Further, observe that for $L = 1$, i.e., linear classifiers with bounded Euclidean norm, this bound is independent of the input dimension d . Together with (1.12), this motivates why the regularized linear model in Figure 1.4 did perform well in the overparametrized regime.

The proof of Theorem 2.1 is based on the contraction property of the Rademacher complexity [LT91] which establishes that

$$\mathfrak{R}_m(\varrho \circ \tilde{\mathcal{F}}_{\ell,C}) \leq 2\mathfrak{R}_m(\tilde{\mathcal{F}}_{\ell,C}), \quad \ell \in \mathbb{N}.$$

We can iterate this together with the fact that for every $\tau \in \{-1, 1\}^m$, and $x \in \mathbb{R}^{N_{\ell-1}}$ it holds that

$$\sup_{\|W^{(\ell)}\|_F \leq C} \left\| \sum_{i=1}^m \tau_i \varrho(W^{(\ell)} x) \right\|_2 = C \sup_{\|w\|_2 \leq 1} \left| \sum_{i=1}^m \tau_i \varrho(\langle w, x \rangle) \right|.$$

In summary, one establishes that

$$\mathfrak{R}_m(\tilde{\mathcal{F}}_{L,C}) = \frac{C}{m} \mathbb{E} \left[\sup_{f \in \tilde{\mathcal{F}}_{L-1,C}} \left\| \sum_{i=1}^m \tau_i \varrho(f(X^{(i)})) \right\|_2 \right] \leq \frac{C(2C)^{L-1}}{m} \mathbb{E} \left[\left\| \sum_{i=1}^m \tau_i X^{(i)} \right\|_2 \right],$$

which by Jensen's inequality yields the claim.

Recall that for classification problems one typically minimizes a surrogate loss \mathcal{L}^{sur} , see Remark 1.9. This suggests that there could be a trade-off happening between complexity of the hypothesis class \mathcal{F}_a and the corresponding regression fit underneath, i.e., the *margin* $M(f, z) := yf(x)$ by which a training example $z = (x, y)$ has been classified correctly by $f \in \mathcal{F}_a$, see [BFT17, NBS18, JKMB19]. For simplicity, let us focus on the ramp surrogate loss with confidence $\gamma > 0$, i.e., $\mathcal{L}_\gamma^{\text{sur}}(f, z) := \ell_\gamma(M(f, z))$, where

$$\ell_\gamma(t) := \mathbb{1}_{(-\infty, \gamma]}(t) - \frac{t}{\gamma} \mathbb{1}_{[0, \gamma]}(t), \quad t \in \mathbb{R}.$$

Note that the ramp function ℓ_γ is $1/\gamma$ -Lipschitz continuous. Using McDiarmid's inequality and a symmetrization argument similar to the proof of Theorem 1.19, combined with the contraction property of the Rademacher complexity, yields the following bound on the probability of misclassification: With probability $1 - \delta$ for every $f \in \mathcal{F}_a$ it holds that

$$\begin{aligned} \mathbb{P}[\text{sgn}(f(X)) \neq Y] &\leq \mathbb{E}[\mathcal{L}_\gamma^{\text{sur}}(f, Z)] \lesssim \frac{1}{m} \sum_{i=1}^m \mathcal{L}_\gamma^{\text{sur}}(f, Z^{(i)}) + \mathfrak{R}_m(\mathcal{L}_\gamma^{\text{sur}} \circ \mathcal{F}_a) + \sqrt{\frac{\ln(1/\delta)}{m}} \\ &\lesssim \frac{1}{m} \sum_{i=1}^m \mathbb{1}_{(-\infty, \gamma)}(Y^{(i)} f(X^{(i)})) + \frac{\mathfrak{R}_m(M \circ \mathcal{F}_a)}{\gamma} + \sqrt{\frac{\ln(1/\delta)}{m}} \\ &= \frac{1}{m} \sum_{i=1}^m \mathbb{1}_{(-\infty, \gamma)}(Y^{(i)} f(X^{(i)})) + \frac{\mathfrak{R}_m(\mathcal{F}_a)}{\gamma} + \sqrt{\frac{\ln(1/\delta)}{m}}. \end{aligned}$$

This shows the trade-off between the complexity of \mathcal{F}_a measured by $\mathfrak{R}_m(\mathcal{F}_a)$ and the fraction of training data that has been classified correctly with a margin of at least γ . In particular this suggests, that (even if we classify the training data correctly with respect to the 0-1 loss) it might be beneficial to further increase the complexity of \mathcal{F}_a to simultaneously increase the margins by which the training data has been classified correctly and thus obtain a better generalization bound.

2.3 Optimization and implicit regularization

The optimization algorithm, which is usually a variant of SGD, seems to play an important role for the generalization performance. Potential indicators for good generalization performance are high speed of convergence [HRS16] or flatness of the local minimum to which SGD converged, which can be characterized by the magnitude of the eigenvalues of the Hessian (or approximately as the robustness of the minimizer to adversarial perturbations on the parameter space), see [KMN⁺17]. In [DR17, NBMS17] generalization bounds depending on a concept of flatness are established by employing a PAC-Bayesian framework, which can be viewed as a generalization of Theorem 1.17, see [McA99]. Further, one can also unite flatness and norm-based bounds by the *Fisher-Rao metric* of information geometry [LPRS19].

Let us motivate the link between generalization and flatness in the case of simple linear models: We assume that our model takes the form $\langle \theta, \cdot \rangle$, $\theta \in \mathbb{R}^d$, and we will use the abbreviations

$$r(\theta) := \widehat{\mathcal{R}}_s(\langle \theta, \cdot \rangle) \quad \text{and} \quad \gamma(\theta) := \min_{i \in [m]} M(\langle \theta, \cdot \rangle, z^{(i)}) = \min_{i \in [m]} y^{(i)} \langle \theta, x^{(i)} \rangle$$

throughout this subsection to denote the empirical risk and the margin for given training data $s = ((x^{(i)}, y^{(i)}))_{i=1}^m$. We assume that we are solving a classification task with the 0-1 loss and that our training

data is linearly separable. This means that there exists a minimizer $\hat{\theta} \in \mathbb{R}^d$ such that $r(\hat{\theta}) = 0$. We observe that δ -robustness in the sense that

$$\max_{\theta \in B_\delta(0)} r(\hat{\theta} + \theta) = r(\hat{\theta}) = 0$$

implies that

$$0 < \min_{i \in [m]} y^{(i)} \langle \hat{\theta} - \delta y^{(i)} \frac{x^{(i)}}{\|x^{(i)}\|_2}, x^{(i)} \rangle \leq \gamma(\hat{\theta}) - \delta \min_{i \in [m]} \|x^{(i)}\|_2,$$

see also [PKL⁺17]. This lower bound on the margin $\gamma(\hat{\theta})$ then ensures generalization guarantees as described in Subsection 2.2.

Even without explicit¹⁹ control on the complexity of \mathcal{F}_a , there do exist results showing that SGD acts as implicit regularization [NTS14]. This is motivated by linear models where SGD converges to the minimal Euclidean norm solution for the quadratic loss and in the direction of the hard margin support vector machine solution for the logistic loss on linearly separable data [SHN⁺18]. Note that convergence to minimum norm or maximum margin solutions in particular decreases the complexity of our hypothesis set and thus improves generalization bounds, see Subsection 2.2.

While we have seen this behavior of gradient descent for linear regression already in the more general context of kernel regression in Subsection 2.1, we want to motivate the corresponding result for classification tasks in the following. We focus on the exponential surrogate loss $\mathcal{L}^{\text{sur}}(f, z) = \ell(M(f, z)) = e^{-yf(x)}$ with $\ell(z) = e^{-z}$, but similar observations can be made for the logistic loss defined in Remark 1.9. We assume that the training data is linearly separable, which guarantees the existence of $\hat{\theta} \neq 0$ with $\gamma(\hat{\theta}) > 0$. Then for every linear model $\langle \theta, \cdot \rangle$, $\theta \in \mathbb{R}^d$, it holds that

$$\langle \hat{\theta}, \nabla_\theta r(\theta) \rangle = \frac{1}{m} \sum_{i=1}^m \underbrace{\ell'(y^{(i)} \langle \theta, x^{(i)} \rangle)}_{<0} \underbrace{y^{(i)} \langle \hat{\theta}, x^{(i)} \rangle}_{>0}.$$

A critical point $\nabla_\theta r(\theta) = 0$ can therefore be approached if and only if for every $i \in [m]$ we have

$$\ell'(y^{(i)} \langle \theta, x^{(i)} \rangle) = -e^{-y^{(i)} \langle \theta, x^{(i)} \rangle} \rightarrow 0,$$

which is equivalent to $\|\theta\|_2 \rightarrow \infty$ and $\gamma(\theta) > 0$. Let us now define

$$r_\beta(\theta) := \frac{\ell^{-1}(r(\beta\theta))}{\beta}, \quad \theta \in \mathbb{R}^d, \quad \beta \in (0, \infty),$$

and observe that

$$r_\beta(\theta) = -\frac{\log(r(\beta\theta))}{\beta} \rightarrow \gamma(\theta), \quad \beta \rightarrow \infty. \quad (2.6)$$

Due to this property, r_β is often referred to as the *smoothed margin* [LL19, JT19b]. We evolve θ according to gradient flow with respect to the smoothed margin r_1 , i.e.,

$$\frac{d\theta(t)}{dt} = \nabla_\theta r_1(\theta(t)) = -\frac{1}{r(\theta(t))} \nabla_\theta r(\theta(t)),$$

which produces the same trajectory as gradient flow with respect to the empirical risk r under a rescaling of the time t . Looking at the evolution of the normalized parameters $\tilde{\theta}(t) = \theta(t)/\|\theta(t)\|_2$, the chain rule establishes that

$$\frac{d\tilde{\theta}(t)}{dt} = P_{\tilde{\theta}(t)} \frac{\nabla_\theta r_{\beta(t)}(\tilde{\theta}(t))}{\beta(t)} \quad \text{with} \quad \beta(t) := \|\theta(t)\|_2 \quad \text{and} \quad P_\theta := \text{Id} - \theta\theta^T, \quad \theta \in \mathbb{R}^d.$$

¹⁹Note that also different architectures can exhibit vastly different inductive biases [ZBH⁺20] and also within the architecture different parameters have different importance, see [FC18, ZBS19] and Proposition 6.2.

This shows that the normalized parameters perform projected gradient ascent with respect to the function $r_{\beta(t)}$, which converges to the margin due to (2.6) and the fact that $\beta(t) = \|\theta(t)\|_2 \rightarrow \infty$ when approaching a critical point. This motivates that during gradient flow the normalized parameters implicitly maximize the margin. See [GLSS18a, GLSS18b, LL19, NLG⁺19, CB20, JT20] for a precise analysis and various extensions, e.g., to homogeneous or two-layer NNs and other optimization geometries.

To illustrate one research direction, we present an exemplary result in the following. Let $\Phi = \Phi_{(N, \varrho)}$ be a biasless NN with parameters $\theta = ((W^{(\ell)}, 0))_{\ell=0}^L$ and output dimension $N_L = 1$. For given input features $x \in \mathbb{R}^{N_0}$, the gradient $\nabla_{W^{(\ell)}} \Phi = \nabla_{W^{(\ell)}} \Phi(x, \theta) \in \mathbb{R}^{N_{\ell-1} \times N_\ell}$ with respect to the weight matrix in the ℓ -th layer satisfies that

$$\nabla_{W^{(\ell)}} \Phi = \varrho(\Phi^{(\ell-1)}) \frac{\partial \Phi}{\partial \Phi^{(\ell+1)}} \frac{\partial \Phi^{(\ell+1)}}{\partial \Phi^{(\ell)}} = \varrho(\Phi^{(\ell-1)}) \frac{\partial \Phi}{\partial \Phi^{(\ell+1)}} W^{(\ell+1)} \text{diag}(\varrho'(\Phi^{(\ell)})),$$

where the pre-activations $(\Phi^{(\ell)})_{\ell=1}^L$ are given as in (1.1). Evolving the parameters according to gradient flow as in (2.1) and using an activation function ϱ with $\varrho(x) = \varrho'(x)x$, such as the ReLU, this implies that

$$\text{diag}(\varrho'(\Phi^{(\ell)})) W^{(\ell)}(t) \left(\frac{dW^{(\ell)}(t)}{dt} \right)^T = \left(\frac{dW^{(\ell+1)}(t)}{dt} \right)^T W^{(\ell+1)}(t) \text{diag}(\varrho'(\Phi^{(\ell)})). \quad (2.7)$$

Note that this ensures the conservation of balancedness between weight matrices of adjacent layers, i.e.,

$$\frac{d}{dt} (\|W^{(\ell+1)}(t)\|_F^2 - \|W^{(\ell)}(t)\|_F^2) = 0,$$

see [DHL18]. Furthermore, for deep linear NNs, i.e., $\varrho(x) = x$, the property in (2.7) implies conservation of alignment of left and right singular spaces of $W^{(\ell)}$ and $W^{(\ell+1)}$. This can then be used to show implicit preconditioning and convergence of gradient descent [ACH18, ACGH19] and that, under additional assumptions, gradient descent converges to a linear predictor that is aligned with the maximum margin solution [JT19a].

2.4 Limits of classical theory and double descent

There is ample evidence that classical tools from statistical learning theory alone, such as Rademacher averages, uniform convergence, or algorithmic stability may be unable to explain the full generalization capabilities of NNs [ZBH⁺17, NK19]. It is especially hard to reconcile the classical bias-variance trade-off with the observation of good generalization performance when achieving zero empirical risk on noisy data using a regression loss. On top of that, this behavior of overparametrized models in the interpolation regime turns out not to be unique to NNs. Empirically, one observes for various methods (decision trees, random features, linear models) that the test error decreases even below the sweet-spot in the u-shaped bias-variance curve when further increasing the number of parameters [BHMM19, GJS⁺20, NKB⁺20]. This is often referred to as the *double descent curve* or *benign overfitting*, see Figure 2.1. For special cases, e.g., linear regression or random feature regression, such behavior can even be proven, see [HMRT19, MM19, BLT20, BHX20, MVSS20].

In the following we analyze this phenomenon in the context of linear regression. Specifically, we focus on a prediction task with quadratic loss, input features given by a centered \mathbb{R}^d -valued random variable X , and labels given by $Y = \langle \theta^*, X \rangle + \nu$, where $\theta^* \in \mathbb{R}^d$ and ν is a centered random variable independent of X . For training data $S = ((X^{(i)}, Y^{(i)}))_{i=1}^m$, we consider the empirical risk minimizer $\hat{f}_S = \langle \hat{\theta}, \cdot \rangle$ with minimum Euclidean norm of its parameters $\hat{\theta}$ or, equivalently, the limit of gradient flow with zero initialization. Using (1.3) and a bias-variance decomposition we can write

$$\begin{aligned} \mathbb{E}[\mathcal{R}(\hat{f}_S) | (X^{(i)})_{i=1}^m] - \mathcal{R}^* &= \mathbb{E}[\|\hat{f}_S - f^*\|_{L^2(\mathbb{P}_X)} | (X^{(i)})_{i=1}^m] \\ &= (\theta^*)^T P \mathbb{E}[X X^T] P \theta^* + \mathbb{E}[\nu^2] \text{Tr}(\Sigma^+ \mathbb{E}[X X^T]), \end{aligned}$$

where $\Sigma := \sum_{i=1}^m X^{(i)} (X^{(i)})^T$, Σ^+ denotes the Moore–Penrose inverse of Σ , and $P := I_d - \Sigma^+ \Sigma$ is the orthogonal projector onto the kernel of Σ . For simplicity, we focus on the variance $\text{Tr}(\Sigma^+ \mathbb{E}[X X^T])$, which can

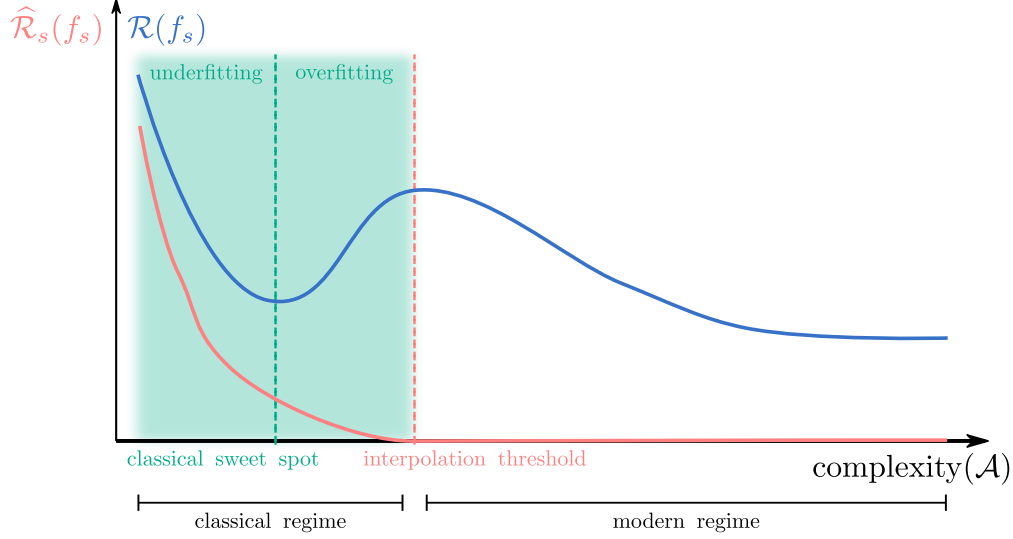


Figure 2.1: This illustration shows the classical, underparametrized regime in green, where the u-shaped curve depicts the bias-variance trade-off as explained in Section 1.2. Starting with complexity of our algorithm \mathcal{A} larger than the interpolation threshold we can achieve zero empirical risk $\hat{\mathcal{R}}_s(f_s)$ (training error), where $f_s = \mathcal{A}(s)$. Within this modern interpolation regime, the risk $\mathcal{R}(f_s)$ (test error) might be even lower than at the classical sweet spot. Whereas complexity(\mathcal{A}) traditionally refers to the complexity of the hypothesis set \mathcal{F} , there is evidence that also the optimization scheme and the data is influencing the complexity leading to definitions like $\text{complexity}(\mathcal{A}) := \max \{m \in \mathbb{N} : \mathbb{E}[\hat{\mathcal{R}}_S(\mathcal{A}(S))] \leq \varepsilon \text{ with } S \sim \mathbb{P}_Z^m\}$, for suitable $\varepsilon > 0$ [NKB⁺20]. This illustration is based on [BHMM19].

be viewed as setting $\theta^* = 0$ and $\mathbb{E}[\nu^2] = 1$. Assuming that X has i.i.d. entries with unit variance and bounded fifth moment, the distribution of the eigenvalues of $\frac{1}{m}\Sigma^+$ in the limit $d, m \rightarrow \infty$ with $\frac{d}{m} \rightarrow \kappa \in (0, \infty)$ can be described via the Marchenko–Pastur law. Therefore, the asymptotic variance can be computed explicitly as

$$\text{Tr}(\Sigma^+ \mathbb{E}[XX^T]) \rightarrow \frac{1 - \max\{1 - \kappa, 0\}}{|1 - \kappa|} \quad \text{for } d, m \rightarrow \infty \quad \text{with } \frac{d}{m} \rightarrow \kappa,$$

almost surely, see [HMRT19]. This shows that despite interpolating the data we can decrease the risk in the overparametrized regime $\kappa > 1$. In the limit $d, m \rightarrow \infty$, such benign overfitting can also be shown for more general settings (including lazy training of NNs), some of which even achieve their optimal risk in the overparametrized regime [MM19, MZ20].

For normally distributed input features X such that $\mathbb{E}[XX^T]$ has rank larger than m , one can also compute the behavior of the variance in the non-asymptotic regime [BLLT20]. Define

$$k^* := \min\{k \geq 0 : \frac{\sum_{i>k} \lambda_i}{\lambda_{k+1}} \geq cm\},$$

where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0$ are the eigenvalues of $\mathbb{E}[XX^T]$ in decreasing order and $c \in (0, \infty)$ is a universal constant. Assuming that k^*/m is sufficiently small, with high probability it holds that

$$\text{Tr}(\Sigma^+ \mathbb{E}[XX^T]) \approx \frac{k^*}{m} + \frac{m \sum_{i>k^*} \lambda_i^2}{(\sum_{i>k^*} \lambda_i)^2}.$$

This precisely characterizes the regimes for benign overfitting in terms of the eigenvalues of the covariance matrix $\mathbb{E}[XX^T]$. Furthermore, it shows that adding new input feature coordinates and thus increasing the number of parameters d can lead to either an increase or decrease of the risk.

To motivate this phenomenon, which is considered in much more depth in [CMBK20], let us focus on a single sample $m = 1$ and features X that take values in $\mathcal{X} = \{-1, 1\}^d$. Then it holds that $\Sigma^+ = \frac{X^{(1)}(X^{(1)})^T}{\|X^{(1)}\|^4} = \frac{X^{(1)}(X^{(1)})^T}{d^2}$ and thus

$$\mathbb{E}[\text{Tr}(\Sigma^+ \mathbb{E}[XX^T])] = \frac{1}{d^2} \|\mathbb{E}[XX^T]\|_F^2. \quad (2.8)$$

In particular, this shows that incrementing the input feature dimensions $d \mapsto d + 1$ one can increase or decrease the risk depending on the correlation of the coordinate X_{d+1} with respect to the previous coordinates $(X_i)_{i=1}^d$, see also Figure 2.2.

Generally speaking, overparametrization and perfectly fitting noisy data does not exclude good generalization performance, see also [BRT19]. However, the risk crucially depends on the data distribution and the chosen algorithm.

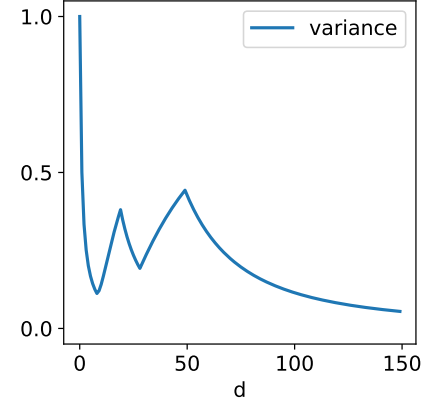


Figure 2.2: The expected variance of linear regression in (2.8) with $d \in [150]$ and $X_i \sim U(\{-1, 1\})$, $i \in [150]$, where $X_i = X_1$ for $i \in \{10, \dots, 20\} \cup \{30, \dots, 50\}$ and all other coordinates are independent.

3 The role of depth in the expressivity of neural networks

The approximation theoretical aspect of a NN architecture, responsible for the approximation component $\varepsilon^{\text{approx}} := \mathcal{R}(f_{\mathcal{F}}^*) - \mathcal{R}^*$ of the error $\mathcal{R}(f_S) - \mathcal{R}^*$ in (1.4), is probably one of the most well-studied parts of the deep learning pipe-line. The achievable approximation error of an architecture most directly describes the power of the architecture.

As mentioned in Subsection 1.3, many classical approaches only study the approximation theory of NNs with few layers, whereas modern architectures are typically very deep. A first observation into the effect of depth is that it can often compensate for insufficient width. For example, in the context of the universal approximation theorem, it was shown that very narrow NNs are still universal if instead of increasing the width, the number of layers can be chosen arbitrarily [HS17, Han19, KL20]. However, if the width of a NN falls below a critical number, then the universality will not hold any longer.

Below, we discuss three additional observations that shed light on the effect of depth on the approximation capacities or alternative notions of expressivity of NNs.

3.1 Approximation of radial functions

One technique to study the impact of depth relies on the construction of specific functions which can be well approximated by NNs of a certain depth, but require significantly more parameters when approximated to the same accuracy by NNs of smaller depth. In the following we present one example for this type of approach, which can be found in [ES16].

Theorem 3.1 (Power of depth). *Let $\varrho \in \{\varrho_R, \varrho_\sigma, \mathbb{1}_{(0,\infty)}\}$ be the ReLU, the logistic, or the Heaviside function. Then there exist constants $c, C \in (0, \infty)$ with the following property: For every $d \in \mathbb{N}$ with $d \geq C$ there exist a probability measure μ on \mathbb{R}^d , a three-layer NN architecture $a = (N, \varrho) = ((d, N_1, N_2, 1), \varrho)$ with $\|N\|_\infty \leq Cd^5$, and corresponding parameters $\theta^* \in \mathbb{R}^{P(N)}$ with $\|\theta^*\|_\infty \leq Cd^C$ and $\|\Phi_a(\cdot, \theta^*)\|_{L^\infty(\mathbb{R}^d)} \leq 2$ such that for every $n \leq ce^{cd}$ it holds that*

$$\inf_{\theta \in \mathbb{R}^{P((d,n,1), \varrho)}} \|\Phi_{((d,n,1), \varrho)}(\cdot, \theta) - \Phi_a(\cdot, \theta^*)\|_{L^2(\mu)} \geq c.$$

In fact, the activation function in Theorem 3.1 is only required to satisfy mild conditions and the result holds, for instance, also for more general sigmoidal functions. The proof of Theorem 3.1 is based on the construction of a suitable radial function $g: \mathbb{R}^d \rightarrow \mathbb{R}$, i.e., $g(x) = \tilde{g}(\|x\|_2^2)$ for some $\tilde{g}: [0, \infty) \rightarrow \mathbb{R}$, which can be efficiently approximated by three-layer NNs but approximation by only a two-layer NN requires exponentially large complexity, i.e., the width being exponential in d .

The first observation of [ES16] is that g can typically be well approximated on a bounded domain by a three-layer NN, if \tilde{g} is Lipschitz continuous. Indeed, for the ReLU activation function it is not difficult to show that, emulating a linear interpolation, one can approximate a univariate C -Lipschitz function uniformly on $[0, 1]$ up to precision ε by a two-layer architecture of width $\mathcal{O}(C/\varepsilon)$. The same holds for smooth, non-polynomial activation functions due to Theorem 1.16. This implies that the squared Euclidean norm, as a sum of d univariate functions, i.e., $[0, 1]^d \ni x \mapsto \sum_{i=1}^d x_i^2$, can be approximated up to precision ε by a two-layer architecture of width $\mathcal{O}(d^2/\varepsilon)$. Moreover, this shows that the third layer can efficiently approximate \tilde{g} , establishing approximation of g on a bounded domain up to precision ε using a three-layer architecture with number of parameters polynomial in d/ε .

The second step in [ES16] is to choose g in such a way that the realization of any two-layer neural network $\Phi = \Phi_{((d,n,1),\varrho)}(\cdot, \theta)$ with width n not being exponential in d is on average (with respect to the probability measure μ) a constant distance away from g . Their argument is heavily based on ideas from Fourier analysis and will be outlined below. In this context, let us recall that we denote by \hat{f} the Fourier transform of a suitable function or, more generally, tempered distribution f .

Assuming that the square-root φ of the density function associated with the probability measure μ as well as Φ and g are well-behaved, the Plancherel theorem yields that

$$\|\Phi - g\|_{L^2(\mu)}^2 = \|\Phi\varphi - g\varphi\|_{L^2(\mathbb{R}^d)}^2 = \|\widehat{\Phi\varphi} - \widehat{g\varphi}\|_{L^2(\mathbb{R}^d)}^2. \quad (3.1)$$

Next, the specific structure of two-layer NNs is used, which implies that for every $j \in [n]$ there exists $w_j \in \mathbb{R}^d$ with $\|w_j\|_2 = 1$ and $\varrho_j: \mathbb{R} \rightarrow \mathbb{R}$ (subsuming the activation function ϱ , the norm of w_j , and the remaining parameters corresponding to the j -th neuron in the hidden layer) such that Φ is of the form

$$\Phi = \sum_{j=1}^n \varrho_j(\langle w_j, \cdot \rangle) = \sum_{j=1}^n (\varrho_j \otimes \mathbf{1}_{\mathbb{R}^{d-1}}) \circ R_{w_j}.$$

The second equality follows by viewing the action of the j -th neuron as a tensor product of ϱ_j and the indicator function $\mathbf{1}_{\mathbb{R}^{d-1}}(x) = 1$, $x \in \mathbb{R}^{d-1}$, composed with a d -dimensional rotation $R_{w_j} \in SO(d)$ which maps w_j to the first standard basis vector $e^{(1)} \in \mathbb{R}^d$. Noting that the Fourier transform respects linearity, rotations, and tensor products, we can compute

$$\hat{\Phi} = \sum_{j=1}^n (\hat{\varrho}_j \otimes \delta_{\mathbb{R}^{d-1}}) \circ R_{w_j},$$

where $\delta_{\mathbb{R}^{d-1}}$ denotes the Dirac distribution on \mathbb{R}^{d-1} . In particular, the support of $\hat{\Phi}$ has a particular star-like shape, namely $\bigcup_{j=1}^n \text{span}\{w_j\}$, which are in fact lines passing through the origin.

Now we choose φ to be the inverse Fourier transform of the indicator function of a ball $B_r(0) \subset \mathbb{R}^d$ with $\text{vol}(B_r(0)) = 1$, ensuring that φ^2 is a valid probability density for μ as

$$\mu(\mathbb{R}^d) = \|\varphi^2\|_{L^1(\mathbb{R}^d)} = \|\varphi\|_{L^2(\mathbb{R}^d)}^2 = \|\hat{\varphi}\|_{L^2(\mathbb{R}^d)}^2 = \|\mathbf{1}_{B_r(0)}\|_{L^2(\mathbb{R}^d)}^2 = 1.$$

Using the convolution theorem, this choice of φ yields that

$$\text{supp}(\widehat{\Phi\varphi}) = \text{supp}(\hat{\Phi} * \hat{\varphi}) \subset \bigcup_{j=1}^n (\text{span}\{w_j\} + B_r(0)).$$

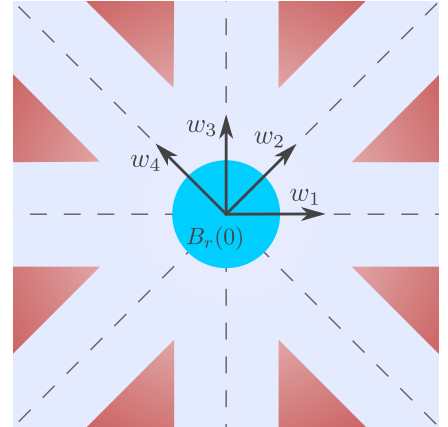


Figure 3.1: This illustration shows the largest possible support (blue) of $\widehat{\Phi\varphi}$, where $\hat{\varphi} = \mathbf{1}_{B_r(0)}$ and Φ is a shallow neural network with architecture $N = (2, 4, 1)$ and weight matrix $W^{(1)} = [w_1 \dots w_4]^T$ in the first layer. Any radial function with enough of its L^2 -mass located at high frequencies (indicated by the red area) cannot be well approximated by $\Phi\varphi$.

Thus the lines passing through the origin are enlarged to tubes. It is this particular shape which allows the construction of some g so that $\|\widehat{\Phi\varphi} - \widehat{g\varphi}\|_{L^2(\mathbb{R}^d)}^2$ can be suitably lower bounded, see also Figure 3.1. Intriguingly, the peculiar behavior of high-dimensional sets now comes into play. Due to the well known concentration of measure principle, the variable n needs to be exponentially large for the set $\bigcup_{j=1}^n (\text{span}\{w_j\} + B_r(0))$ to be not sparse. If it is smaller, one can construct a function \widehat{g} so that the main energy content of $\widehat{g\varphi}$ has a certain distance from the origin, yielding a lower bound for $\|\widehat{\Phi\varphi} - \widehat{g\varphi}\|^2$ and hence $\|\Phi - g\|_{L^2(\mu)}^2$, see (3.1). One key technical problem is the fact that such a behavior for \widehat{g} does not immediately imply a similar behavior of $\widehat{g\varphi}$, requiring a quite delicate construction of g .

3.2 Deep ReLU networks

Maybe for no activation function is the effect of depth clearer than for the ReLU activation function $\varrho_R(x) = \max\{0, x\}$. We refer to corresponding NN architectures (N, ϱ_R) as *ReLU (neural) networks* (ReLU NNs). A two-layer ReLU NN with one-dimensional input and output is a function of the form

$$\Phi(x) = \sum_{i=1}^n w_i^{(2)} \varrho_R(w_i^{(1)} x + b_i^{(1)}) + b^{(2)}, \quad x \in \mathbb{R},$$

where $w_i^{(1)}, w_i^{(2)}, b_i^{(1)}, b^{(2)} \in \mathbb{R}$ for $i \in [n]$. It is not hard to see that Φ is a continuous piecewise affine linear function. Moreover, Φ has at most $n + 1$ affine linear pieces. On the other hand, notice that the *hat function*

$$h: [0, 1] \rightarrow [0, 1],$$

$$x \mapsto 2\varrho_R(x) - 4\varrho_R(x - \tfrac{1}{2}) = \begin{cases} 2x, & \text{if } 0 \leq x < \tfrac{1}{2}, \\ 2(1 - x), & \text{if } \tfrac{1}{2} \leq x \leq 1, \end{cases}$$

is a NN with two layers and two neurons. Telgarsky observed that the n -fold convolution $h_n(x) := h \circ \dots \circ h$ produces a sawtooth function with 2^n spikes [Tel15]. In particular, h_n admits 2^n affine linear pieces with only $2n$ many neurons. In this case, we see that deep ReLU NNs are in some sense exponentially more efficient in generating affine linear pieces.

Moreover, it was noted in [Yar17] that the difference of interpolations of $[0, 1] \ni x \mapsto x - x^2$ at $2^n + 1$ and $2^{n-1} + 1$ equidistant points equals the scaled sawtooth function $\frac{h_n}{2^{2n}}$, see Figure 3.2. This allows to efficiently implement approximative squaring and, by polarization, also approximative multiplication using ReLU NNs. Composing these simple functions one can approximate localized Taylor polynomials and thus smooth functions, see [Yar17]. We state below a generalization [GKP20] of the result of [Yar17] which includes more general norms, but for $p = \infty$ and $s = 0$ coincides with the original result of Dmitry Yarotsky.

Theorem 3.2 (Approximation of Sobolev-regular functions). *Let $d, k \in \mathbb{N}$ with $k \geq 2$, let $p \in [1, \infty]$, $s \in [0, 1]$, $B \in (0, \infty)$, and let ϱ be a piecewise linear activation function with at least one break-point. Then there exists a constant $c \in (0, \infty)$ with the following property: For every $\varepsilon \in (0, 1/2)$ there exists a NN architecture $a = (N, \varrho)$ with*

$$P(N) \leq c\varepsilon^{-d/(k-s)} \log(1/\varepsilon)$$

such that for every function $g \in W^{k,p}((0, 1)^d)$ with $\|g\|_{W^{k,p}((0, 1)^d)} \leq B$ it holds that

$$\inf_{\theta \in \mathbb{R}^{P(N)}} \|\Phi_a(\theta, \cdot) - g\|_{W^{s,p}((0, 1)^d)} \leq \varepsilon.$$

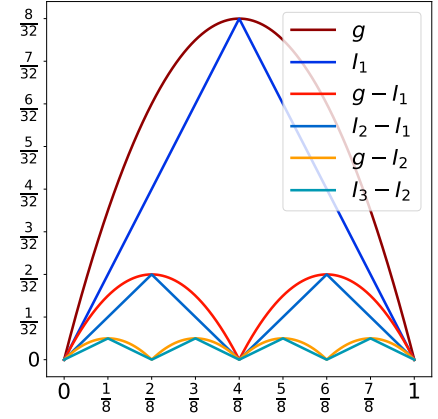


Figure 3.2: Interpolation I_n of $[0, 1] \ni x \mapsto g(x) := x - x^2$ on $2^n + 1$ equidistant points, which can be represented as a sum $I_n = \sum_{k=1}^n I_k - I_{k-1} = \sum_{k=1}^n \frac{h_k}{2^{2k}}$ of n sawtooth functions. Each sawtooth function $h_k = h_{k-1} \circ h$ in turn can be written as a k -fold composition of a hat function h . This illustration is based on [EPGB19].

The ability of deep ReLU neural networks to emulate multiplication has also been employed to reapproximate wide ranges of high-order finite element spaces. In [OPS20] and [MOPS20] it was shown that deep ReLU neural networks are capable of achieving the approximation rates of hp -finite element methods. Concretely, this means that for piecewise analytic functions, which appear, for example, as solutions of elliptic boundary and eigenvalue problems with analytic data, exponential approximation rates can be achieved. In other words, the number of parameters of neural networks to approximate such a function in the $W^{1,2}$ -norm up to an error of ε is logarithmic in ε .

Theorem 3.2 requires the depth of the NN to grow. In fact, it can be shown that the same approximation rate cannot be achieved with shallow NNs. Indeed, there exists a certain optimal number of layers and, if the architecture has fewer layers than optimal, then the NNs need to have significantly more parameters, to achieve the same approximation fidelity. This has been observed in many different settings in [LS17, SS17, Yar17, PV18, EPGB19]. We state here the result of [Yar17]:

Theorem 3.3 (Depth-width approximation trade-off). *Let $d, L \in \mathbb{N}$ with $L \geq 2$ and let $g \in C^2([0, 1]^d)$ be a function which is not affine linear. Then there exists a constant $c \in (0, \infty)$ with the following property: For every $\varepsilon \in (0, 1)$ and every ReLU NN architecture $a = (N, \varrho_R) = ((d, N_1, \dots, N_{L-1}, 1), \varrho_R)$ with L layers and $\|N\|_1 \leq c\varepsilon^{-1/(2(L-1))}$ neurons it holds that*

$$\inf_{\theta \in \mathbb{R}^{P(N)}} \|\Phi_a(\cdot, \theta) - g\|_{L^\infty([0, 1]^d)} \geq \varepsilon.$$

This result is based on the observation that ReLU NNs are piecewise affine linear. The number of pieces they admit is linked to their capacity of approximating functions that have non-vanishing curvature. Using a construction similar to the example at the beginning of this subsection, it can be shown that the number of pieces that can be generated using an architecture $((1, N_1, \dots, N_{L-1}, 1), \varrho_R)$ scales roughly like $\prod_{\ell=1}^{L-1} N_\ell$.

In the framework of the aforementioned results, we can speak of a depth-width trade-off, see also Figure 3.3. A fine-grained estimate of achievable rates for freely varying depths has also been established in [She20].

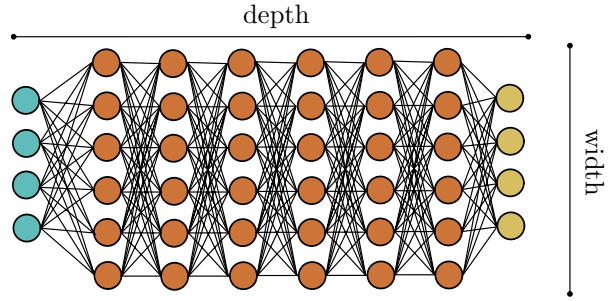


Figure 3.3: Standard feed-forward neural network. For certain approximation results, depth and width need to be in a fixed relationship to achieve optimal results.

3.3 Alternative notions of expressivity

Conceptual approaches to study the approximation power of deep NNs besides the classical approximation framework usually aim to relate structural properties of the NN to the “richness” of the set of possibly expressed functions. One early result in this direction is [MPCB14] which describes bounds on the number of *affine linear regions* of a ReLU NN $\Phi_{(N, \varrho_R)}(\cdot, \theta)$. In a simplified setting, we have seen estimates on the number of affine linear pieces already at the beginning of Subsection 3.2. Affine linear regions can be defined as the connected components of $\mathbb{R}^{N_0} \setminus H$, where H is the set of non-differentiability of the realization²⁰ $\Phi_{(N, \varrho_R)}(\cdot, \theta)$. A refined analysis on the number of such regions was, for example, conducted by [HvdG19]. It is found that deep ReLU neural networks can exhibit significantly more regions than their shallow counterparts.

²⁰One can also study the potentially larger set of *activation regions* given by the connected components of $\mathbb{R}^{N_0} \setminus (\cup_{\ell=1}^{L-1} \cup_{i=1}^{N_\ell} H_{i, \ell})$, where

$$H_{i, \ell} := \{x \in \mathbb{R}^{N_0} : \Phi_i^{(\ell)}(x, \theta) = 0\},$$

with $\Phi_i^{(\ell)}$ as in (1.1), is the set of non-differentiability of the activation of the i -th neuron in the ℓ -th layer. In contrast to the linear regions, the activation regions are necessarily convex [RPK⁺17, HR19].

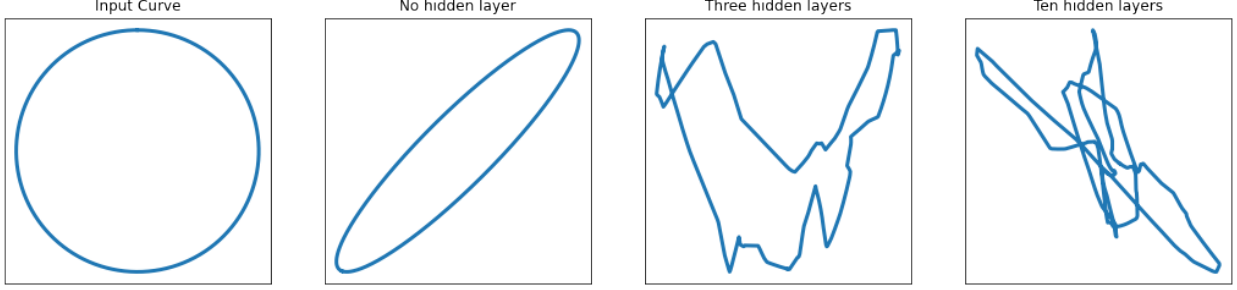


Figure 3.4: Shape of the trajectory $t \mapsto \Phi_{((2,n,\dots,n,2),\varrho_R)}(\gamma(t), \theta)$ of the output of a randomly initialized network with 0, 3, 10 hidden layers. The input curve γ is the circle given in the leftmost image. The hidden layers have $n = 20$ neurons and the variance of the initialization is taken as $4/n$.

The reason for this effectiveness of depth is described by the following analogy: Through the ReLU each neuron $\mathbb{R}^d \ni x \mapsto \varrho_R(\langle x, w \rangle + b)$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, splits the space into two affine linear regions separated by the hyperplane

$$\{x \in \mathbb{R}^d: \langle x, w \rangle + b = 0\}.$$

A shallow ReLU NN $\Phi_{((d,n,1),\varrho_R)}(\cdot, \theta)$ with n neurons in the hidden layer therefore produces a number of regions defined through n hyperplanes. Using classical bounds on the number of regions defined through hyperplane arrangements [Zas75], one can bound the number of affine linear regions by $\sum_{j=0}^d \binom{n}{j}$. Deepening neural networks then corresponds to a certain folding of the input space. Through this interpretation it can be seen that composing NNs can lead to a multiplication of the number of regions of the individual NNs resulting in an exponential efficiency of deep neural networks in generating affine linear regions²¹.

This approach was further developed in [RPK⁺17] to a framework to study expressivity that to some extent allows to include the training phase. One central object studied in [RPK⁺17] are so-called *trajectory lengths*. In this context, one analyzes how the length of a non-constant curve in the input space changes in expectation through the layers of a NN. The authors find an exponential dependence of the expected curve length on the depth. Let us motivate this in the special case of a ReLU NN with architecture $a = ((N_0, n, \dots, n, N_L), \varrho_R)$ and depth $L \in \mathbb{N}$.

Given a non-constant continuous curve $\gamma: [0, 1] \rightarrow \mathbb{R}^{N_0}$ in the input space, the length of the trajectory in the ℓ -th layer of the NN $\Phi_a(\cdot, \theta)$ is then given by

$$\text{Length}(\bar{\Phi}^{(\ell)}(\gamma(\cdot), \theta)), \quad \ell \in [L - 1],$$

where $\bar{\Phi}^{(\ell)}(\cdot, \theta)$ is the activation in the ℓ -th layer, see (1.1). Here the length of the curve is well-defined since $\bar{\Phi}^{(\ell)}(\cdot, \theta)$ is continuous and therefore $\bar{\Phi}^{(\ell)}(\gamma(\cdot), \theta)$ is continuous. Now, let the parameters Θ_1 of the NN Φ_a be initialized independently so that the entries corresponding to the weight matrices and bias vectors follow a normal distribution with zero mean and variances $1/n$ and 1 , respectively. It is not hard to see, e.g., by Proposition 1.1, that the probability that $\bar{\Phi}^{(\ell)}(\cdot, \Theta_1)$ will map γ to a non-constant curve is positive and hence, for fixed $\ell \in [L - 1]$,

$$\mathbb{E}[\text{Length}(\bar{\Phi}^{(\ell)}(\gamma(\cdot), \Theta_1))] = c > 0.$$

Let $\sigma \in (0, \infty)$ and consider a second initialization Θ_σ , where we change the variances of the entries corresponding to the weight matrices and bias vectors to σ^2/n and σ^2 , respectively. Recall that the ReLU is positively homogeneous, i.e., we have that $\varrho_R(\lambda x) = \lambda \varrho_R(x)$ for all $\lambda \in (0, \infty)$. Then it is clear that

$$\bar{\Phi}^{(\ell)}(\cdot, \Theta_\sigma) \sim \sigma^\ell \bar{\Phi}^{(\ell)}(\cdot, \Theta_1),$$

²¹However, to exploit this efficiency with respect to the depth, one requires highly oscillating pre-activations which in turn can only be achieved with a delicate selection of parameters. In fact, it can be shown that through random initialization the expected number of activation regions per unit cube depends mainly on the number of neurons in the NN, rather than its depth [HR19].

i.e., the activations corresponding to the two initialization strategies are identically distributed up to the factor σ^ℓ . Therefore, we immediately conclude that

$$\mathbb{E}[\text{Length}(\bar{\Phi}^{(\ell)}(\gamma(\cdot), \Theta_\sigma))] = \sigma^\ell c.$$

This shows that the expected trajectory length depends exponentially on the depth of the NN, which is in line with the behavior of other notions of expressivity [PLR⁺16]. In [RPK⁺17] this result is also extended to the tanh activation function and the constant c is more carefully resolved. Empirically one also finds that the shapes of the trajectories become more complex in addition to becoming longer on average, see Figure 3.4.

4 Deep neural networks overcome the curse of dimensionality

In Subsection 1.3, one of the main puzzles of deep learning that we identified was the surprising performance of deep architectures on problems where the input dimensions are very high. This performance cannot be explained in the framework of classical approximation theory, since such results always suffer from the curse of dimensionality [Bel52, DeV98, NW09].

In this section, we present three approaches that offer explanations of this phenomenon. As before, we had to omit certain ideas which have been very influential in the literature to keep the length of this section under control. In particular, an important line of reasoning is that functions to be approximated often have compositional structures which NNs may approximate very well as reviewed in [PMR⁺17]. Note that also a suitable feature descriptor, factoring out invariances, might lead to a significantly reduced effective dimension, see Subsection 7.1.

4.1 Manifold assumption

A first remedy to the high-dimensional curse of dimensionality is what we call the *manifold assumption*. Here it is assumed that we are trying to approximate a function

$$g: \mathbb{R}^d \supset \mathcal{X} \rightarrow \mathbb{R},$$

where d is very large. However, we are not seeking to optimize with respect to the uniform norm or a regular L^p space, but instead consider a measure μ which is supported on a d' -dimensional manifold $\mathcal{M} \subset \mathcal{X}$. Then the error is measured in the $L^p(\mu)$ -norm. Here we consider the case where $d' \ll d$. This setting is appropriate if the data $z = (x, y)$ of a prediction task is generated from a measure supported on $\mathcal{M} \times \mathbb{R}$.

This set-up or generalizations thereof have been fundamental in [CM18, SCC18, CJLZ19, SH19, CK20]. Let us describe an exemplary approach, where we consider locally C^k -regular functions and NNs with ReLU activation functions below:

1. *Describe the regularity of g on the manifold:* Naturally, we need to quantify the regularity of the function g restricted to \mathcal{M} in an adequate way. The typical approach would be to make a definition via local coordinate charts. If we assume that \mathcal{M} is an embedded submanifold of \mathcal{X} , then locally, i.e., in a neighborhood of a point $x \in \mathcal{M}$, the orthogonal projection of \mathcal{M} onto the d' -dimensional tangent space $T_x \mathcal{M}$ is a diffeomorphism. The situation is depicted in Figure 4.1. Assuming \mathcal{M} to be compact, we can choose a finite set of open balls $(U_i)_{i=1}^p$ that cover \mathcal{M} and on which the local projections γ_i onto the respective tangent spaces as described above exists and are diffeomorphisms. Now we can define the regularity of g via classical regularity. In this example, we say that $g \in C^k(\mathcal{M})$ if $g \circ \gamma_i^{-1} \in C^k(\gamma_i(\mathcal{M} \cap U_i))$ for all $i \in [p]$.

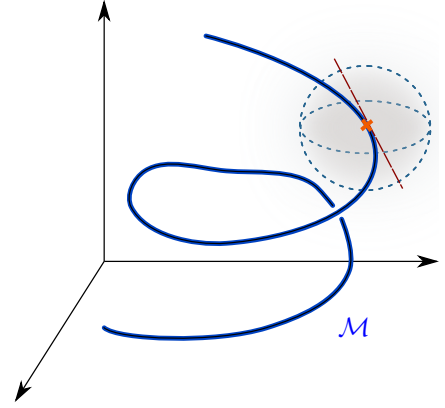


Figure 4.1: Illustration of a one-dimensional manifold \mathcal{M} embedded in \mathbb{R}^3 . For every point $x \in \mathcal{M}$ there exists a neighborhood in which the manifold can be linearly projected onto its tangent space at x such that the corresponding inverse function is differentiable.

2. *Construct localization and charts via neural networks:* According to the construction of local coordinate charts in Step 1, we can write g as follows:

$$g(x) = \sum_{i=1}^p \phi_i(x) (g \circ \gamma_i^{-1}(\gamma_i(x))) =: \sum_{i=1}^p \tilde{g}_i(\gamma_i(x), \phi_i(x)), \quad x \in \mathcal{M}, \quad (4.1)$$

where ϕ_i is a partition of unity such that $\text{supp}(\phi_i) \subset U_i$. Note that γ_i is a linear map, hence representable by a one-layer NN. Since multiplication is a smooth operation, we have that if $g \in C^k(\mathcal{M})$ then $\tilde{g}_i \in C^k(\gamma_i(\mathcal{M} \cap U_i) \times [0, 1])$.

The partition of unity ϕ_i needs to be emulated by NNs. For example, if the activation function is the ReLU, then such a partition can be efficiently constructed. Indeed, in [HLXZ20] it was shown that such NNs can represent linear finite elements exactly with fixed-size NNs and hence a partition of unity subordinate to any given covering of \mathcal{M} can be constructed.

3. *Use a classical approximation result on the localized functions:* By some form of Whitney's extension theorem [Whi34], we can extend each \tilde{g}_i to a function $\bar{g}_i \in C^k(\mathcal{X} \times [0, 1])$ which by classical results can be approximated up to an error of $\varepsilon > 0$ by NNs of size $\mathcal{O}(\varepsilon^{-(d'+1)/k})$ for $\varepsilon \rightarrow 0$, see [Mha96, Yar17, SCC18].
4. *Use the compositionality of neural networks to build the final network:* We have seen that every component in the representation (4.1), i.e., \tilde{g}_i , γ_i , and ϕ_i can be efficiently represented by NNs. In addition, composition and summation are operations which can directly be implemented by NNs through increasing their depth and widening their layers. Hence (4.1) is efficiently—i.e., with a rate depending only on d' instead of the potentially much larger d —approximated by a NN.

Overall, we see that NNs are capable of learning local coordinate transformations and therefore reduce the complexity of a high-dimensional problem to the underlying low-dimensional problem given by the data distribution.

4.2 Random sampling

Already in 1992, Andrew Barron showed that under certain seemingly very natural assumptions on the function to approximate, a dimension-independent approximation rate by NNs can be achieved [Bar92, Bar93]. Specifically, the assumption is formulated as a condition on the Fourier transform of a function and the result is as follows.

Theorem 4.1 (Approximation of Barron-regular functions). *Let $\varrho: \mathbb{R} \rightarrow \mathbb{R}$ be the ReLU or a sigmoidal function. Then there exists a constant $c \in (0, \infty)$ with the following property: For every $d, n \in \mathbb{N}$, every probability measure μ supported on $B_1(0) \subset \mathbb{R}^d$, and every $g \in L^1(\mathbb{R}^d)$ with $C_g := \int_{\mathbb{R}^d} \|\xi\|_2 |\hat{g}(\xi)| d\xi < \infty$ it holds that*

$$\inf_{\theta \in \mathbb{R}^{P((d,n,1),\varrho)}} \|\Phi_{((d,n,1),\varrho)}(\cdot, \theta) - g\|_{L^2(\mu)} \leq \frac{c}{\sqrt{n}} C_g,$$

Note that the L^2 -approximation error can be replaced by an L^∞ -estimate over the unit ball at the expense of a factor of the order of \sqrt{d} on the right-hand side.

The key idea behind Theorem 4.1 is the following application of the law of large numbers: First, we observe that, per assumption, g can be represented via the inverse Fourier transform, as

$$\begin{aligned} g - g(0) &= \int_{\mathbb{R}^d} \hat{g}(\xi) (e^{2\pi i \langle \cdot, \xi \rangle} - 1) d\xi \\ &= C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2} (e^{2\pi i \langle \cdot, \xi \rangle} - 1) \frac{1}{C_g} \|\xi\|_2 \hat{g}(\xi) d\xi \\ &= C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2} (e^{2\pi i \langle \cdot, \xi \rangle} - 1) d\mu_g(\xi), \end{aligned} \quad (4.2)$$

where μ_g is a probability measure. Then it is further shown in [Bar92] that there exist $(\mathbb{R}^d \times \mathbb{R})$ -valued random variables $(\Xi, \tilde{\Xi})$ such that (4.2) can be written as

$$g(x) - g(0) = C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2} (e^{2\pi i \langle x, \xi \rangle} - 1) d\mu_g(\xi) = C_g \mathbb{E}[\Gamma(\Xi, \tilde{\Xi})(x)], \quad x \in \mathbb{R}^d, \quad (4.3)$$

where for every $\xi \in \mathbb{R}^d$, $\tilde{\xi} \in \mathbb{R}$ the function $\Gamma(\xi, \tilde{\xi}): \mathbb{R}^d \rightarrow \mathbb{R}$ is given by

$$\Gamma(\xi, \tilde{\xi}) := s(\xi, \tilde{\xi})(\mathbf{1}_{(0, \infty)}(-\langle \xi / \|\xi\|_2, \cdot \rangle - \tilde{\xi}) - \mathbf{1}_{(0, \infty)}(\langle \xi / \|\xi\|_2, \cdot \rangle - \tilde{\xi})) \quad \text{with} \quad s(\xi, \tilde{\xi}) \in \{-1, 1\}.$$

Now, let $((\Xi^{(i)}, \tilde{\Xi}^{(i)}))_{i \in \mathbb{N}}$ be i.i.d. random variables with $(\Xi^{(1)}, \tilde{\Xi}^{(1)}) \sim (\Xi, \tilde{\Xi})$. Then, Bienaymé's identity and Fubini's theorem establish that

$$\begin{aligned} \mathbb{E} \left[\left\| g - g(0) - \frac{C_g}{n} \sum_{i=1}^n \Gamma(\Xi^{(i)}, \tilde{\Xi}^{(i)}) \right\|_{L^2(\mu)}^2 \right] &= \int_{B_1(0)} \mathbb{V} \left[\frac{C_g}{n} \sum_{i=1}^n \Gamma(\Xi^{(i)}, \tilde{\Xi}^{(i)})(x) \right] d\mu(x) \\ &= \frac{C_g^2 \int_{B_1(0)} \mathbb{V}[\Gamma(\Xi, \tilde{\Xi})(x)] d\mu(x)}{n} \leq \frac{(2\pi C_g)^2}{n}, \end{aligned} \quad (4.4)$$

where the last inequality follows from combining (4.3) with the fact that $|e^{2\pi i \langle x, \xi \rangle} - 1| / \|\xi\|_2 \leq 2\pi$, $x \in B_1(0)$.

This implies that there exists a realization $((\xi^{(i)}, \tilde{\xi}^{(i)}))_{i \in \mathbb{N}}$ of the random variables $((\Xi^{(i)}, \tilde{\Xi}^{(i)}))_{i \in \mathbb{N}}$ that achieves L^2 -approximation error of $n^{-1/2}$. Therefore, it remains to show that NNs can well approximate the functions $(\Gamma(\xi^{(i)}, \tilde{\xi}^{(i)}))_{i \in \mathbb{N}}$. Now it is not hard to see that the function $\mathbf{1}_{(0, \infty)}$ and hence functions of the form $\Gamma(\xi, \tilde{\xi})$, $\xi \in \mathbb{R}^d$, $\tilde{\xi} \in \mathbb{R}$, can be arbitrarily well approximated with a fixed-size, two-layer NN with a sigmoidal or ReLU activation function. Thus, we obtain an approximation rate of $n^{-1/2}$ when approximating functions with one finite Fourier moment by two-layer NNs with n hidden neurons.

It was pointed out already in the dissertation of Emmanuel Candès [Can98] that the approximation rate of NNs for Barron-regular functions is also achievable by n -term approximation with complex exponentials, as is apparent by considering (4.2). However, for deeper NNs, the results also extend to high-dimensional non-smooth functions, where Fourier-based methods are certain to suffer from the curse of dimensionality [CPV20].

In addition, the random sampling idea above was extended in [EMW19b, EMWW20, EW20b, EW20c] to facilitate dimension-independent approximation of vastly more general function spaces. Basically, the idea is to use (4.3) as an inspiration and define the *generalized Barron space* as all functions that may be represented as

$$\mathbb{E}[\mathbf{1}_{(0, \infty)}(\langle \Xi, \cdot \rangle - \tilde{\Xi})]$$

for any random variable $(\Xi, \tilde{\Xi})$. In this context, deep and compositional versions of Barron spaces were introduced and studied in [BK18, EMW19a, EW20a], which considerably extend the original theory.

4.3 PDE assumption

Another structural assumption that leads to the absence of the curse of dimensionality in some cases is that the function we are trying to approximate is given as the solution to a partial differential equation. It is by no means clear that this assumption leads to approximation without the curse of dimensionality, since most standard methods, such as finite elements, sparse grids, or spectral methods typically suffer from the curse of dimensionality.

This is not merely an abstract theoretical problem: Very recently, in [AHNB⁺20] it was shown that two different gold standard methods for solving the multi-electron Schrödinger equation produce completely different interaction energy predictions when applied to large delocalized molecules. Classical numerical representations are simply not expressive enough to accurately represent complicated high-dimensional structures such as wave functions with long-range interactions.

Interestingly, there exists an emerging body of work that shows that NNs do not suffer from these shortcomings and enjoy superior expressivity properties as compared to standard numerical representations.

Such results include, for example, [GHJVW20, GS20, HJKN20] for (linear and semilinear) parabolic evolution equations, [EPGB19] for stationary elliptic PDEs, [GH21] for nonlinear Hamilton–Jacobi–Bellman equations, or [KPRS19] for parametric PDEs. In all these cases, the absence of the curse of dimensionality in terms of the theoretical approximation power of NNs could be rigorously established.

One way to prove such results is via stochastic representations of the PDE solutions, as well as associated sampling methods. We illustrate the idea for the simple case of linear Kolmogorov PDEs, that is the problem of representing the function $g: \mathbb{R}^d \times [0, \infty) \rightarrow \mathbb{R}$ satisfying²²

$$\frac{\partial g}{\partial t}(x, t) = \frac{1}{2} \text{Tr}(\sigma(x, t)[\sigma(x, t)]^* \nabla_x^2 g(x, t)) + \langle \mu(x, t), \nabla_x g(x, t) \rangle, \quad g(x, 0) = \varphi(x), \quad (4.5)$$

where the functions

$$\varphi: \mathbb{R}^d \rightarrow \mathbb{R} \quad (\text{initial condition}) \quad \text{and} \quad \sigma: \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}, \quad \mu: \mathbb{R}^d \rightarrow \mathbb{R}^d \quad (\text{coefficient functions})$$

are continuous and satisfy suitable growth conditions. A stochastic representation of g is given via the Ito processes $(\mathcal{S}_{x,t})_{t \geq 0}$ satisfying

$$d\mathcal{S}_{x,t} = \mu(\mathcal{S}_{x,t})dt + \sigma(\mathcal{S}_{x,t})dB_t, \quad \mathcal{S}_{x,0} = x, \quad (4.6)$$

where $(B_t)_{t \geq 0}$ is a d -dimensional Brownian motion. Then g is described via the Feynman–Kac formula which states that

$$g(x, t) = \mathbb{E}[\varphi(\mathcal{S}_{x,t})], \quad x \in \mathbb{R}^d, \quad t \in [0, \infty). \quad (4.7)$$

Roughly speaking, a NN approximation result can be proven by first approximating, via the law of large numbers,

$$g(x, t) = \mathbb{E}[\varphi(\mathcal{S}_{x,t})] \approx \frac{1}{n} \sum_{i=1}^n \varphi(\mathcal{S}_{x,t}^{(i)}), \quad (4.8)$$

where $(\mathcal{S}_{x,t}^{(i)})_{i=1}^n$ are i.i.d. random variables with $\mathcal{S}_{x,t}^{(1)} \sim \mathcal{S}_{x,t}$. Care has to be taken to establish such an approximation *uniformly in the computational domain*, for example, for every (x, t) in the unit cube $[0, 1]^d \times [0, 1]$, see (4.4) for a similar estimate and [GHJVW20, GS20] for two general approaches to ensure this property. Aside from this issue, (4.8) represents a standard Monte Carlo estimator which can be shown to be free of the curse of dimensionality.

As a next step, one needs to establish that realizations of the processes $(x, t) \mapsto \mathcal{S}_{x,t}$ can be efficiently approximated by NNs. This can be achieved by emulating a suitable time-stepping scheme for the SDE (4.6) by NNs which, roughly speaking, can be done without incurring the curse of dimensionality whenever the coefficient functions μ, σ can be approximated by NNs without incurring the curse of dimensionality and some growth conditions hold true. In a last step one assumes that the initial condition φ can be approximated by NNs without incurring the curse of dimensionality which, by the compositionality of NNs and the previous step, directly implies that realizations of the processes $(x, t) \mapsto \varphi(\mathcal{S}_{x,t})$ can be approximated by NNs without incurring the curse of dimensionality. By (4.8) this implies a corresponding approximation result for the solution of the Kolmogorov PDE g in (4.5).

Informally, we have discovered a regularity result for linear Kolmogorov equations, namely that (modulo some technical conditions on μ, σ), *the solution g of (4.5) can be approximated by NNs without incurring the curse of dimensionality whenever the same holds true for the initial condition φ , as well as the coefficient functions μ, σ . In other words, the property of being approximable by NNs without curse of dimensionality is preserved under the flow induced by the PDE (4.5).* Some comments are in order:

²²The natural solution concept to this type of PDEs is the viscosity solution concept, a thorough study of which can be found in [HHJ15].

Assumption on the initial condition: One may wonder if the assumption that the initial condition φ can be approximated by NNs without incurring the curse of dimensionality is justified. This is at least the case in many applications in computational finance where the function φ typically represents an option pricing formula and (4.5) represents the famous Black–Scholes model. It turns out that nearly all common option pricing formulas are constructed from iterative applications of linear maps and maximum/minimum functions—in other words, in many applications in computational finance, the initial condition φ can be *exactly* represented by a small ReLU NN.

Generalization and optimization error: The Feynman–Kac representation (4.7) directly implies that $g(\cdot, t)$ can be computed as the Bayes optimal function of a regression task with input features $X \sim \mathcal{U}([0, 1]^d)$ and labels $Y = \varphi(\mathcal{S}_{X,t})$, which allows for an analysis of the generalization error as well as implementations based on ERM algorithms [BBG⁺18, BGJ20].

While it is in principle possible to analyze the approximation and generalization error, the analysis of the computational cost and/or convergence of corresponding SGD algorithms is completely open. Some promising numerical results exist, see, for instance, Figure 4.2, but the stable training of NNs approximating PDEs to very high accuracy (that is needed in several applications such as quantum chemistry) remains very challenging. The recent work [GV21] has even proven several impossibility results in that direction.

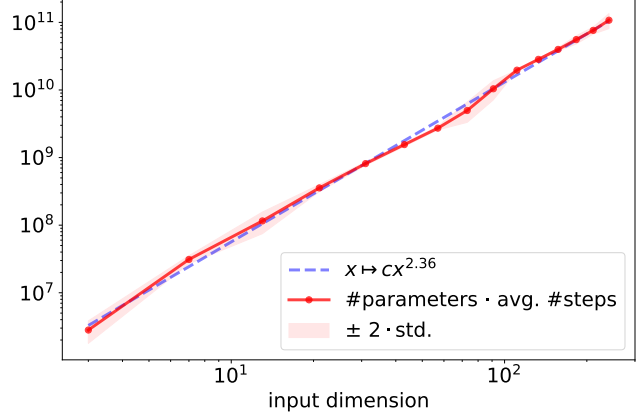


Figure 4.2: Computational complexity as number of neural network parameters times number of SGD steps to solve heat equations of varying dimensions up to a specified precision. According to the fit above, the scaling is polynomial in the dimension [BDG20].

Extensions and abstract idea: Similar techniques may be used to prove expressivity results for nonlinear PDEs, for example, using nonlinear Feynman–Kac-type representations of [PP92] in place of (4.7) and multilevel Picard sampling algorithms of [EHJK19] in place of (4.8).

We can also formulate the underlying idea in an abstract setting (a version of which has also been used in Subsection 4.2). Assume that a high-dimensional function $g: \mathbb{R}^d \rightarrow \mathbb{R}$ admits a probabilistic representation of the form

$$g(x) = \mathbb{E}[Y_x], \quad x \in \mathbb{R}^d, \quad (4.9)$$

for some random variable Y_x which can be approximated by an iterative scheme

$$\mathcal{Y}_x^{(L)} \approx Y_x \quad \text{and} \quad \mathcal{Y}_x^{(\ell)} = T_\ell(\mathcal{Y}_x^{(\ell-1)}), \quad \ell = 1, \dots, L,$$

with dimension-independent convergence rate. If we can approximate realizations of the initial mapping $x \mapsto \mathcal{Y}_x^0$ and the maps T_ℓ , $\ell \in [L]$, by NNs and the numerical scheme is stable enough, then we can also approximate $\mathcal{Y}_x^{(L)}$ using compositionality. Emulating a uniform Monte-Carlo approximator of (4.9) then leads to approximation results for g without curse of dimensionality. In addition, one can choose a \mathbb{R}^d -valued random variable X as input features and define the corresponding labels by Y_X to obtain a prediction task, which can be solved by means of ERM.

Other methods: There exist a number of additional works related to the approximation capacities of NNs for high-dimensional PDEs, for example, [EGJS18, LTY19, SZ19]. In most of these works, the proof technique consists of emulating an existing method that does not suffer from the curse of dimensionality. For instance, in the case of first-order transport equations, one can show in some cases that NNs are capable of emulating the method of characteristics, which then also yields approximation results that are free of the curse of dimensionality [LP21].

5 Optimization of deep neural networks

We recall from Subsections 1.3 and 1.2.1 that the standard algorithm to solve the empirical risk minimization problem over the hypothesis set of NNs is stochastic gradient descent. This method would be guaranteed to converge to a global minimum of the objective if the empirical risk were convex, viewed as a function of the NN parameters. However, this function is severely nonconvex, may exhibit (higher-order) saddle points, seriously suboptimal local minima, and wide flat areas where the gradient is very small.

On the other hand, in applications, excellent performance of SGD is observed. This indicates that the trajectory of the optimization routine somehow misses suboptimal critical points and other areas that may lead to slow convergence. Clearly, the classical theory does not explain this performance. Below we describe some exemplary novel approaches that give partial explanations of this success.

In the flavor of this article, the aim of this section is to present some selected ideas rather than giving an overview of the literature. To give at least some detail about the underlying ideas and to keep the length of this section reasonable, a selection of results had to be made and some ground-breaking results had to be omitted.

5.1 Loss landscape analysis

Given a NN $\Phi(\cdot, \theta)$ and training data $s \in \mathcal{Z}^m$ the function $\theta \mapsto r(\theta) := \widehat{\mathcal{R}}_s(\Phi(\cdot, \theta))$ describes, in a natural way, through its graph, a high-dimensional surface. This surface may have regions associated with lower values of $\widehat{\mathcal{R}}_s$ which resemble valleys of a landscape if they are surrounded by regions of higher values. The analysis of the topography of this surface is called *loss landscape analysis*. Below we shall discuss a couple of approaches that yield deep insights into the shape of this landscape.

Spin glass interpretation: One of the first discoveries about the shape of the loss landscape comes from deep results in statistical physics. The Hamiltonian of the *spin glass model* is a random function on the $(n - 1)$ -dimensional sphere of radius \sqrt{n} . Making certain simplifying assumptions, it was shown in [CHM⁺15] that the loss of a NN with random inputs can be considered as the Hamiltonian of a spin glass model, where the inputs of the model are the parameters of the NN.

This connection has far-reaching implications for the loss landscape of NNs because of the following surprising property of the Hamiltonian of spin glass models: Consider the set of critical points of this set, and associate to each point an *index* that denotes the percentage of the eigenvalues of the Hessian at that point which are negative. This index corresponds to the relative number of directions in which the loss landscape has negative curvature. Then with high probability, a picture like we see in Figure 5.1 emerges [AAC13]. More precisely, the further away from the optimal loss we are, the more unstable the critical points become. Conversely, if one finds oneself in a local minimum, it is reasonable to assume that the loss is close to the global minimum.

While some of the assumptions establishing the connection between the spin glass model and NNs are unrealistic in practice [CLA15], the theoretical distribution of critical points as in Figure 5.1 is visible in many practical applications [DPG⁺14].

Paths and level sets: Another line of research is to understand the loss landscape by analyzing paths through the parameter space. In particular, the existence of paths in parameter space, such that the associated empirical risks are monotone along the path. Surely, should there exist a path of nonincreasing empirical risk from every point to the global minimum, then we can be certain that no non-global minima exist, since no

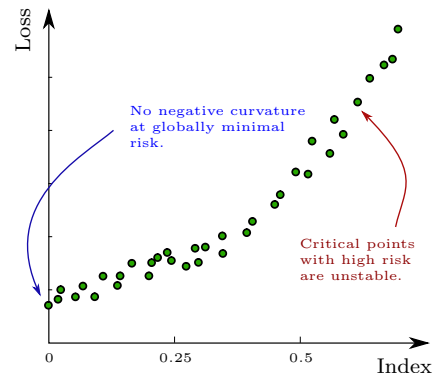


Figure 5.1: Sketch of the distribution of critical points of the Hamiltonian of a spin glass model.

such path can escape a minimum. An even stronger result holds. In fact, the existence of such paths shows that the loss landscape has connected level sets [FB17, VBB19].

A crucial ingredient of the analysis of such paths are *linear substructures*. Consider a biasless two-layer NN Φ of the form

$$\mathbb{R}^d \ni x \mapsto \Phi(x, \theta) := \sum_{j=1}^n \theta_j^{(2)} \varrho(\langle \theta_j^{(1)}, \begin{bmatrix} x \\ 1 \end{bmatrix} \rangle), \quad (5.1)$$

where $\theta_j^{(1)} \in \mathbb{R}^{d+1}$ for $j \in [n]$, $\theta^{(2)} \in \mathbb{R}^n$, ϱ is a Lipschitz continuous activation function, and we augment the vector x by a constant 1 in the last coordinate as outlined in Remark 1.5. If we consider $\theta^{(1)}$ to be fixed, then it is clear that the space

$$\tilde{\mathcal{F}}_{\theta^{(1)}} := \{\Phi(\cdot, \theta) : \theta = (\theta^{(1)}, \theta^{(2)}), \theta^{(2)} \in \mathbb{R}^n\}$$

is a linear space. If the risk²³ is convex, as is the case for the widely used quadratic or logistic loss, then this implies that $\theta^{(2)} \mapsto r((\theta^{(1)}, \theta^{(2)}))$ is a convex map and hence, for every parameter set $\mathcal{P} \subset \mathbb{R}^n$ this map assumes its maximum on $\partial\mathcal{P}$. Therefore, within the vast parameter space, there are many paths traveling along which does not increase the risk above the risk of the start and end points.

This idea was, for example, used in [FB17] in a way similar to the following simple sketch: Assume that, for two parameters θ and θ_{\min} there exists a linear subspace of NNs $\tilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$ such that there are paths γ_1 and γ_2 connecting $\Phi(\cdot, \theta)$ and $\Phi(\cdot, \theta_{\min})$ to $\tilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$ respectively. Further assume that the paths are such that along γ_1 and γ_2 the risk does not significantly exceed $\max\{r(\theta), r(\theta_{\min})\}$. Figure 5.2 shows a visualization of these paths. In this case, a path from θ to θ_{\min} not significantly exceeding $r(\theta)$ along the way is found by concatenating the paths γ_1 , a path along $\tilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$, and γ_2 . By the previous discussion, we know that only γ_1 and γ_2 determine the extent to which the combined path exceeds $r(\theta)$ along its way. Hence, we need to ask about the existence of $\tilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$ that facilitates the construction of appropriate γ_1 and γ_2 .

To understand why a good choice of $\tilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$, so that the risk along γ_1 and γ_2 will not rise much higher than $r(\theta)$, is likely possible, we set²⁴

$$\hat{\theta}_j^{(1)} := \begin{cases} \theta_j^{(1)} & \text{for } j \in [n/2], \\ (\theta_{\min}^{(1)})_j & \text{for } j \in [n] \setminus [n/2]. \end{cases}$$

In other words, the first half of $\hat{\theta}^{(1)}$ is made from $\theta^{(1)}$ and the second from $\theta_{\min}^{(1)}$. If $\theta_j^{(1)}$, $j \in [N]$, are realizations of random variables distributed uniformly on the d -dimensional unit sphere, then by invoking standard covering bounds of spheres (e.g., [Ver18, Corollary 4.2.13]), we expect that, for $\varepsilon > 0$ and a sufficiently large number of neurons n , the vectors $(\theta_j^{(1)})_{j=1}^{n/2}$ already ε -approximate all vectors $(\theta_j^{(1)})_{j=1}^n$. Replacing all vectors $(\theta_j^{(1)})_{j=1}^n$ by their nearest neighbor in $(\theta_j^{(1)})_{j=1}^{n/2}$ can be done with a linear path in the parameter space, and, given that r is locally Lipschitz continuous and $\|\theta^{(2)}\|_1$ is bounded, this operation will not increase the risk by more than $\mathcal{O}(\varepsilon)$. We denote the vector resulting from this replacement procedure by $\theta_*^{(1)}$. Since for all $j \in [n] \setminus [n/2]$ we now have that

$$\varrho(\langle (\theta_*^{(1)})_j, \begin{bmatrix} \cdot \\ 1 \end{bmatrix} \rangle) \in \left\{ \varrho(\langle (\theta_*^{(1)})_k, \begin{bmatrix} \cdot \\ 1 \end{bmatrix} \rangle) : k \in [n/2] \right\},$$

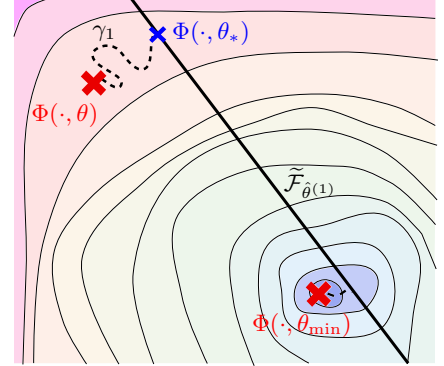


Figure 5.2: Construction of a path from an initial point θ to the global minimum θ_{\min} that does not have significantly higher risk than the initial point along the way. We depict here the landscape as a function of the neural network realizations instead of their parametrizations so that this landscape is convex.

²³As most statements in this subsection are valid for the empirical risk $r(\theta) = \hat{\mathcal{R}}_s(\Phi(\cdot, \theta))$ as well as the risk $r(\theta) = \mathcal{R}(\Phi(\cdot, \theta))$, given a suitable distribution of Z , we will just call r the risk.

²⁴We assume w.l.o.g. that n is a multiple of 2.

there exists a vector $\theta_*^{(2)}$ with $(\theta_*^{(2)})_j = 0$, $j \in [n] \setminus [n/2]$, so that

$$\Phi(\cdot, (\theta_*^{(1)}, \theta^{(2)})) = \Phi(\cdot, (\theta_*^{(1)}, \lambda\theta_*^{(2)} + (1-\lambda)\theta^{(2)})), \quad \lambda \in [0, 1].$$

In particular, this path does not change the risk between $(\theta_*^{(1)}, \theta^{(2)})$ and $(\theta_*^{(1)}, \theta_*^{(2)})$. Now, since $(\theta_*^{(2)})_j = 0$ for $j \in [n] \setminus [n/2]$, the realization $\Phi(\cdot, (\theta_*^{(1)}, \theta_*^{(2)}))$ is computed by a sub-network consisting of the first $n/2$ hidden neurons and we can replace the parameters corresponding to the other neurons without any effect on the realization function. Specifically, it holds that

$$\Phi(\cdot, (\theta_*^{(1)}, \theta_*^{(2)})) = \Phi(\cdot, (\lambda\hat{\theta}^{(1)} + (1-\lambda)\theta_*^{(1)}, \theta_*^{(2)})), \quad \lambda \in [0, 1],$$

yielding a path of constant risk between $(\theta_*^{(1)}, \theta_*^{(2)})$ and $(\hat{\theta}^{(1)}, \theta_*^{(2)})$. Connecting these paths completes the construction of γ_1 and shows that the risk along γ_1 does not exceed that at θ by more than $\mathcal{O}(\varepsilon)$. Of course, γ_2 can be constructed in the same way. The entire construction is depicted in Figure 5.2.

Overall, this derivation shows that for sufficiently wide NNs (appropriately randomly initialized) it is very likely possible to almost connect a random parameter value to the global minimum with a path which along the way does not need to climb much higher than the initial risk.

In [VBB19], a similar approach is taken and the convexity in the last layer is used. However, the authors invoke the concept of intrinsic dimension to elegantly solve the non-linearity of $r((\theta^{(1)}, \theta^{(2)}))$ with respect to $\theta^{(1)}$. Additionally, [SS16] constructs a path of decreasing risk from random initializations. The idea here is that if one starts at a point of sufficiently high risk, one can always find a path to the global optimum with strictly decreasing risk. The intriguing insight behind this result is that if the initialization is sufficiently bad, i.e., worse than that of a NN outputting only zero, then there exist two operations that influence the risk directly. Multiplying the last layer with a number smaller than one will decrease the risk, whereas the opposite will increase it. Using this tuning mechanism, any given potentially non-monotone path from the initialization to the global minimum can be modified so that it is strictly monotonically decreasing. In a similar spirit, [NH17] shows that if a deep NN has a layer with more neurons than training data points, then under certain assumptions the training data will typically be mapped to linearly independent points in that layer. Of course, this layer could then be composed with a linear map that maps the linearly independent points to any desirable output, in particular one that achieves vanishing empirical risk, see also Proposition 1.1. As for two-layer NNs, the previous discussion on linear paths immediately shows that in this situation a monotone path to the global minimum exists.

5.2 Lazy training and provable convergence of stochastic gradient descent

When training highly overparametrized NNs, one often observes that the parameters of the NNs barely change during training. In Figure 5.3, we show the relative distance that the parameters travel through the parameter space during the training of NNs of varying numbers of neurons per layer.

The effect described above has been observed repeatedly and theoretically explained, see, e.g., [DZPS18, LL18, AZLS19, DLL⁺19, ZCZG20]. In Subsection 2.1, we have already seen a high-level overview and, in particular, the function space perspective of this phenomenon in the infinite width limit. Below we present a short and highly simplified derivation of this effect and show how it leads to provable convergence of gradient descent for sufficiently overparametrized deep NNs.

A simple learning model: We consider again the simple NN model of (5.1) with a smooth activation function ϱ which is not affine linear. For the quadratic loss and training data $s = ((x^{(i)}, y^{(i)}))_{i=1}^m \in (\mathbb{R}^d \times \mathbb{R})^m$, where $x_i \neq x_j$ for all $i \neq j$, the empirical risk is given by

$$r(\theta) = \widehat{\mathcal{R}}_s(\theta) = \frac{1}{m} \sum_{i=1}^m (\Phi(x^{(i)}, \theta) - y^{(i)})^2.$$

Let us further assume that $\Theta_j^{(1)} \sim \mathcal{N}(0, 1/n)^{d+1}$, $j \in [n]$, and $\Theta_j^{(2)} \sim \mathcal{N}(0, 1/n)$, $j \in [n]$, are independent random variables.

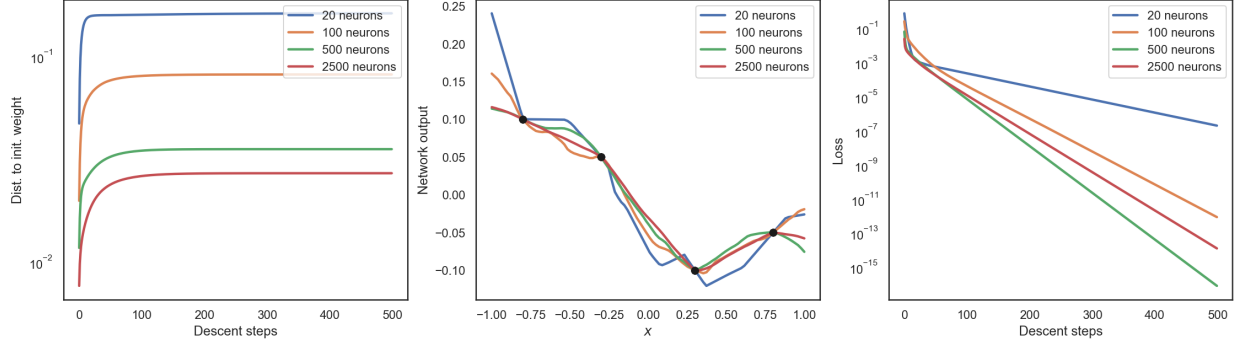


Figure 5.3: Four networks with architecture $((1, n, n, 1), \varrho_R)$ and $n \in \{20, 100, 500, 2500\}$ neurons per hidden layer were trained by gradient descent to fit four points that are shown in the middle figure as black dots. We depict on the left the relative Euclidean distance of the parameters from the initialization through the training process. In the middle, we show the final trained NNs. On the right we show the behavior of the training error.

A peculiar kernel: Next, we would like to understand how the gradient $\nabla_{\theta} r(\Theta)$ looks like with high probability over the initialization $\Theta = (\Theta^{(1)}, \Theta^{(2)})$. Similar to (2.3), we have by restricting the gradient to $\theta^{(2)}$ and applying the chain rule that

$$\begin{aligned} \|\nabla_{\theta} r(\Theta)\|_2^2 &\geq \frac{4}{m^2} \left\| \sum_{i=1}^m \nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta) (\Phi(x^{(i)}, \Theta) - y^{(i)}) \right\|_2^2 \\ &= \frac{4}{m^2} ((\Phi(x^{(i)}, \Theta) - y^{(i)})_{i=1}^m)^T \bar{K}_{\Theta} (\Phi(x^{(j)}, \Theta) - y^{(j)})_{j=1}^m, \end{aligned} \quad (5.2)$$

where \bar{K}_{Θ} is a random $\mathbb{R}^{m \times m}$ -valued kernel given by

$$(\bar{K}_{\Theta})_{i,j} := (\nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta))^T \nabla_{\theta^{(2)}} \Phi(x^{(j)}, \Theta), \quad i, j \in [m].$$

This kernel is closely related to the neural tangent kernel in (2.4) evaluated at the features $(x^{(i)})_{i=1}^m$ and the random initialization Θ . It is a slightly simplified version thereof, as in (2.4) the gradient is taken with respect to the full vector θ . This can also be regarded as the kernel associated with a *random features model* [RR⁺07].

Note that for our two-layer NN we have that

$$(\nabla_{\theta^{(2)}} \Phi(x, \Theta))_k = \varrho \left(\left\langle \Theta_k^{(1)}, \begin{bmatrix} x \\ 1 \end{bmatrix} \right\rangle \right), \quad x \in \mathbb{R}^d, \quad k \in [n].$$

Thus, we can write \bar{K}_{Θ} as the following sum of (random) rank one matrices:

$$\bar{K}_{\Theta} = \sum_{k=1}^n v_k v_k^T \quad \text{with} \quad v_k = \left(\varrho \left(\left\langle \Theta_k^{(1)}, \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix} \right\rangle \right) \right)_{i=1}^m \in \mathbb{R}^m, \quad k \in [n]. \quad (5.3)$$

The kernel \bar{K}_{Θ} are symmetric and positive semi-definite by construction. It is positive definite if it is non-singular, i.e., if at least m of the n vectors v_k , $k \in [n]$, are linearly independent. Proposition 1.1 shows that for $n = m$ the probability of that event is not zero, say δ , and is therefore at least $1 - (1 - \delta)^{\lfloor n/m \rfloor}$ for arbitrary n . In other words, the probability increases rapidly with n . It is also clear from (5.3) that $\mathbb{E}[\bar{K}_{\Theta}]$ scales linearly with n .

From this intuitive derivation, we conclude that for sufficiently large n , with high probability \bar{K}_{Θ} is a positive definite kernel with smallest eigenvalue $\lambda_{\min}(\bar{K}_{\Theta})$ scaling linearly with n . The properties of \bar{K}_{Θ} , in particular its positive definiteness, have been studied much more rigorously as already described in Subsection 2.1.

Control of the gradient: Applying the expected behavior of the smallest eigenvalue $\lambda_{\min}(\bar{K}_\Theta)$ of \bar{K}_Θ to (5.2), we conclude that with high probability

$$\|\nabla_{\theta} r(\Theta)\|_2^2 \geq \frac{4}{m^2} \lambda_{\min}(\bar{K}_\Theta) \|(\Phi(x^{(i)}, \Theta) - y^{(i)})_{i=1}^m\|_2^2 \gtrsim \frac{n}{m} r(\Theta). \quad (5.4)$$

To understand what will happen when applying gradient descent, we first need to understand how the situation changes in a neighborhood of Θ . We fix $x \in \mathbb{R}^d$ and observe that by the mean value theorem for all $\bar{\theta} \in B_1(0)$ we have

$$\|\nabla_{\theta} \Phi(x, \Theta) - \nabla_{\theta} \Phi(x, \Theta + \bar{\theta})\|_2^2 \lesssim \sup_{\hat{\theta} \in B_1(0)} \|\nabla_{\theta}^2 \Phi(x, \Theta + \hat{\theta})\|_{\text{op}}^2, \quad (5.5)$$

where $\|\nabla_{\theta}^2 \Phi(x, \Theta + \hat{\theta})\|_{\text{op}}$ denotes the operator norm of the Hessian of $\Phi(x, \cdot)$ at $\Theta + \hat{\theta}$. From inspection of (5.1), it is not hard to see that for all $i, j \in [n]$ and $k, \ell \in [d+1]$

$$\mathbb{E} \left[\left(\frac{\partial^2 \Phi(x, \Theta)}{\partial \theta_i^{(2)} \partial \theta_j^{(2)}} \right)^2 \right] = 0, \quad \mathbb{E} \left[\left(\frac{\partial^2 \Phi(x, \Theta)}{\partial \theta_i^{(2)} \partial (\theta_j^{(1)})_k} \right)^2 \right] \lesssim \delta_{i,j}, \quad \text{and} \quad \mathbb{E} \left[\left(\frac{\partial^2 \Phi(x, \Theta)}{\partial (\theta_i^{(1)})_k \partial (\theta_j^{(1)})_\ell} \right)^2 \right] \lesssim \frac{\delta_{i,j}}{n},$$

where $\delta_{i,j} = 0$ if $i \neq j$ and $\delta_{i,i} = 1$ for all $i, j \in [n]$. For sufficiently large n , we have that $\nabla_{\theta}^2 \Phi(x, \Theta)$ is in expectation approximately a block band matrix with band-width $d+1$. Therefore, we conclude that $\mathbb{E}[\|\nabla_{\theta}^2 \Phi(x, \Theta)\|_{\text{op}}^2] \lesssim 1$. Hence, we obtain by concentration of Gaussian random variables that with high probability $\|\nabla_{\theta}^2 \Phi(x, \Theta)\|_{\text{op}}^2 \lesssim 1$. By the block-banded form of $\nabla_{\theta}^2 \Phi(x, \Theta)$ we have that, even after perturbation of Θ by a vector $\bar{\theta}$ with norm bounded by 1, the term $\|\nabla_{\theta}^2 \Phi(x, \Theta + \bar{\theta})\|_{\text{op}}^2$ is bounded, which yields that the right-hand side of (5.5) is bounded with high probability.

Using (5.5), we can extend (5.4), which holds with high probability, to a neighborhood of Θ by the following argument: Let $\bar{\theta} \in B_1(0)$, then

$$\begin{aligned} \|\nabla_{\theta} r(\Theta + \bar{\theta})\|_2^2 &\geq \frac{4}{m^2} \left\| \sum_{i=1}^m \nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta + \bar{\theta}) (\Phi(x^{(i)}, \Theta + \bar{\theta}) - y^{(i)}) \right\|_2^2 \\ &\stackrel{(5.5)}{=} \frac{4}{m^2} \left\| \sum_{i=1}^m (\nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta) + \mathcal{O}(1)) (\Phi(x^{(i)}, \Theta + \bar{\theta}) - y^{(i)}) \right\|_2^2 \\ &\gtrsim \frac{1}{m^2} (\lambda_{\min}(\bar{K}_\Theta) + \mathcal{O}(1)) \|(\Phi(x^{(i)}, \Theta + \bar{\theta}) - y^{(i)})_{i=1}^m\|_2^2 \\ &\stackrel{(*)}{\gtrsim} \frac{n}{m} r(\Theta + \bar{\theta}), \end{aligned} \quad (5.6)$$

where the estimate marked by $(*)$ uses the positive definiteness of \bar{K}_Θ again and only holds for sufficiently large n , so that the $\mathcal{O}(1)$ term is negligible.

We conclude that, with high probability over the initialization Θ , on a ball of fixed radius around Θ the squared Euclidean norm of the gradient of the empirical risk is lower bounded by $\frac{n}{m}$ times the empirical risk.

Exponential convergence of gradient descent: For sufficiently small step sizes η , the observation in the previous paragraph yields the following convergence rate for gradient descent as in Algorithm 1, specifically (1.5), with $m' = m$ and $\Theta^{(0)} = \Theta$: If $\|\Theta^{(k)} - \Theta\| \leq 1$ for all $k \in [K+1]$, then²⁵

$$r(\Theta^{(K+1)}) \approx r(\Theta^{(K)}) - \eta \|\nabla_{\theta} r(\Theta^{(K)})\|_2^2 \leq \left(1 - \frac{c\eta n}{m}\right) r(\Theta^{(K)}) \lesssim \left(1 - \frac{c\eta n}{m}\right)^K, \quad (5.7)$$

for $c \in (0, \infty)$ so that $\|\nabla_{\theta} r(\Theta^{(k)})\|_2^2 \geq \frac{cn}{m} r(\Theta^{(k)})$ for all $k \in [K]$.

²⁵Note that the step-size η needs to be small enough to facilitate the approximation step in (5.7). Hence, we cannot simply put $\eta = m/(cn)$ in (5.7) and have convergence after one step.

Let us assume without proof that the estimate (5.6) could be extended to an equivalence. In other words, we assume that we additionally have that $\|\nabla_{\theta} r(\Theta + \bar{\theta})\|_2^2 \lesssim \frac{n}{m} r(\Theta + \bar{\theta})$. This, of course, could be shown with similar tools as were used for the lower bound. Then we have that $\|\Theta^{(k)} - \Theta\|_2 \leq 1$ for all $k \lesssim \sqrt{m/(\eta^2 n)}$. Setting $t = \sqrt{m/(\eta^2 n)}$ and using the limit definition of the exponential function, i.e., $\lim_{t \rightarrow \infty} (1 - x/t)^t = e^{-x}$, yields for sufficiently small η that (5.7) is bounded by $e^{-c\sqrt{n/m}}$.

We conclude that, with high probability over the initialization, *gradient descent converges with an exponential rate to an arbitrary small empirical risk if the width n is sufficiently large. In addition, the iterates of the descent algorithm even stay in a small fixed neighborhood of the initialization during training.* Because the parameters only move very little, this type of training has also been coined lazy training [COB19].

Similar ideas as above, have led to groundbreaking convergence results of SGD for overparametrized NNs in much more complex and general settings, see, e.g., [DZPS18, LL18, AZLS19].

In the infinite width limit, NN training is practically equivalent to kernel regression, see Subsection 2.1. If we look at Figure 5.3 we see that the most overparametrized NN interpolates the data like a kernel-based interpolator would. In a sense, which was also highlighted in [COB19], this shows that, while overparametrized NNs in the lazy training regime have very nice properties, they essentially act like linear methods.

6 Tangible effects of special architectures

In this section, we describe results that isolate the effects of certain aspects of NN architectures. As we have discussed in Subsection 1.3, typically only either the depth or the number of parameters are used to study theoretical aspects of NNs. We have seen instances of this throughout Sections 3 and 4. Moreover, also in Section 5, we saw that wider NNs enjoy certain very favorable properties from an optimization point of view.

Below, we introduce certain specialized NN architectures. We start with one of the most widely used types of NNs, the *convolutional neural network* (CNN). In Subsection 6.2 we introduce *skip connections* and in Subsection 6.3 we discuss a specific class of CNNs equipped with an encoder-decoder structure that are frequently used in image processing techniques. We introduce the *batch normalization block* in Subsection 6.4. Then, we discuss *sparsely connected* NNs that typically result as an extraction from fully connected NNs in Subsection 6.5. Finally, we briefly comment on recurrent neural networks in Subsection 6.6.

As we have noted repeatedly throughout this manuscript, it is impossible to give a full account of the literature in a short introductory article. In this section, this issue is especially severe since the number of special architectures studied in practice is enormous. Therefore, we had to omit many very influential and widely used neural network architectures. Among those are *graph neural networks*, which handle data from non-Euclidean input spaces. We refer to the survey articles [BBL⁺17, WPC⁺21] for a discussion. Another highly successful type of architectures are (*variational*) *autoencoders* [AHS85, HZ94]. These are neural networks with a bottleneck that enforce a more efficient representation of the data. Similarly, *generative adversarial networks* [GPAM⁺14] which are composed of two neural networks, one generator and one discriminator, could not be discussed here. Another widely used component of architectures used in practice is the so-called *dropout layer*. This layer functions through removing some neurons randomly during training. This procedure empirically prevents overfitting. An in-detail discussion of the mathematical analysis behind this effect is beyond the scope of this manuscript. We refer to [WZZ⁺13, SHK⁺14, HV17, MAV18] instead. Finally, the very successful *attention mechanism* [BCB15, VSP⁺17], that is the basis of *transformer neural networks*, had to be omitted.

Before we start describing certain effects of special NN architectures, a word of warning is required. The special building blocks, which will be presented below, have been developed based on a specific need in applications and are used and combined in a very flexible way. To describe these tools theoretically without completely inflating the notational load, some simplifying assumptions need to be made. It is very likely that the simplified building blocks do not accurately reflect the practical applications of these tools in all use cases.

6.1 Convolutional neural networks

Especially for very high-dimensional inputs where the input dimensions are spatially related, fully connected NNs seem to require unnecessarily many parameters. For example, in image classification problems, neighboring pixels very often share information and the spatial proximity should be reflected in the architecture. Based on this observation, it appears reasonable to have NNs that have local receptive fields in the sense that they collect information jointly from spatially close inputs. In addition, in image processing, we are not necessarily interested in a universal hypothesis set. A good classifier is invariant under many operations, such as translation or rotation of images. It seems reasonable to hard-code such invariances into the architecture.

These two principles suggest that the receptive field of a NN should be the same on different translated patches of the input. In this sense, parameters of the architecture can be reused. Together, these arguments make up the three fundamental principles of convolutional NNs: local receptive fields, parameter sharing, and equivariant representations, as introduced in [LBD⁺89]. We will provide a mathematical formulation of convolutional NNs below and then revisit these concepts.

A convolutional NN corresponds to multiple convolutional blocks, which are special types of layers. For a group G , which typically is either $[d] \cong \mathbb{Z}/(d\mathbb{Z})$ or $[d]^2 \cong (\mathbb{Z}/(d\mathbb{Z}))^2$ for $d \in \mathbb{N}$, depending on whether we are performing one-dimensional or two-dimensional convolutions, the convolution of two vectors $a, b \in \mathbb{R}^G$ is defined as

$$(a * b)_i = \sum_{j \in G} a_j b_{j^{-1}i}, \quad i \in G.$$

Now we can define a *convolutional block* as follows: Let \tilde{G} be a subgroup of G , let $p : G \rightarrow \tilde{G}$ be a so-called *pooling-operator*, and let $C \in \mathbb{N}$ denote the number of channels. Then, for a series of kernels $\kappa_i \in \mathbb{R}^G$, $i \in [C]$, the output of a convolutional block is given by

$$\mathbb{R}^G \ni x \mapsto x' := (p(x * \kappa_i))_{i=1}^C \in (\mathbb{R}^{\tilde{G}})^C. \quad (6.1)$$

A typical example of a pooling operator is for $G = (\mathbb{Z}/(2d\mathbb{Z}))^2$ and $\tilde{G} = (\mathbb{Z}/(d\mathbb{Z}))^2$ the 2×2 subsampling operator

$$p : \mathbb{R}^G \rightarrow \mathbb{R}^{\tilde{G}}, \quad x \mapsto (x_{2i-1, 2j-1})_{i,j=1}^d.$$

Popular alternatives are average pooling or max pooling. These operations then either pass the average or the maximum over patches of similar size. The convolutional kernels correspond to the aforementioned receptive fields. They can be thought of as local if they have small supports, i.e., few nonzero entries.

As explained earlier, a convolutional NN is built by stacking multiple convolutional blocks after another²⁶. At some point, the output can be *flattened*, i.e., mapped to a vector and is then fed into a FC NN (see Definition 1.4). We depict this setup in Figure 6.1.

Owing to the fact that convolution is a linear operation, depending on the pooling operation, one may write a convolutional block (6.1) as a FC NN. For example, if $G = (\mathbb{Z}/(2d\mathbb{Z}))^2$ and the 2×2 subsampling pooling operator is used, then the convolutional block could be written as $x \mapsto Wx$ for a block circulant matrix $W \in \mathbb{R}^{(Cd^2) \times (2d)^2}$. Since we require W to have a special structure, we can interpret a convolutional block as a special, restricted feed-forward architecture.

After these considerations, it is natural to ask how the restriction of a NN to a pure convolutional structure, i.e., consisting only of convolutional blocks, will affect the resulting hypothesis set. The first natural question is whether the set of such NNs is still universal in the sense of Theorem 1.15. The answer to this question depends strongly on the type of pooling and convolution that is allowed. If the convolution is performed with padding, then the answer is yes [OS19, Zho20b]. On the other hand, for circular convolutions and without pooling, universality does not hold, but the set of translation equivariant functions can be universally approximated [Yar18b, PV20]. Furthermore, [Yar18b] illuminates the effect of subsample pooling by showing that, if no pooling is applied, then universality cannot be achieved, whereas if pooling is applied

²⁶We assume that the definition of a convolutional block is suitably extended to input data in the Cartesian product $(\mathbb{R}^G)^C$. For instance, one can take an affine linear combination of C mappings as in (6.1) acting on each coordinate. Moreover, one may also interject an activation function between the blocks.

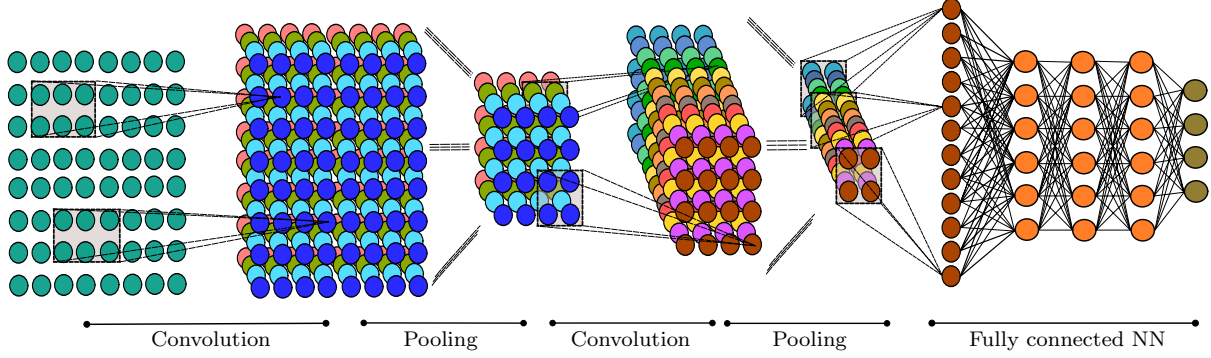


Figure 6.1: Illustration of a convolutional neural network with two-dimensional convolutional blocks and 2×2 subsampling as pooling operation.

then universality is possible. The effect of subsampling in CNNs from the viewpoint of approximation theory is further discussed in [Zho20a]. The role of other types of pooling in enhancing invariances of the hypothesis set will be discussed in Subsection 7.1 below.

6.2 Residual neural networks

Let us first illustrate a potential obstacle when training deep NNs. Consider for $L \in \mathbb{N}$ the product operation

$$\mathbb{R}^L \ni x \mapsto \pi(x) = \prod_{\ell=1}^L x_{\ell}.$$

It is clear that

$$\frac{\partial}{\partial x_k} \pi(x) = \prod_{\ell \neq k} x_{\ell}, \quad x \in \mathbb{R}^L.$$

Therefore, for sufficiently large L , we expect that $|\frac{\partial \pi}{\partial x_k}|$ will be exponentially small, if $|x_{\ell}| < \lambda < 1$ for all $\ell \in [L]$ or exponentially large, if $|x_{\ell}| > \lambda > 1$ for all $\ell \in [L]$. The output of a general NN, considered as a directed graph, is found by repeatedly multiplying the input with parameters in every layer along the paths that lead from the input to the output neuron. Due to the aforementioned phenomenon, it is often observed that training NNs suffers from either the exploding or the vanishing gradient problem, which may prevent lower layers from training at all. The presence of an activation function is likely to exacerbate this effect. The exploding or vanishing gradient problem seems to be a serious obstacle towards efficient training of deep NNs.

In addition to the vanishing and exploding gradient problems, there is an empirically observed *degradation problem* [HZRS16]. This phrase describes the phenomenon that FC NNs seem to achieve lower accuracy on both the training and test data when increasing their depth.

From an approximation theoretic perspective, deep NNs should always be superior to shallow NNs. The reason for this is that NNs with two layers can either exactly represent the identity map or approximate it arbitrarily well. Concretely, for the ReLU activation function ϱ_R we have that $x = \varrho_R(x + b) - b$ for $x \in \mathbb{R}^d$ with $x_i > -b_i$, where $b \in \mathbb{R}^d$. In addition, for any activation function ϱ which is continuously differentiable on a neighborhood of some point $\lambda \in \mathbb{R}$ with $\varrho'(\lambda) \neq 0$ one can approximate the identity arbitrary well, see (1.8). Because of this, extending a NN architecture by one layer can only enlarge the associated hypothesis set.

Therefore, one may expect that the degradation problem is more associated with the optimization aspect of learning. This problem is addressed by a small change to the architecture of a feed-forward NN in [HZRS16].

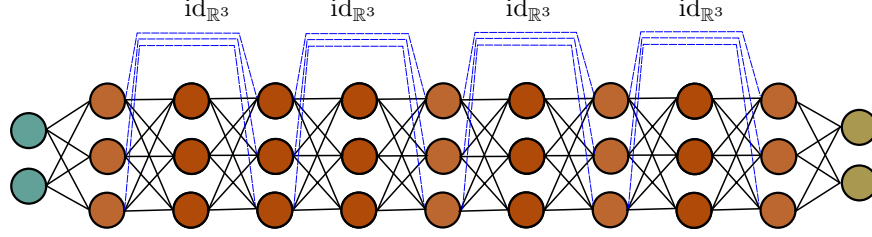


Figure 6.2: Illustration of a neural network with residual blocks.

Instead of defining a FC NN Φ as in (1.1), one can insert a residual block in the ℓ -th layer by redefining²⁷

$$\bar{\Phi}^{(\ell)}(x, \theta) = \varrho(\Phi^{(\ell)}(x, \theta)) + \bar{\Phi}^{(\ell-1)}(x, \theta), \quad (6.2)$$

where we assume that $N_\ell = N_{\ell-1}$. Such a block can be viewed as the sum of a regular FC NN and the identity which is referred to as skip connection or *residual connection*. A sketch of a NN with residual blocks is shown in Figure 6.2. Inserting a residual block in all layers leads to a so-called *residual NN*.

A prominent approach to analyze residual NNs is by establishing a connection with optimal control problems and dynamical systems [E17, TvG18, EHL19, LLS19, RH19, LML⁺20]. Concretely, if each layer of a NN Φ is of the form (6.2), then we have that

$$\bar{\Phi}^{(\ell)} - \bar{\Phi}^{(\ell-1)} = \varrho(\Phi^{(\ell)}) =: h(\ell, \Phi^{(\ell)}),$$

where we abbreviate $\bar{\Phi}^{(\ell)} = \bar{\Phi}^{(\ell)}(x, \theta)$ and set $\bar{\Phi}^{(0)} = x$. Hence, $(\bar{\Phi}^{(\ell)})_{\ell=0}^{L-1}$ corresponds to an Euler discretization of the ODE

$$\dot{\phi}(t) = h(t, \phi(t)), \quad \phi(0) = x,$$

where $t \in [0, L-1]$ and h is an appropriate function.

Using this relationship, deep residual NNs can be studied in the framework of the well-established theory of dynamical systems, where strong mathematical guarantees can be derived.

6.3 Framelets and U-Nets

One of the most prominent application areas of deep NNs are inverse problems, particularly those in the field of imaging science, see also Subsection 8.1. A specific architectural design of CNNs, namely so-called *U-nets* introduced in [RFB15], seems to perform best for this range of problems. We depict a sketch of a U-net in Figure 6.3. However, a theoretical understanding of the success of this architecture was lacking.

Recently, an innovative approach called *deep convolutional framelets* was suggested in [YHC18], which we now briefly explain. The core idea is to take a frame-theoretic viewpoint, see, e.g., [CKP12], and regard the forward pass of a CNN as a decomposition in terms of a frame (in the sense of a generalized basis). A similar approach will be taken in Subsection 7.2 for understanding the learned kernels using sparse coding. However, based on the analysis and synthesis operators of the corresponding frame, the usage of deep convolutional framelets naturally leads to a theoretical understanding of encoder-decoder architectures, such as U-nets.

Let us describe this approach for one-dimensional convolutions on the group $G := \mathbb{Z}/(d\mathbb{Z})$ with kernels defined on the subgroup $H := \mathbb{Z}/(n\mathbb{Z})$, where $d, n \in \mathbb{N}$ with $n < d$, see also Subsection 6.1. We define the convolution between $u \in \mathbb{R}^G$ and $v \in \mathbb{R}^H$ by zero-padding v , i.e., $g \circ v := g * \bar{v}$, where $\bar{v} \in \mathbb{R}^G$ is defined by $\bar{v}_i = v_i$ for $i \in H$ and $\bar{v}_i = 0$ else. As an important tool, we consider the Hankel matrix $\mathbb{H}_n(x) = (x_{i+j})_{i \in G, j \in H} \in \mathbb{R}^{d \times n}$ associated with $x \in \mathbb{R}^G$. As one key property, matrix-vector multiplications with Hankel matrices are translated to convolutions via²⁸

$$\langle e^{(i)}, \mathbb{H}_n(x)v \rangle = \sum_{j \in H} x_{i+j}v_j = \langle x, e^{(i)} \circ v \rangle, \quad i \in G, \quad (6.3)$$

²⁷One can also skip multiple layers, e.g., in [HZRS16] two or three layers skipped, use a simple transformation instead of the identity [SGS15], or randomly drop layers [HSL⁺16].

²⁸Here and in the following we naturally identify elements in \mathbb{R}^G and \mathbb{R}^H with the corresponding vectors in \mathbb{R}^d and \mathbb{R}^n .

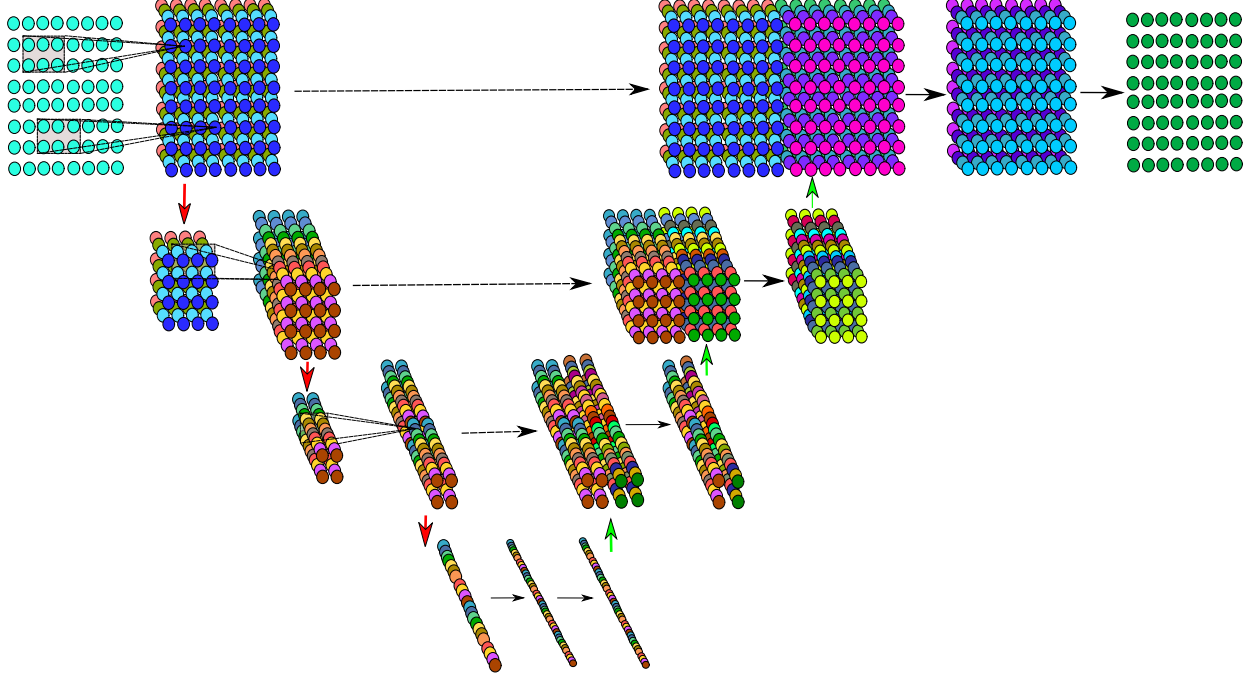


Figure 6.3: Illustration of a simplified U-net neural network. Down-arrows stand for pooling, up arrows for deconvolution or upsampling, right arrows for convolution or fully connected steps. Dashed lines are skip connections.

where $e^{(i)} := \mathbb{1}_{\{i\}} \in \mathbb{R}^G$ and $v \in \mathbb{R}^H$, see [YGLD17]. Further, we can recover the k -th coordinate of x by the Frobenius inner product between $\mathbb{H}_n(x)$ and the Hankel matrix associated with $e^{(k)}$, i.e.,

$$\frac{1}{n} \text{Tr}(\mathbb{H}_n(e^{(k)})^T \mathbb{H}_n(x)) = \frac{1}{n} \sum_{j \in H} \sum_{i \in G} e_{i+j}^{(k)} x_{i+j} = \frac{1}{n} |H| x_k = x_k. \quad (6.4)$$

This allows us to construct global and local bases as follows: Let $p, q \in \mathbb{N}$, let $U = [u_1 \cdots u_p] \in \mathbb{R}^{d \times p}$, $V = [v_1 \cdots v_q] \in \mathbb{R}^{n \times q}$, $\tilde{U} = [\tilde{u}_1 \cdots \tilde{u}_p] \in \mathbb{R}^{d \times p}$, and $\tilde{V} = [\tilde{v}_1 \cdots \tilde{v}_q] \in \mathbb{R}^{n \times q}$, and assume that

$$\mathbb{H}_n(x) = \tilde{U} U^T \mathbb{H}_n(x) V \tilde{V}^T. \quad (6.5)$$

For $p \geq d$ and $q \geq n$, this is, for instance, satisfied if U and V constitute frames with \tilde{U} and \tilde{V} being their respective dual frames, i.e., $\tilde{U} U^T = I_d$ and $V \tilde{V}^T = I_n$. As a special case, one can consider orthonormal bases $U = \tilde{U}$ and $V = \tilde{V}$ with $p = d$ and $q = n$. In the case $p = q = r \leq n$, where r is the rank of $\mathbb{H}_n(x)$, one can establish (6.5) by choosing the left and right singular vectors of $\mathbb{H}_n(x)$ as $U = \tilde{U}$ and $V = \tilde{V}$, respectively. The identity in (6.5), in turn, ensures the following decomposition:

$$x = \frac{1}{n} \sum_{i=1}^p \sum_{j=1}^q \langle x, u_i \circ v_j \rangle \tilde{u}_i \circ \tilde{v}_j. \quad (6.6)$$

Observing that the vector $v_j \in \mathbb{R}^H$ interacts locally with $x \in \mathbb{R}^G$ due to the fact that $H \subset G$, whereas $u_i \in \mathbb{R}^G$ acts on the entire vector x , we refer to $(v_j)_{j=1}^q$ as local and $(u_i)_{i=1}^p$ as global bases. In the context of CNNs, v_i can be interpreted as local convolutional kernel and u_i as pooling operation²⁹. The proof of (6.6)

²⁹Note that $\langle x, u_i \circ v_j \rangle$ can also be interpreted as $\langle u_i, v_j \star x \rangle$, where \star denotes the cross-correlation between the zero-padded v_j and x . This is in line with software implementations for deep learning applications, e.g., TensorFlow and PyTorch, where typically cross-correlations are used instead of convolutions.

follows directly from properties (6.3), (6.4), and (6.5) as

$$x_k = \frac{1}{n} \text{Tr}(\mathbb{H}_n(e^{(k)})^T \mathbb{H}_n(x)) = \frac{1}{n} \text{Tr}(\mathbb{H}_n(e^{(k)})^T \tilde{U} U^T \mathbb{H}_n(x) V \tilde{V}^T) = \frac{1}{n} \sum_{i=1}^p \sum_{j=1}^q \langle u_i, \mathbb{H}_n(x) v_j \rangle \langle \tilde{u}_i, \mathbb{H}_n(e^{(k)}) \tilde{v}_j \rangle.$$

The decomposition in (6.6) can now be interpreted as a composition of an encoder and a decoder,

$$x \mapsto C = (\langle x, u_i *_{\circ} v_j \rangle)_{i \in [p], j \in [q]} \quad \text{and} \quad C \mapsto \frac{1}{n} \sum_{i=1}^p \sum_{j=1}^q C_{i,j} \tilde{u}_i *_{\circ} \tilde{v}_j, \quad (6.7)$$

which relates it to CNNs equipped with an encoder-decoder structure such as U-nets, see Figure 6.3. Generalizing this approach to multiple channels, it is possible to stack such encoders and decoders which leads to a layered version of (6.6). In [YHC18] it is shown that one can make an informed decision on the number of layers based on the rank of $\mathbb{H}_n(x)$, i.e., the complexity of the input features x . Moreover, also an activation function such as the ReLU or bias vectors can be included. The key question one can then ask is how the kernels can be chosen to obtain sparse coefficients C in (6.7) and a decomposition such as (6.6), i.e., perfect reconstruction. If U and V are chosen as the left and right singular vectors of $\mathbb{H}_n(x)$, one obtains a very sparse, however input-dependent, representation in (6.6) due to the fact that

$$C_{i,j} = \langle x, u_i *_{\circ} v_j \rangle = \langle u_i, \mathbb{H}_n(x) v_j \rangle = 0, \quad i \neq j.$$

Finally, using the framework of deep convolutional framelets, theoretical reasons for including skip connections can be derived, since they aid to obtain a perfect reconstruction.

6.4 Batch normalization

Batch normalization is a building block of NNs that was invented in [IS15] with the goal to reduce so-called *internal covariance shift*. In essence, this phrase describes the (undesirable) situation where during training each layer receives inputs with different distribution. A batch normalization block is defined as follows: For points $b = (y^{(i)})_{i=1}^m \in (\mathbb{R}^n)^m$ and $\beta, \gamma \in \mathbb{R}$, we define

$$\text{BN}_b^{(\beta, \gamma)}(y) := \gamma \frac{y - \mu_b}{\sigma_b} + \beta, \quad y \in \mathbb{R}^n, \quad \text{with} \quad \mu_b = \frac{1}{m} \sum_{i=1}^m y^{(i)} \quad \text{and} \quad \sigma_b^2 = \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \mu_b)^2, \quad (6.8)$$

where all operations are to be understood componentwise, see Figure 6.4.

Such a batch normalization block can be added into a NN architecture. Then b is the output of the previous layer over a batch or the whole training data³⁰. Furthermore, the parameters β, γ are variable and can be learned during training. Note that, if one sets $\beta = \mu_b$ and $\gamma = \sigma_b$, then $\text{BN}_b^{(\beta, \gamma)}(y) = y$ for all $y \in \mathbb{R}^n$. Therefore, a batch normalization block does not negatively affect the expressivity of the architecture. On the other hand, batch normalization does have a tangible effect on the optimization aspects of deep learning. Indeed, in [STIM18, Theorem 4.1], the following observation was made:

³⁰In practice, one typically uses a moving average to estimate the mean μ and the standard deviation σ of the output of the previous layer over the whole training data by only using batches.

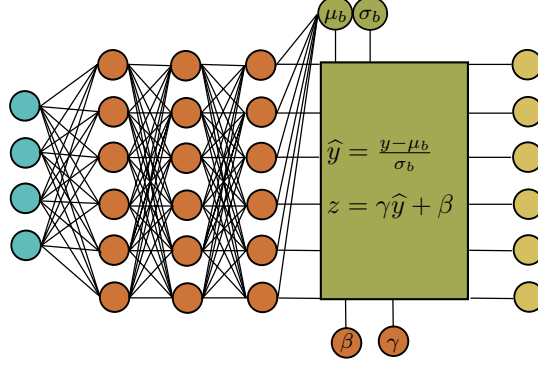


Figure 6.4: A batch normalization block after a fully connected neural network. The parameters μ_b, σ_b are the mean and the standard deviation of the output of the fully connected network computed over a batch s , i.e., a set of inputs. The parameters β, γ are learnable parts of the batch normalization block.

Proposition 6.1 (Smoothing effect of batch normalization). *Let $m \in \mathbb{N}$ with $m \geq 2$ and for every $\beta, \gamma \in \mathbb{R}$ define $\mathcal{B}^{(\beta, \gamma)}: \mathbb{R}^m \rightarrow \mathbb{R}^m$ by*

$$\mathcal{B}^{(\beta, \gamma)}(b) = (\text{BN}_b^{(\beta, \gamma)}(y^{(1)}), \dots, \text{BN}_b^{(\beta, \gamma)}(y^{(m)})), \quad b = (y^{(i)})_{i=1}^m \in \mathbb{R}^m,$$

where $\text{BN}_b^{(\beta, \gamma)}$ is given as in (6.8). Let $\beta, \gamma \in \mathbb{R}$ and let $r: \mathbb{R}^m \rightarrow \mathbb{R}$ be a differentiable function. Then it holds for every $b \in \mathbb{R}^m$ that

$$\|\nabla(r \circ \mathcal{B}^{(\beta, \gamma)})(b)\|_2^2 = \frac{\gamma^2}{\sigma_b^2} \left(\|\nabla r(b)\|^2 - \frac{1}{m} \langle \mathbf{1}, \nabla r(b) \rangle^2 - \frac{1}{m} \langle \mathcal{B}^{(0,1)}(b), \nabla r(b) \rangle^2 \right),$$

where $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^m$ and σ_b^2 is given as in (6.8).

For multi-dimensional $y^{(i)} \in \mathbb{R}^n$, $i \in [m]$, the same statement holds for all components as, by definition, the batch normalization block acts componentwise. Proposition 6.1 follows from a convenient representation of the Jacobian of the mapping $\mathcal{B}^{(\beta, \gamma)}$, given by

$$\frac{\partial \mathcal{B}^{(\beta, \gamma)}(b)}{\partial b} = \frac{\gamma}{\sigma_b} \left(\mathbf{I}_m - \frac{1}{m} \mathbf{1} \mathbf{1}^T - \frac{1}{m} \mathcal{B}^{(0,1)}(b) (\mathcal{B}^{(0,1)}(b))^T \right), \quad b \in \mathbb{R}^m,$$

and the fact that $\{\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}} \mathcal{B}^{(0,1)}(b)\}$ constitutes an orthonormal set.

Choosing r to mimic the empirical risk of a learning task, Proposition 6.1 shows that, in certain situations—for instance, if γ is smaller than σ_b or if m is not too large—a batch normalization block can considerably reduce the magnitude of the derivative of the empirical risk with respect to the input of the batch normalization block. By the chain rule, this implies that also the derivative of the empirical risk with respect to NN parameters influencing the input of the batch normalization block is reduced.

Interestingly, a similar result holds for second derivatives [STIM18, Theorem 4.2] if r is twice differentiable. One can conclude that adding a batch normalization block increases the smoothness of the optimization problem. Since the parameters β and γ were introduced, including a batch normalization block also increases the dimension of the optimization problem by two.

6.5 Sparse neural networks and pruning

For deep FC NNs, the number of trainable parameters usually scales like the square of the number of neurons. For reasons of computational complexity and memory efficiency, it appears sensible to seek for techniques to reduce the number of parameters or extract *sparse subnetworks* (see Figure 6.5) without affecting the output

of a NN much. One way to do this is by *pruning* [LDS89, HMD16]. Here, certain parameters of a NN are removed after training. This is done, for example, by setting these parameters to zero.

In this context, the *lottery ticket hypothesis* was formulated in [FC18]. It states: “A randomly-initialized, dense NN contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original NN after training for at most the same number of iterations”. In [RWK⁺20] a similar hypothesis was made and empirically studied. There, it is claimed that, for a sufficiently overparametrized NN, there exists a subnetwork that matches the performance of the large NN after training without being trained itself, i.e., already at initialization.

Under certain simplifying assumptions, the existence of favorable subnetworks is quite easy to prove. We can use a technique that was previously indirectly used in Subsection 4.2—the Carathéodory Lemma. This result states the following: Let $n \in \mathbb{N}$, $C \in (0, \infty)$, and let $(\mathcal{H}, \|\cdot\|)$ be a Hilbert space. Let $F \subset \mathcal{H}$ with $\sup_{f \in F} \|f\| \leq C$ and let $g \in \mathcal{H}$ be in the convex hull of F . Then there exist $f_i \in F$, $i \in [n]$, and $c \in [0, 1]^n$ with $\|c\|_1 = 1$ such that

$$\left\| g - \sum_{i=1}^n c_i f_i \right\| \leq \frac{C}{\sqrt{n}},$$

see, e.g., [Ver18, Theorem 0.0.2].

Proposition 6.2 (Carathéodory pruning). *Let $d, n \in \mathbb{N}$, with $n \geq 100$ and let μ be a probability measure on the unit ball $B_1(0) \subset \mathbb{R}^d$. Let $a = ((d, n, 1), \varrho_R)$ be the architecture of a two-layer ReLU network and let $\theta \in \mathbb{R}^{P((d, n, 1))}$ be corresponding parameters such that*

$$\Phi_a(\cdot, \theta) = \sum_{i=1}^n w_i^{(2)} \varrho_R(\langle w_i^{(1)}, \cdot \rangle + b_i^{(1)}),$$

where $(w_i^{(1)}, b_i^{(1)}) \in \mathbb{R}^d \times \mathbb{R}$, $i \in [n]$, and $w^{(2)} \in \mathbb{R}^n$. Assume that for every $i \in [n]$ it holds that $\|w_i^{(1)}\|_2 \leq 1/2$ and $b_i^{(1)} \leq 1/2$. Then there exists a parameter $\tilde{\theta} \in \mathbb{R}^{P((d, n, 1))}$ with at least 99% of its entries being zero such that

$$\|\Phi_a(\cdot, \theta) - \Phi_a(\cdot, \tilde{\theta})\|_{L^2(\mu)} \leq \frac{15\|w^{(2)}\|_1}{\sqrt{n}}.$$

Specifically, there exists an index set $I \subset [n]$ with $|I| \leq n/100$ such that $\tilde{\theta}$ satisfies that

$$\tilde{w}_i^{(2)} = 0, \quad \text{if } i \notin I, \quad \text{and} \quad (\tilde{w}_i^{(1)}, \tilde{b}_i^{(1)}) = \begin{cases} (w_i^{(1)}, b_i^{(1)}), & \text{if } i \in I, \\ (0, 0), & \text{if } i \notin I. \end{cases}$$

The result is clear if $w^{(2)} = 0$. Otherwise, define

$$f_i := \|w^{(2)}\|_1 \varrho_R(\langle w_i^{(1)}, \cdot \rangle + b_i^{(1)}), \quad i \in [n],$$

and observe that $\Phi_a(\cdot, \theta)$ is in the convex hull of $\{f_i\}_{i=1}^n \cup \{-f_i\}_{i=1}^n$. Moreover, by the Cauchy–Schwarz inequality, it holds that

$$\|f_i\|_{L^2(\mu)} \leq \|w^{(2)}\|_1 \|f_i\|_{L^\infty(B_1(0))} \leq \|w^{(2)}\|_1.$$

We conclude with the Carathéodory Lemma that there exists $I \subset [n]$ with $|I| = \lfloor n/100 \rfloor \geq n/200$ and $c_i \in [-1, 1]$, $i \in I$, such that

$$\left\| \Phi_a(\cdot, \theta) - \sum_{i \in I} c_i f_i \right\|_{L^2(\mu)} \leq \frac{\|w^{(2)}\|_1}{\sqrt{|I|}} \leq \frac{\sqrt{200}\|w^{(2)}\|_1}{\sqrt{n}},$$

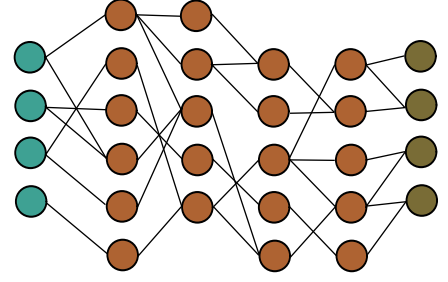


Figure 6.5: A neural network with sparse connections.

which yields the result.

Proposition 6.2 shows that certain, very wide NNs can be approximated very well by sparse subnetworks where only the output weight matrix needs to be changed. The argument of Proposition 6.2 is inspired by [BK18], where a much more refined result is shown for deep NNs.

6.6 Recurrent neural networks

Recurrent NNs are NNs where the underlying graph is allowed to exhibit cycles as in Figure 6.6, see [Hop82, RHW86, Elm90, Jor90]. Previously, we had excluded cyclic computational graphs. For a feed-forward NN, the computation of internal states is naturally performed step by step through the layers. Since the output of a layer does not affect previous layers, the order in which the computations of the NN are performed corresponds to the order of the layers. For recurrent NNs, the concept of layers does not exist, and the order of operations is much more delicate. Therefore, one considers time steps. In each time step, all possible computations of the graph are applied to the current state of the NN. This yields a new internal state. Given that time steps arise naturally from the definition of recurrent NNs, this NN type is typically used for sequential data.

If the input to a recurrent NN is a sequence, then every input determines the internal state of the recurrent NN for the following inputs. Therefore, one can claim that these NNs exhibit a memory. This fact is extremely desirable in natural language processing, which is why recurrent NNs are widely used in this application.

Recurrent NNs can be trained similarly to regular feed-forward NNs by an algorithm called *backpropagation through time* [MP69, Wer88, WZ95]. This procedure essentially unfolds the recurrent structure yielding a classical NN structure. However, the algorithm may lead to very deep structures. Due to the vanishing and exploding gradient problem discussed earlier, very deep NNs are often hard to train. Because of this, special recurrent structures were introduced that include gates which prohibit too many recurrent steps; these include the widely used LSTMs [HS97].

The application area of recurrent NNs is typically quite different from that of regular NNs since they are specialized on sequential data. Therefore, it is hard to quantify the effect of a recurrent connection on a fully connected NN. However, it is certainly true that with recurrent connections certain computations can be performed much more efficiently than with feed-forward NN structures. A particularly interesting construction can be found in [BF19, Theorem 4.4], where it is shown that a fixed size, recurrent NN with ReLU activation function, can approximate the function $x \mapsto x^2$ to any desired accuracy. The reason for this efficient representation can be seen when considering the self-referential definition of the approximant to $x - x^2$ shown in Figure 3.2.

On the other hand, with feed-forward NNs, it transpires from Theorem 3.3 that the approximation error of fixed-sized ReLU NNs for any non-affine function is greater than a positive lower bound.

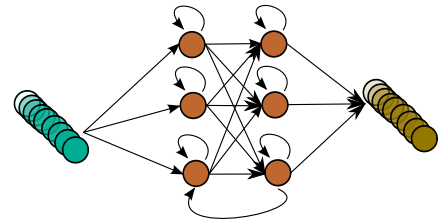


Figure 6.6: Sketch of a recurrent neural network. Cycles in the computational graph incorporate the sequential structure of the input and output.

7 Describing the features a deep neural network learns

This section presents two viewpoints which help in understanding the nature of features that can be described by NNs. Section 7.1 summarizes aspects of the so-called *scattering transform* which constitutes a specific NN architecture that can be shown to satisfy desirable properties, such as translation and deformation invariance. Section 7.2 relates NN features to the current paradigm of *sparse coding*.

7.1 Invariances and the scattering transform

One of the first theoretical contributions to the understanding of the mathematical properties of CNNs is [Mal12]. The approach taken in that work is to consider specific CNN architectures with *fixed* parameters

that result in a stand-alone feature descriptor whose output may be fed into a subsequent classifier (for example, a kernel support vector machine or a trainable FC NN). From an abstract point of view, a feature descriptor is a function Ψ mapping from a signal space, such as $L^2(\mathbb{R}^d)$ or the space of piecewise smooth functions, to a feature space. In an ideal world, such a classifier should “factor” out invariances that are irrelevant to a subsequent classification problem while preserving all other information of the signal. A very simple example of a classifier which is invariant under translations is the Fourier modulus $\Psi: L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$, $u \mapsto |\hat{u}|$. This follows from the fact that a translation of a signal u results in a modulation of its Fourier transform, i.e., $\widehat{u(\cdot - \tau)}(\omega) = e^{-2\pi i \langle \tau, \omega \rangle} \hat{u}(\omega)$, $\tau, \omega \in \mathbb{R}^d$. Furthermore, in most cases (for example, if u is a generic compactly supported function [GKR20]), u can be reconstructed up to a translation from its Fourier modulus [GKR20] and an energy conservation property of the form $\|\Psi(u)\|_{L^2} = \|u\|_{L^2}$ holds true. Translation invariance is, for example, typically exhibited by image classifiers, where the label of an image does not change if it is translated.

In practical problems many more invariances arise. Providing an analogous representation that factors out general invariances would lead to a significant reduction in the problem dimensionality and constitutes an extremely promising route towards dealing with the very high dimensionality that is commonly encountered in practical problems [Mal16]. This program is carried out in [Mal12] for additional invariances with respect to deformations $u \mapsto u_\tau := u(\cdot - \tau(\cdot))$, where $\tau: \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a smooth mapping. Such transformations may occur in practice, for instance, as image warpings. In particular, a feature descriptor Ψ is designed that, with a suitable norm $\|\cdot\|$ on the image of Ψ ,

- (a) is Lipschitz continuous with respect to deformations in the sense that $\|\Psi(u) - \Psi(u_\tau)\| \lesssim K(\tau, \nabla\tau, \nabla^2\tau)$ holds for some K that only mildly depends on τ and essentially grows linearly in $\nabla\tau$ and $\nabla^2\tau$,
- (b) is almost (i.e., up to a small and controllable error) invariant under translations of the input data, and
- (c) contains all relevant information on the input data in the sense that an energy conservation property

$$\|\Psi(u)\| \approx \|u\|_{L^2}$$

holds true.

Observe that, while the action of translations only represents a d -parameter group, the action of deformations/warpings represents an infinite-dimensional group. Hence, a deformation invariant feature descriptor represents a big potential for dimensionality reduction. Roughly speaking, the feature descriptor Ψ of [Mal12] (also coined the *scattering transform*) is defined by collecting features that are computed by iteratively applying a wavelet transform followed by a pointwise modulus non-linearity and a subsequent low-pass filtering step, i.e.,

$$||u * \psi_{j_1}| * \psi_{j_2} * \dots * \psi_{j_\ell}| * \varphi_J,$$

where ψ_j refers to a wavelet at scale j and φ_J refers to a scaling function at scale J . The collection of all these so-called *scattering coefficients* can then be shown to satisfy the properties in (a)–(c) above in a suitable (asymptotic) sense. The proof of this result relies on a subtle interplay between a “deformation covariance” property of the wavelet transform and a “regularizing” property of the operation of convolution with the modulus of a wavelet. We remark that similar results can be shown also for different systems, such as Gabor frames [WGB17, CL19].

7.2 Hierarchical sparse representations

The previous approach modeled the learned features by a specific dictionary, namely wavelets. It is well known that one of the striking properties of wavelets is to provide sparse representations for functions belonging to certain function classes. More generally, we speak of sparse representations with respect to a representation system. For a vector $x \in \mathbb{R}^d$, a sparsifying representation system $D \in \mathbb{R}^{d \times p}$ —also called *dictionary*—is such that $x = D\phi$ with the coefficients $\phi \in \mathbb{R}^p$ being sparse in the sense that $\|\phi\|_0 := |\text{supp}(\phi)| = |\{i \in [p] : \phi_i \neq 0\}|$ is small compared to p . A similar definition can be made for signals in infinite-dimensional spaces. Taking

sparse representations into account, the theory of sparse coding provides an approach to a theoretical understanding of the features a deep NN learns.

One common method in image processing is the utilization of not the entire image but overlapping patches of it, coined *patch-based image processing*. Thus of particular interest are local dictionaries which sparsify those patches, but presumably not the global image. This led to the introduction of the *convolutional sparse coding* model (CSC model), which links such local and global behaviors. Let us describe this model for one-dimensional convolutions on the group $G := \mathbb{Z}/(d\mathbb{Z})$ with kernels supported on the subgroup $H := \mathbb{Z}/(n\mathbb{Z})$, where $d, n \in \mathbb{N}$ with $n < d$, see also Subsection 6.1. The corresponding CSC model is based on a decomposition of a global signal $x \in (\mathbb{R}^G)^c$ with $c \in \mathbb{N}$ channels as

$$x_i = \sum_{j=1}^C \kappa_{i,j} * \phi_j, \quad i \in [c], \quad (7.1)$$

where $\phi \in (\mathbb{R}^G)^C$ is supposed to be a sparse representation with $C \in \mathbb{N}$ channels and $\kappa_{i,j} \in \mathbb{R}^G$, $i \in [c]$, $j \in [C]$, are local kernels with $\text{supp}(\kappa_{i,j}) \subset H$. Let us consider a patch $((x_i)_{g+h})_{i \in [c], h \in H}$ of n adjacent entries, starting at position $g \in G$, in each channel of x . The condition on the support of the kernels $\kappa_{i,j}$ and the representation in (7.1) imply that this patch is only affected by a stripe of at most $(2n-1)$ entries in each channel of ϕ . The local, patch-based sparsity of the representation ϕ can thus be appropriately measured via

$$\|\phi\|_{0,\infty}^{(n)} := \max_{g \in G} \|((\phi_j)_{g+k})_{j \in [C], k \in [2n-1]}\|_0,$$

see [PSE17]. Furthermore, note that we can naturally identify x and ϕ with vectors in \mathbb{R}^{dc} and \mathbb{R}^{dC} and write $x = D\phi$, where $D \in \mathbb{R}^{dc \times dC}$ is a matrix consisting of circulant blocks, typically referred to as a *convolutional dictionary*.

The relation between the CSC model and deep NNs is revealed by applying the CSC model in a layer-wise fashion [PRE17, SPRE18, PRSE18]. To see this, let $C_0 \in \mathbb{N}$ and for every $\ell \in [L]$ let $C_\ell, k_\ell \in \mathbb{N}$ and let $D^{(\ell)} \in \mathbb{R}^{dC_{\ell-1} \times dC_\ell}$ be a convolutional dictionary with kernels supported on $\mathbb{Z}/(n_\ell\mathbb{Z})$. A signal $x = \phi^{(0)} \in \mathbb{R}^{dC_0}$ is said to belong to the corresponding *multi-layered CSC model* (ML-CSC model) if there exist coefficients $\phi^{(\ell)} \in \mathbb{R}^{dC_\ell}$ with

$$\phi^{(\ell-1)} = D^{(\ell)} \phi^{(\ell)} \quad \text{and} \quad \|\phi^{(\ell)}\|_{0,\infty}^{(n_\ell)} \leq k_\ell, \quad \ell \in [L]. \quad (7.2)$$

We now consider the problem of reconstructing the sparse coefficients $(\phi^{(\ell)})_{\ell=1}^L$ from a noisy signal $\tilde{x} := x + \nu$, where the noise $\nu \in \mathbb{R}^{dC_0}$ is assumed to have small ℓ^2 -norm and x is assumed to follow the ML-CSC model in (7.2). In general, this problem is NP-hard. However, under suitable conditions on the ML-CSC model, it can be approximately solved, for instance, by a layered thresholding algorithm.

More precisely, for $D \in \mathbb{R}^{dc \times dC}$ and $b \in \mathbb{R}^{dC}$, we define a *soft-thresholding operator* by

$$\mathcal{T}_{D,b}(x) := \varrho_R(D^T x - b) - \varrho_R(-D^T x - b), \quad x \in \mathbb{R}^{dc}, \quad (7.3)$$

where $\varrho_R(x) = \max\{0, x\}$ is applied componentwise. If $x = D\phi$ as in (7.1), we obtain $\phi \approx \mathcal{T}_{D,b}(x)$ roughly under the following conditions: The distance of ϕ and $\psi := D^T x = D^T D\phi$ can be bounded using the local sparsity of ϕ and the mutual coherence and locality of the kernels of the convolutional dictionary D . For a suitable threshold b , the mapping $\psi \mapsto \varrho_R(\psi - b) - \varrho_R(-\psi - b)$ further recovers the support of ϕ by nullifying entries of ψ with $\psi_i \leq |b_i|$. Utilizing the soft-thresholding operator (7.3) iteratively for corresponding vectors $b^{(\ell)} \in \mathbb{R}^{dC_\ell}$, $\ell \in [L]$, then suggests the following approximations:

$$\phi^{(\ell)} \approx (\mathcal{T}_{D^{(\ell)}, b^{(\ell)}} \circ \cdots \circ \mathcal{T}_{D^{(1)}, b^{(1)}})(\tilde{x}), \quad \ell \in [L].$$

The resemblance with the realization of a CNN with ReLU activation function is evident. The transposed dictionary $(D^{(\ell)})^T$ can be regarded as modeling the learned convolutional kernels, the threshold $b^{(\ell)}$ models the bias vector, and the soft-thresholding operator $\mathcal{T}_{D^{(\ell)}, b^{(\ell)}}$ mimics the application of a convolutional block with a ReLU non-linearity in the ℓ -th layer.

Using this model, a theoretical understanding of CNNs from the perspective of sparse coding is now at hand. This novel perspective gives a precise mathematical meaning of the kernels in a CNN as sparsifying dictionaries of an ML-CSC model. Moreover, the forward pass of a CNN can be understood as a layered thresholding algorithm for decomposing a noisy signal \tilde{x} . The results derived are then of the following flavor: Given a suitable reconstruction procedure such as thresholding or ℓ_1 -minimization, the sparse coefficients $(\phi^{(\ell)})_{\ell=1}^L$ of a signal x following a ML-CSC model can be stably recovered from the noisy signal \tilde{x} under certain hypotheses on the ingredients of the ML-CSC model.

8 Effectiveness in natural sciences

The theoretical insights of the previous sections do not always accurately describe the performance of NNs in applications. Indeed, there often exists a considerable gap between the predictions of approximation theory and the practical performance of NNs [AD20].

In this section, we consider concrete applications which have been very successfully solved with deep-learning-based methods. In Section 8.1 we present an overview of deep-learning-based algorithms applied to inverse problems. Section 8.2 then continues by describing how NNs can be used as a numerical ansatz for solving PDEs, highlighting their use in the solution of the multi-electron Schrödinger equation.

8.1 Deep neural networks meet inverse problems

The area of inverse problems, predominantly in imaging, was presumably the first class of mathematical methods embracing deep learning with overwhelming success. Let us consider a forward operator $K: \mathcal{Y} \rightarrow \mathcal{X}$ with \mathcal{X}, \mathcal{Y} being Hilbert spaces and the associated inverse problem of finding $y \in \mathcal{Y}$ such that $Ky = x$ for given features $x \in \mathcal{X}$. The classical model-based approach to regularization aims to approximate K by invertible operators, and is hence strongly based on functional analytic principles. Today, such approaches take well-posedness of the approximation, convergence properties, as well as the structure of regularized solutions into account. The last item allows to incorporate prior information of the original solution such as regularity, sharpness of edges, or—in the case of sparse regularization [JMS17]—a sparse coefficient sequence with respect to a prescribed representation system. Such approaches are typically realized in a variational setting and hence aim to minimize functionals of the form

$$\|Ky - x\|^2 + \alpha R(y),$$

where $\alpha \in (0, \infty)$ is a regularization parameter, $R: \mathcal{Y} \rightarrow [0, \infty)$ a regularization term, and $\|\cdot\|$ denotes the norm on \mathcal{Y} . As said, the regularization term aims to model structural information about the desired solution. However, one main hurdle in this approach is the problem that typically solution classes such as images from computed tomography cannot be modeled accurately enough to, for instance, allow reconstruction under the constraint of a significant amount of missing features.

This has opened the door to data-driven approaches, and recently, deep NNs. Solvers of inverse problems which are based on deep learning techniques can be roughly categorized into three classes:

1. *Supervised approaches:* The most straightforward approach is to train a NN $\Phi(\cdot, \theta): \mathcal{X} \rightarrow \mathcal{Y}$ end-to-end, i.e., to completely learn the map from data x to the solution y . More advanced approaches in this direction incorporate information about the operator K into the NN such as in [AÖ17, GOW19, MLE21]. Yet another type of approaches aims to combine deep NNs with classical model-based approaches. The first suggestion in this realm was to start by applying a standard solver, followed by a deep NN $\Phi(\cdot, \theta): \mathcal{Y} \rightarrow \mathcal{Y}$ which serves as a denoiser for specific reconstruction artifacts, e.g., [JMFU17]. This was followed by more sophisticated methods such as plug-and-play frameworks for coupling inversion and denoising [REM17].
2. *Semi-supervised approaches:* These type of approaches aim to encode the regularization by a deep NN $\Phi(\cdot, \theta): \mathcal{Y} \rightarrow [0, \infty)$. The underlying idea is often to require stronger regularization on solutions $y^{(i)}$

that are more prone to artifacts or other effects of the instability of the problem. On solutions where typically few artifacts are observed less regularization can be used. Therefore, the learning algorithm only requires a set of labels $(y^{(i)})_{i=1}^m$ as well as a method to assess how hard the inverse problem for this label would be. In this sense, the algorithm can be considered semi-supervised. This idea was followed, for example, in [LÖS18, LSAH20]. Taking a Bayesian viewpoint, one can also learn prior distributions as deep NNs, which was done in [BZAJ20].

3. *Unsupervised approaches*: One highlight of what we might coin unsupervised approaches in our problem setting is the introduction of deep image priors in [DKMB20, UVL18]. The key idea is to parametrize the solutions y as the output of a NN $\Phi(\xi, \cdot): \mathcal{P} \rightarrow \mathcal{Y}$ with parameters in a suitable space \mathcal{P} , applied to a fixed input ξ . Then, for given features x , one tries to solve $\min_{\theta \in \mathcal{P}} \|K\Phi(\xi, \theta) - x\|^2$ in order to obtain parameters $\hat{\theta} \in \mathcal{P}$ that yield a solution candidate $y = \Phi(\xi, \hat{\theta})$. Here often early stopping is applied in the training of the network parameters.

As can be seen, one key conceptual question is how to “take the best out of both worlds”, in the sense of optimally combining classical (model-based) methods—in particular the forward operator K —with deep learning. This is certainly sensitively linked to all characteristics of the particular application at hand, such as availability and accuracy of training data, properties of the forward operator, or requirements for the solution. And each of the three classes of hybrid solvers follows a different strategy.

Let us now discuss advantages and disadvantages of methods from the three categories with a particular focus on a mathematical foundation. *Supervised* approaches suffer on the one hand from the problem that often ground-truth data is not available or only in a very distorted form, leading to the fact that synthetic data constitutes a significant part of the training data. Thus the learned NN will mainly perform as well as the algorithm which generated the data, but not significantly improve it—only from an efficiency viewpoint. On the other hand, the inversion is often highly ill-posed, i.e., the inversion map has a large Lipschitz constant, which negatively affects the generalization ability of the NN. Improved approaches incorporate knowledge about the forward operator K as discussed, which helps to circumvent this issue.

One significant advantage of *semi-supervised* approaches is that the underlying mathematical model of the inverse problem is merely augmented by the neural network-based regularization. Assuming that the learned regularizer satisfies natural assumptions, convergence proofs or stability estimates for the resulting regularized methods are still available.

Finally, *unsupervised* approaches have the advantage that the regularization is then fully due to the specific architecture of the deep NN. This makes these methods slightly easier to understand theoretically, although, for instance, the deep prior approach in its full generality is still lacking a profound mathematical analysis.

8.2 PDE-based models

Besides applications in image processing and artificial intelligence, deep learning methods have recently strongly impacted the field of numerical analysis. In particular, regarding the numerical solution of high-dimensional PDEs. These PDEs are widely used as a model for complex processes and their numerical solution presents one of the biggest challenges in scientific computing. We mention three exemplary problem classes:

1. *Black–Scholes model*: The Nobel award-winning theory of Fischer Black, Robert Merton, and Myron Scholes proposes a linear PDE model for the determination of a fair price of a (complex) financial derivative. The dimensionality of the model corresponds to the number of financial assets which is typically quite large. The classical linear model, which can be solved efficiently via Monte Carlo methods is quite limited. In order to take into account more realistic phenomena such as default risk, the PDE that models a fair price becomes nonlinear, and much more challenging to solve. In particular (with the notable exception of Multilevel Picard algorithms [EHJK19]) no general algorithm exists that provably scales well with the dimension.

2. *Schrödinger equation*: The electronic Schrödinger equation describes the stationary nonrelativistic behavior of a quantum mechanical electron system in the electric field generated by the nuclei of a molecule. Its numerical solution is required to obtain stable molecular configurations, compute vibrational spectra, or obtain forces governing molecular dynamics. If the number of electrons is large, this is again a high-dimensional problem and to date there exist no satisfactory algorithms for its solution: It is well known that different gold standard methods may produce completely different energy predictions, for example, when applied to large delocalized molecules, rendering these methods useless for those problems.
3. *Hamilton–Jacobi–Bellman equation*: The Hamilton–Jacobi–Bellman (HJB) equation models the value function of (deterministic or stochastic) optimal control problems. The underlying dimensionality of the model corresponds to the dimension of the space of states to be controlled and tends to be rather high in realistic applications. The high dimensionality, together with the fact that HJB equations typically tend to be fully nonlinear with non-smooth solutions, renders the numerical solution of HJB equations extremely challenging and no general algorithms exist for this problem.

Due to the favorable approximation results of NNs for high-dimensional functions (see especially Subsection 4.3), it might not come as a surprise that a NN ansatz has proven to be quite successful in solving the aforementioned PDE models. A pioneering work in this direction is [HJE18] which uses the backwards SDE reformulation of semilinear parabolic PDEs to reformulate the evaluation of such a PDE at a specific point as an optimization problem that can be solved by the deep learning paradigm. The resulting algorithm proves quite successful in the high-dimensional regime and, for instance, enables the efficient modeling of complex financial derivatives including nonlinear effects such as default risk. Another approach specifically tailored to the numerical solution of HJB equations is [NZGK21]. In this work, one uses the Pontryagin principle to generate samples of the PDE solution along solutions of the corresponding boundary value problem. Other numerical approaches include the *Deep Ritz Method* [EY18], where a Dirichlet energy is minimized over a set of NNs, or so-called *Physics Informed Neural Networks* [RPK19], where typically the PDE residual is minimized along with some natural constraints, for instance, to enforce boundary conditions.

Deep-learning-based methods arguably work best if they are combined with domain knowledge to inspire NN architecture choices. We would like to illustrate this interplay at the hand of a specific and extremely relevant example: the electronic Schrödinger equation (under the Born–Oppenheimer approximation) which amounts to finding the smallest nonzero eigenvalue of the eigenvalue problem

$$\mathcal{H}_R \psi = \lambda_\psi \psi, \quad (8.1)$$

for $\psi: \mathbb{R}^{3 \times n} \rightarrow \mathbb{R}$, where the Hamiltonian

$$(\mathcal{H}_R \psi)(r) = - \sum_{i=1}^n \frac{1}{2} (\Delta_{r_i} \psi)(r) - \left(\sum_{i=1}^n \sum_{j=1}^p \frac{Z_j}{\|r_i - R_j\|_2} - \sum_{i=1}^{p-1} \sum_{j=i+1}^p \frac{Z_i Z_j}{\|R_i - R_j\|_2} - \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{1}{\|r_i - r_j\|_2} \right) \psi(r)$$

describes the kinetic energy (first term) as well as Coulomb attraction force between electrons and nuclei (second and third term) and the Coulomb repulsion force between different electrons (third term). Here, the coordinates $R = [R_1 \dots R_p] \in \mathbb{R}^{3 \times p}$ refer to the positions of the nuclei, $(Z_i)_{i=1}^p \in \mathbb{N}^p$ denote the atomic numbers of the nuclei, and the coordinates $r = [r_1, \dots, r_n] \in \mathbb{R}^{3 \times n}$ refer to the positions of the electrons. The associated eigenfunction ψ describes the so-called *wavefunction* which can be interpreted in the sense that $|\psi(r)|^2 / \|\psi\|_{L^2}^2$ describes the joint probability density of the n electrons to be located at r . The smallest solution λ_ψ of (8.1) describes the *ground state energy* associated with the nuclear coordinates R . It is of particular interest to know the ground state energy for all nuclear coordinates, the so-called *potential energy surface* whose gradient determines the forces governing the dynamic motions of the nuclei. The numerical solution of (8.1) is complicated by the *Pauli principle* which states that the wave function ψ must be antisymmetric in all coordinates representing electrons of equal spin. To state it, we need to clarify that every electron is not only defined by its location but also by its spin which may be positive or negative.

Depending on whether two electrons have the same spin or not, their interaction changes massively. This is reflected by the Pauli principle that we already mentioned: Suppose that electrons i and j have equal spin, then the wave function must satisfy

$$P_{i,j}\psi = -\psi, \quad (8.2)$$

where $P_{i,j}$ denotes the operator that swaps r_i and r_j , i.e., $(P_{i,j}\psi)(r) = \psi(r_1, \dots, r_j, \dots, r_i, \dots, r_n)$. In particular, no two electrons with the same spin can occupy the same location. The challenges associated with solving the Schrödinger equation inspired the following famous quote by Paul Dirac [Dir29]:

“The fundamental laws necessary for the mathematical treatment of a large part of physics and the whole of chemistry are thus completely known, and the difficulty lies only in the fact that application of these laws leads to equations that are too complex to be solved.”

We now describe how deep learning methods might help to mitigate this claim to a certain extent. Let X be a random variable with density $|\psi(r)|^2/\|\psi\|_{L^2}^2$. Using the Rayleigh–Ritz principle, finding the minimal nonzero eigenvalue of (8.1) can be reformulated as minimizing the Rayleigh quotient

$$\frac{\int_{\mathbb{R}^{3 \times n}} \overline{\psi(r)} (\mathcal{H}_R \psi)(r) dr}{\|\psi\|_{L^2}^2} = \mathbb{E} \left[\frac{(\mathcal{H}_R \psi)(X)}{\psi(X)} \right] \quad (8.3)$$

over all ψ ’s satisfying the Pauli principle, see [SO12]. Since this represents a minimization problem it can in principle be solved via a NN ansatz by generating training data distributed according to X using MCMC sampling³¹. Since the wave function ψ will be parametrized as a NN, the minimization of (8.3) will require the computation of the gradient of (8.3) with respect to the NN parameters (the method in [PSMF20] even requires second order derivatives) which, at first sight, might seem to require the computation of third order derivatives. However, due to the Hermitian structure of the Hamiltonian one does not need to compute the derivative of the Laplacian of ψ , see, for example, [HSN20, Equation (8)].

Compared to the other PDE problems we have discussed, an additional complication arises from the need to incorporate structural properties and invariances such as the Pauli principle. Furthermore, empirical evidence shows that it is also necessary to hard code the so-called *cusp conditions* which describe the asymptotic behavior of nearby electrons and electrons close to a nucleus into the NN architecture. A first attempt in this direction has been made in [HZE19] and significantly improved NN architectures have been developed in [HSN20, PSMF20, SGR⁺21] opening the possibility of accurate ab initio computations for previously intractable molecules. The mathematical properties of this exciting line of work remain largely unexplored. We briefly describe the main ideas behind the NN architecture of [HSN20, SGR⁺21]. Standard numerical approaches (notably the Multireference Hartree Fock Method, see [SO12]) use a low rank approach to minimize (8.3). Such a low rank approach would approximate ψ by sums of products of *one electron orbitals* $\prod_{i=1}^n \varphi_i(r_i)$ but clearly this does not satisfy the Pauli principle (8.2). In order to ensure the Pauli principle, one constructs so-called *Slater determinants* from one electron orbitals with equal spin. More precisely, suppose that the first n_+ electrons with coordinates r_1, \dots, r_{n_+} have positive spin and the last $n - n_+$ electrons have negative spin. Then any function of the form

$$\det \left((\varphi_i(r_j))_{i,j=1}^{n_+} \right) \cdot \det \left((\varphi_i(r_j))_{i,j=n_++1}^n \right) \quad (8.4)$$

satisfies (8.2) and is typically called a Slater determinant. While the Pauli principle establishes an (non-classical) interaction between electrons of equal spin, the so-called *exchange correlation*, electrons with opposite spins are uncorrelated in the representation (8.4). In particular, (8.4) ignores interactions between electrons that arise through Coulomb forces, implying that no nontrivial wavefunction can be accurately represented by a single Slater determinant. To capture physical interactions between different electrons, one needs to use sums of Slater determinants as an ansatz. However, it turns out that the number of such determinants that are needed to guarantee a given accuracy scales very badly with the system size n (to the

³¹Observe that for such sampling methods one can just use the unnormalized density $|\psi(r)|^2$ and thus avoid the computation of the normalization $\|\psi\|_{L^2}^2$.

best of our knowledge the best currently known approximation results are contained in [Yse10], where an n -independent error rate is shown, however the implicit constant in this rate depends at least exponentially on the system size n).

We would like to highlight the approach of [HSN20] whose main idea is to use NNs to incorporate interactions into Slater determinants of the form (8.4) using what is called the *backflow trick* [RMD⁺06]. The basic building blocks would now consist of functions of the form

$$\det \left((\varphi_i(r_j) \Psi_j(r, \theta_j))_{i,j=1}^{n_+} \right) \cdot \det \left((\varphi_i(r_j) \Psi_j(r, \theta_j))_{i,j=n_++1}^n \right), \quad (8.5)$$

where $\Psi_k(\cdot, \theta_k)$, $k \in [n]$, are NNs. If these are arbitrary NNs, it is easy to see that the Pauli principle (8.2) will not be satisfied. However, if we require the NNs to be symmetric, for example, in the sense that for $i, j, s \in [n_+]$ it holds that

$$P_{i,j} \Psi_k(\cdot, \theta_k) = \begin{cases} \Psi_k(\cdot, \theta_k), & \text{if } k \notin \{i, j\}, \\ \Psi_i(\cdot, \theta_i), & \text{if } k = j, \\ \Psi_j(\cdot, \theta_j), & \text{if } k = i, \end{cases} \quad (8.6)$$

and analogous conditions hold for $i, j, k \in [n] \setminus [n_+]$, the expression (8.5) does actually satisfy (8.2). The construction of such symmetric NNs can be achieved by using a modification of the so-called *SchNet Architecture* [SKS⁺17] which can be considered as a specific residual NN.

We describe a simplified construction which is inspired by [HZE19] and used in a slightly more complex form in [SGR⁺21]. We restrict ourselves to the case of positive spin (e.g., the first n_+ coordinates), the case of negative spin being handled in the same way. Let $\Upsilon(\cdot, \theta_{\text{emb}}^+)$ be a univariate NN (with possibly multivariate output) and denote

$$\text{Emb}_k(r, \theta_{\text{emb}}^+) := \sum_{i=1}^{n_+} \Upsilon(\|r_k - r_i\|_2, \theta_{\text{emb}}^+), \quad k \in [n_+],$$

the k -th *embedding layer*. For $k \in [n_+]$, we can now define

$$\Psi_k(r, \theta_k) = \Psi_k(r, (\theta_{k,\text{fc}}, \theta_{\text{emb}}^+)) = \Gamma_k((\text{Emb}_k(r, \theta_{\text{emb}}^+), (r_{n_++1}, \dots, r_n)), \theta_{k,\text{fc}}),$$

where $\Gamma_k(\cdot, \theta_{k,\text{fc}})$ denotes a standard FC NN with input dimension equal to the output dimension of Ψ^+ plus the dimension of negative spin electrons. The networks Ψ_k , $k \in [n] \setminus [n_+]$, are defined analogously using different parameters θ_{emb}^- for the embeddings. It is straightforward to check that the NNs Ψ_k , $k \in [n]$, satisfy (8.6) so that the backflow determinants (8.5) satisfy the Pauli principle (8.2).

In [HSN20] the backflow determinants (8.5) are further augmented by a multiplicative correction term, the so-called *Jastrow factor* which is also represented by a specific symmetric NN, as well as a correction term that ensures the validity of the cusp conditions. The results of [HSN20] show that this ansatz (namely using linear combinations of backflow determinants (8.5) instead of plain Slater determinants (8.4)) is vastly more efficient in terms of number of determinants needed to obtain chemical accuracy. The full architecture provides a general purpose NN architecture to represent complicated wave functions. A distinct advantage of this approach is that some parameters (for example, embedding layers) may be shared across different nuclear geometries $R \in \mathbb{R}^{3 \times p}$ which allows for the efficient computation of potential energy surfaces [SGR⁺21], see Figure 8.1. Finally, we would like to highlight the customized NN design that incorporates physical invariances, domain knowledge (for example, in the form of cusp conditions), and existing numerical methods, all of which are required for the method to reach its full potential.

Acknowledgment

The research of JB was supported by the Austrian Science Fund (FWF) under grant I3403-N32. GK acknowledges support from DFG-SPP 1798 Grants KU 1446/21-2 and KU 1446/27-2, DFG-SFB/TR 109 Grant C09, BMBF Grant MaGrido, and NSF-Simons Foundation Grant SIMONS 81420. The authors would

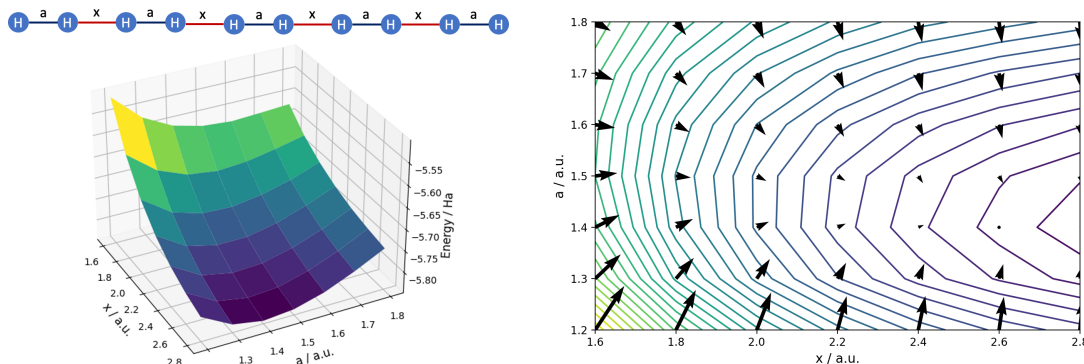


Figure 8.1: By sharing layers across different nuclear geometries one can efficiently compute different geometries in one single training step [SGR⁺21]. Left: Potential energy surface of H10 chain computed by the deep-learning-based algorithm from [SGR⁺21]. The lowest energy is achieved when pairs of H atoms enter into a covalent bond to form five H2 molecules. Right: The method of [SGR⁺21] is capable of accurately computing forces between nuclei which allows for molecular dynamics simulations from first principles.

like to thank Héctor Andrade Loarca, Dennis Elbrächter, Adalbert Fono, Pavol Harar, Lukas Liehr, Duc Anh Nguyen, Mariia Seleznova, and Frieder Simon for their helpful feedback on an early version of this article. In particular, Dennis Elbrächter was providing help for several theoretical results.

References

- [AAČ13] Antonio Auffinger, Gérard Ben Arous, and Jiří Černý, *Random matrices and complexity of spin glasses*, Communications on Pure and Applied Mathematics **66** (2013), no. 2, 165–201.
- [AB99] Martin Anthony and Peter L Bartlett, *Neural network learning: Theoretical foundations*, Cambridge University Press, 1999.
- [ACGH19] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu, *A convergence analysis of gradient descent for deep linear neural networks*, International Conference on Learning Representations, 2019.
- [ACH18] Sanjeev Arora, Nadav Cohen, and Elad Hazan, *On the optimization of deep networks: Implicit acceleration by overparameterization*, International Conference on Machine Learning, 2018, pp. 372–389.
- [AD20] Ben Adcock and Nick Dexter, *The gap between theory and practice in function approximation with deep neural networks*, 2020, arXiv preprint arXiv:2001.07523.
- [ADH⁺19] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang, *On exact computation with an infinitely wide neural net*, Advances in Neural Information Processing Systems, 2019, pp. 8139–8148.
- [AGNZ18] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang, *Stronger generalization bounds for deep nets via a compression approach*, International Conference on Machine Learning, 2018, pp. 254–263.
- [AHNB⁺20] Yasmine S Al-Hamdani, Péter R Nagy, Dennis Barton, Mihály Kállay, Jan Gerit Brandenburg, and Alexandre Tkatchenko, *Interactions between large molecules: Puzzle for reference quantum-mechanical methods*, 2020, arXiv preprint arXiv:2009.08927.

- [AHS85] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 1, 147–169.
- [AHW96] Peter Auer, Mark Herbster, and Manfred K Warmuth, *Exponentially many local minima for single neurons*, Advances in Neural Information Processing Systems, 1996, p. 316–322.
- [AMÖS19] Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb, *Solving inverse problems using data-driven models*, Acta Numerica **28** (2019), 1–174.
- [AÖ17] Jonas Adler and Ozan Öktem, *Solving ill-posed inverse problems using iterative deep neural networks*, Inverse Problems **33** (2017), no. 12, 124007.
- [AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song, *A convergence theory for deep learning via over-parameterization*, International Conference on Machine Learning, 2019, pp. 242–252.
- [Bar92] Andrew R Barron, *Neural net approximation*, Yale Workshop on Adaptive and Learning Systems, vol. 1, 1992, pp. 69–72.
- [Bar93] ———, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Transactions on Information Theory **39** (1993), no. 3, 930–945.
- [Bar98] Peter L Bartlett, *The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network*, IEEE Transactions on Information Theory **44** (1998), no. 2, 525–536.
- [BBC17] Alfred Bourelly, John Patrick Boueri, and Krzysztof Choromonski, *Sparse neural networks topologies*, 2017, arXiv preprint arXiv:1706.05683.
- [BBC⁺19] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, and Chris Hesse, *Dota 2 with large scale deep reinforcement learning*, 2019, arXiv preprint arXiv:1912.06680.
- [BBG⁺18] Christian Beck, Sebastian Becker, Philipp Grohs, Nor Jaafari, and Arnulf Jentzen, *Solving stochastic differential equations and Kolmogorov equations by means of deep learning*, 2018, arXiv preprint arXiv:1806.00421.
- [BBL03] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi, *Introduction to statistical learning theory*, Summer School on Machine Learning, 2003, pp. 169–207.
- [BBL⁺17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst, *Geometric deep learning: going beyond euclidean data*, IEEE Signal Processing Magazine **34** (2017), no. 4, 18–42.
- [BBM05] Peter L Bartlett, Olivier Bousquet, and Shahar Mendelson, *Local Rademacher complexities*, The Annals of Statistics **33** (2005), no. 4, 1497–1537.
- [BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, *Neural machine translation by jointly learning to align and translate*, International Conference on Learning Representations, 2015.
- [BDG20] Julius Berner, Markus Dablander, and Philipp Grohs, *Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning*, Advances in Neural Information Processing Systems, 2020, pp. 16615–16627.
- [BE02] Olivier Bousquet and André Elisseeff, *Stability and generalization*, Journal of Machine Learning Research **2** (2002), no. Mar, 499–526.

- [BEG19] Julius Berner, Dennis Elbrächter, and Philipp Grohs, *How degenerate is the parametrization of neural networks with the ReLU activation function?*, Advances in Neural Information Processing Systems, 2019, pp. 7790–7801.
- [Bel52] Richard Bellman, *On the theory of dynamic programming*, Proceedings of the National Academy of Sciences **38** (1952), no. 8, 716.
- [BF19] Jan Bohn and Michael Feischl, *Recurrent neural networks as optimal mesh refinement strategies*, 2019, arXiv preprint arXiv:1909.04275.
- [BFT17] Peter L Bartlett, Dylan J Foster, and Matus Telgarsky, *Spectrally-normalized margin bounds for neural networks*, Advances in Neural Information Processing Systems, 2017, pp. 6240–6249.
- [BGJ20] Julius Berner, Philipp Grohs, and Arnulf Jentzen, *Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations*, SIAM Journal on Mathematics of Data Science **2** (2020), no. 3, 631–657.
- [BH89] Eric B Baum and David Haussler, *What size net gives valid generalization?*, Neural Computation **1** (1989), no. 1, 151–160.
- [BHLM19] 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** (2019), 63–1.
- [BHMM19] 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** (2019), no. 32, 15849–15854.
- [BHX20] Mikhail Belkin, Daniel Hsu, and Ji Xu, *Two models of double descent for weak features*, SIAM Journal on Mathematics of Data Science **2** (2020), no. 4, 1167–1180.
- [BK18] Andrew R Barron and Jason M Klusowski, *Approximation and estimation for high-dimensional deep learning networks*, 2018, arXiv preprint arXiv:1809.03090.
- [BLLT20] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler, *Benign overfitting in linear regression*, Proceedings of the National Academy of Sciences **117** (2020), no. 48, 30063–30070.
- [BMM98] Peter L Bartlett, Vitaly Maiorov, and Ron Meir, *Almost linear VC-dimension bounds for piecewise polynomial networks*, Neural Computation **10** (1998), no. 8, 2159–2173.
- [BMM18] Mikhail Belkin, Siyuan Ma, and Soumik Mandal, *To understand deep learning we need to understand kernel learning*, International Conference on Machine Learning, 2018, pp. 541–549.
- [BMR⁺20] 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*, Advances in Neural Information Processing Systems, 2020, pp. 1877–1901.
- [BR89] Avrim Blum and Ronald L Rivest, *Training a 3-node neural network is NP-complete*, Advances in Neural Information Processing Systems, 1989, pp. 494–501.

- [BRT19] Mikhail Belkin, Alexander Rakhlin, and Alexandre B Tsybakov, *Does data interpolation contradict statistical optimality?*, International Conference on Artificial Intelligence and Statistics, 2019, pp. 1611–1619.
- [BSW14] Pierre Baldi, Peter Sadowski, and Daniel Whiteson, *Searching for exotic particles in high-energy physics with deep learning*, Nature Communications **5** (2014), no. 1, 1–9.
- [BZAJ20] Riccardo Barbano, Chen Zhang, Simon Arridge, and Bangti Jin, *Quantifying model uncertainty in inverse problems via bayesian deep gradient descent*, 2020, arXiv preprint arXiv:2007.09971.
- [Can98] Emmanuel J Candès, *Ridgelets: Theory and applications*, Ph.D. thesis, Stanford University, 1998.
- [CB20] Lenaïc Chizat and Francis Bach, *Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss*, Conference on Learning Theory, 2020, pp. 1305–1338.
- [CHM⁺15] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun, *The loss surfaces of multilayer networks*, International Conference on Artificial Intelligence and Statistics, 2015, pp. 192–204.
- [CJLZ19] Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao, *Efficient approximation of deep ReLU networks for functions on low dimensional manifolds*, Advances in Neural Information Processing Systems, 2019, pp. 8174–8184.
- [CK20] Alexander Cloninger and Timo Klock, *ReLU nets adapt to intrinsic dimensionality beyond the target domain*, 2020, arXiv preprint arXiv:2008.02545.
- [CKP12] Peter G Casazza, Gitta Kutyniok, and Friedrich Philipp, *Introduction to finite frame theory*, Finite Frames: Theory and Applications, Birkhäuser Boston, 2012, pp. 1–53.
- [CL19] Wojciech Czaja and Weilin Li, *Analysis of time-frequency scattering transforms*, Applied and Computational Harmonic Analysis **47** (2019), no. 1, 149–171.
- [CLA15] Anna Choromanska, Yann LeCun, and Gérard Ben Arous, *Open problem: The landscape of the loss surfaces of multilayer networks*, Conference on Learning Theory, 2015, pp. 1756–1760.
- [CLM94] Charles K Chui, Xin Li, and Hrushikesh N Mhaskar, *Neural networks for localized approximation*, Mathematics of Computation **63** (1994), no. 208, 607–623.
- [CM18] Charles K Chui and Hrushikesh N Mhaskar, *Deep nets for local manifold learning*, Frontiers in Applied Mathematics and Statistics **4** (2018), 12.
- [CMBK20] Lin Chen, Yifei Min, Mikhail Belkin, and Amin Karbasi, *Multiple descent: Design your own generalization curve*, 2020, arXiv preprint arXiv:2008.01036.
- [COB19] Lenaïc Chizat, Edouard Oyallon, and Francis Bach, *On lazy training in differentiable programming*, Advances in Neural Information Processing Systems, 2019, pp. 2937–2947.
- [CPV20] Andrei Caragea, Philipp Petersen, and Felix Voigtlaender, *Neural network approximation and estimation of classifiers with classification boundary in a Barron class*, 2020, arXiv preprint arXiv:2011.09363.
- [CS02] Felipe Cucker and Steve Smale, *On the mathematical foundations of learning*, Bulletin of the American Mathematical Society **39** (2002), no. 1, 1–49.
- [CvMG⁺14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, *Learning phrase representations using rnn encoder–decoder for statistical machine translation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734.

- [Cyb89] George Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals and Systems **2** (1989), no. 4, 303–314.
- [CZ07] Felipe Cucker and Ding-Xuan Zhou, *Learning theory: an approximation theory viewpoint*, vol. 24, Cambridge University Press, 2007.
- [DDS⁺09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, *Imagenet: A large-scale hierarchical image database*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
- [DeV98] Ronald A DeVore, *Nonlinear approximation*, Acta Numerica **7** (1998), 51–150.
- [DGL96] Luc Devroye, László Györfi, and Gábor Lugosi, *A probabilistic theory of pattern recognition*, Springer, 1996.
- [DHL18] Simon S Du, Wei Hu, and Jason D Lee, *Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced*, Advances in Neural Information Processing Systems, 2018, pp. 384–395.
- [DHP20] Ronald DeVore, Boris Hanin, and Guergana Petrova, *Neural network approximation*, 2020, arXiv preprint arXiv:2012.14501.
- [Dir29] Paul Adrien Maurice Dirac, *Quantum mechanics of many-electron systems*, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character **123** (1929), no. 792, 714–733.
- [DKMB20] Sören Dittmer, Tobias Kluth, Peter Maass, and Daniel Otero Baguer, *Regularization by architecture: A deep prior approach for inverse problems*, Journal of Mathematical Imaging and Vision **62** (2020), no. 3, 456–470.
- [DLL⁺19] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai, *Gradient descent finds global minima of deep neural networks*, International Conference on Machine Learning, 2019, pp. 1675–1685.
- [Don69] William F Donoghue, *Distributions and fourier transforms*, Pure and Applied Mathematics, Academic Press, 1969.
- [DPG⁺14] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio, *Identifying and attacking the saddle point problem in high-dimensional non-convex optimization*, Advances in Neural Information Processing Systems, 2014, pp. 2933–2941.
- [DR17] Gintare Karolina Dziugaite and Daniel M Roy, *Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data*, Conference on Uncertainty in Artificial Intelligence, 2017.
- [Dre62] Stuart Dreyfus, *The numerical solution of variational problems*, Journal of Mathematical Analysis and Applications **5** (1962), no. 1, 30–45.
- [Dud67] Richard M Dudley, *The sizes of compact subsets of hilbert space and continuity of Gaussian processes*, Journal of Functional Analysis **1** (1967), no. 3, 290–330.
- [Dud14] ———, *Uniform central limit theorems*, vol. 142, Cambridge University Press, 2014.
- [DZPS18] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh, *Gradient descent provably optimizes over-parameterized neural networks*, International Conference on Learning Representations, 2018.
- [E17] Weinan E, *A proposal on machine learning via dynamical systems*, Communications in Mathematics and Statistics **5** (2017), no. 1, 1–11.

- [EGJS18] Dennis Elbrächter, Philipp Grohs, Arnulf Jentzen, and Christoph Schwab, *DNN expression rate analysis of high-dimensional PDEs: Application to option pricing*, 2018, arXiv preprint arXiv:1809.07669.
- [EHJK19] Weinan E, Martin Hutzenthaler, Arnulf Jentzen, and Thomas Kruse, *On multilevel picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations*, Journal of Scientific Computing **79** (2019), no. 3, 1534–1571.
- [EHL19] Weinan E, Jiequn Han, and Qianxiao Li, *A mean-field optimal control formulation of deep learning*, Research in the Mathematical Sciences **6** (2019), no. 1, 1–41.
- [Elm90] Jeffrey L Elman, *Finding structure in time*, Cognitive Science **14** (1990), no. 2, 179–211.
- [EMW19a] Weinan E, Chao Ma, and Lei Wu, *Barron spaces and the compositional function spaces for neural network models*, 2019, arXiv preprint arXiv:1906.08039.
- [EMW19b] ———, *A priori estimates of the population risk for two-layer neural networks*, Communications in Mathematical Sciences **17** (2019), no. 5, 1407–1425.
- [EMWW20] Weinan E, Chao Ma, Stephan Wojtowytsch, and Lei Wu, *Towards a mathematical understanding of neural network-based machine learning: what we know and what we don’t*, 2020, arXiv preprint arXiv:2009.10713.
- [EPGB19] Dennis Elbrächter, Dmytro Perekrestenko, Philipp Grohs, and Helmut Bölcskei, *Deep neural network approximation theory*, 2019, arXiv preprint arXiv:1901.02220.
- [ES16] Ronen Eldan and Ohad Shamir, *The power of depth for feedforward neural networks*, Conference on Learning Theory, vol. 49, 2016, pp. 907–940.
- [EW20a] Weinan E and Stephan Wojtowytsch, *On the Banach spaces associated with multi-layer ReLU networks: Function representation, approximation theory and gradient descent dynamics*, 2020, arXiv preprint arXiv:2007.15623.
- [EW20b] ———, *A priori estimates for classification problems using neural networks*, 2020, arXiv preprint arXiv:2009.13500.
- [EW20c] ———, *Representation formulas and pointwise properties for Barron functions*, 2020, arXiv preprint arXiv:2006.05982.
- [EY18] Weinan E and Bing Yu, *The deep ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics **6** (2018), no. 1, 1–12.
- [FB17] Daniel C Freeman and Joan Bruna, *Topology and geometry of half-rectified network optimization*, International Conference on Learning Representations, 2017.
- [FC18] Jonathan Frankle and Michael Carbin, *The lottery ticket hypothesis: Finding sparse, trainable neural networks*, International Conference on Learning Representations, 2018.
- [FHH⁺17] Felix A Faber, Luke Hutchison, Bing Huang, Justin Gilmer, Samuel S Schoenholz, George E Dahl, Oriol Vinyals, Steven Kearnes, Patrick F Riley, and O Anatole Von Lilienfeld, *Prediction errors of molecular machine learning models lower than hybrid DFT error*, Journal of Chemical Theory and Computation **13** (2017), no. 11, 5255–5264.
- [Fun89] Ken-Ichi Funahashi, *On the approximate realization of continuous mappings by neural networks*, Neural Networks **2** (1989), no. 3, 183–192.

- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016.
- [Gér17] Aurelien Géron, *Hands-on machine learning with scikit-learn and tensorflow: Concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, 2017.
- [GH21] Philipp Grohs and Lukas Herrmann, *Deep neural network approximation for high-dimensional parabolic Hamilton-Jacobi-Bellman equations*, 2021, arXiv preprint arXiv:2103.05744.
- [GHJVW20] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe Von Wurstemberger, *A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations*, *Memoirs of the American Mathematical Society* (2020).
- [GHJY15] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan, *Escaping from saddle points—online stochastic gradient for tensor decomposition*, *Conference on Learning Theory*, 2015, pp. 797–842.
- [GJS⁺20] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d’Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart, *Scaling description of generalization with number of parameters in deep learning*, *Journal of Statistical Mechanics: Theory and Experiment* (2020), no. 2, 023401.
- [GKP20] Ingo Gühring, Gitta Kutyniok, and Philipp Petersen, *Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms*, *Analysis and Applications* **18** (2020), no. 05, 803–859.
- [GKR20] Philipp Grohs, Sarah Koppensteiner, and Martin Rathmair, *Phase retrieval: Uniqueness and stability*, *SIAM Review* **62** (2020), no. 2, 301–350.
- [GL13] Saeed Ghadimi and Guanghui Lan, *Stochastic first-and zeroth-order methods for nonconvex stochastic programming*, *SIAM Journal on Optimization* **23** (2013), no. 4, 2341–2368.
- [GLSS18a] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nathan Srebro, *Characterizing implicit bias in terms of optimization geometry*, *International Conference on Machine Learning*, 2018, pp. 1832–1841.
- [GLSS18b] ———, *Implicit bias of gradient descent on linear convolutional networks*, *Advances in Neural Information Processing Systems*, 2018, pp. 9461–9471.
- [GMMM21] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari, *Linearized two-layers neural networks in high dimension*, *The Annals of Statistics* **49** (2021), no. 2, 1029–1054.
- [GOW19] Davis Gilton, Greg Ongie, and Rebecca Willett, *Neumann networks for linear inverse problems in imaging*, *IEEE Transactions on Computational Imaging* **6** (2019), 328–343.
- [GPAM⁺14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [GRK20] Ingo Gühring, Mones Raslan, and Gitta Kutyniok, *Expressivity of deep neural networks*, 2020, arXiv preprint arXiv:2007.04759.
- [GRS18] Noah Golowich, Alexander Rakhlin, and Ohad Shamir, *Size-independent sample complexity of neural networks*, *Conference On Learning Theory*, 2018, pp. 297–299.
- [GS20] Lukas Gonon and Christoph Schwab, *Deep ReLU network expression rates for option prices in high-dimensional, exponential Lévy models*, 2020, ETH Zurich SAM Research Report.

- [GV21] Philipp Grohs and Felix Voigtlaender, *Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces*, 2021, arXiv preprint arXiv:2104.02746.
- [GW08] Andreas Griewank and Andrea Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.
- [GZ84] Evarist Giné and Joel Zinn, *Some limit theorems for empirical processes*, The Annals of Probability (1984), 929–989.
- [Han19] Boris Hanin, *Universal function approximation by deep neural nets with bounded width and ReLU activations*, Mathematics **7** (2019), no. 10, 992.
- [Hau95] David Haussler, *Sphere packing numbers for subsets of the boolean n -cube with bounded vapnik-chervonenkis dimension*, Journal of Combinatorial Theory, Series A **2** (1995), no. 69, 217–232.
- [HH19] Catherine F Higham and Desmond J Higham, *Deep learning: An introduction for applied mathematicians*, SIAM Review **61** (2019), no. 4, 860–891.
- [HHJ15] Martin Hairer, Martin Hutzenthaler, and Arnulf Jentzen, *Loss of regularity for Kolmogorov equations*, The Annals of Probability **43** (2015), no. 2, 468–527.
- [HJE18] Jiequn Han, Arnulf Jentzen, and Weinan E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences **115** (2018), no. 34, 8505–8510.
- [HJKN20] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, and Tuan Anh Nguyen, *A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations*, SN Partial Differential Equations and Applications **1** (2020), no. 2, 1–34.
- [HLXZ20] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng, *ReLU deep neural networks and linear finite elements*, Journal of Computational Mathematics **38** (2020), no. 3, 502–527.
- [HMD16] Song Han, Huizi Mao, and William J Dally, *Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding*, International Conference on Learning Representations, 2016.
- [HMRT19] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani, *Surprises in high-dimensional ridgeless least squares interpolation*, 2019, arXiv preprint arXiv:1903.08560.
- [Hoe63] Wassily Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association **58** (1963), no. 301, 13–30.
- [Hop82] John J Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences **79** (1982), no. 8, 2554–2558.
- [HR19] Boris Hanin and David Rolnick, *Deep ReLU networks have surprisingly few activation patterns*, Advances in Neural Information Processing Systems, 2019, pp. 359–368.
- [HRS16] Moritz Hardt, Ben Recht, and Yoram Singer, *Train faster, generalize better: Stability of stochastic gradient descent*, International Conference on Machine Learning, 2016, pp. 1225–1234.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural Computation **9** (1997), no. 8, 1735–1780.
- [HS17] Boris Hanin and Mark Sellke, *Approximating continuous functions by ReLU nets of minimal width*, 2017, arXiv preprint arXiv:1710.11278.

- [HSL⁺16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger, *Deep networks with stochastic depth*, European Conference on Computer Vision, 2016, pp. 646–661.
- [HSN20] Jan Hermann, Zeno Schätzle, and Frank Noé, *Deep-neural-network solution of the electronic Schrödinger equation*, Nature Chemistry **12** (2020), no. 10, 891–897.
- [HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, *Multilayer feedforward networks are universal approximators*, Neural Networks **2** (1989), no. 5, 359–366.
- [HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, Springer Series in Statistics, Springer, 2001.
- [HV17] Benjamin D Haeffele and René Vidal, *Global optimality in neural network training*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7331–7339.
- [HvdG19] Peter Hinz and Sara van de Geer, *A framework for the construction of upper bounds on the number of affine linear regions of ReLU feed-forward neural networks*, IEEE Transactions on Information Theory **65** (2019), 7304–7324.
- [HZ94] Geoffrey E Hinton and Richard S Zemel, *Autoencoders, minimum description length, and helmholtz free energy*, Advances in Neural Information Processing Systems **6** (1994), 3–10.
- [HZE19] Jiequn Han, Linfeng Zhang, and Weinan E, *Solving many-electron Schrödinger equation using deep neural networks*, Journal of Computational Physics **399** (2019), 108929.
- [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, Proceedings of IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
- [HZRS16] ———, *Deep residual learning for image recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
- [IS15] Sergey Ioffe and Christian Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, International Conference on Machine Learning, 2015, pp. 448–456.
- [JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler, *Neural tangent kernel: Convergence and generalization in neural networks*, Advances in Neural Information Processing Systems, 2018, pp. 8571–8580.
- [JKMB19] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio, *Predicting the generalization gap in deep networks with margin distributions*, International Conference on Learning Representations, 2019.
- [JKNvW20] Arnulf Jentzen, Benno Kuckuck, Ariel Neufeld, and Philippe von Wurstemberger, *Strong error analysis for stochastic gradient descent optimization algorithms*, IMA Journal of Numerical Analysis **41** (2020), no. 1, 455–492.
- [JMFU17] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser, *Deep convolutional neural network for inverse problems in imaging*, IEEE Transactions on Image Processing **26** (2017), no. 9, 4509–4522.
- [JMS17] Bangti Jin, Peter Maaß, and Otmar Scherzer, *Sparsity regularization in inverse problems*, Inverse Problems **33** (2017), no. 6, 060301.
- [JNM⁺20] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio, *Fantastic generalization measures and where to find them*, International Conference on Learning Representations, 2020.

- [Jor90] Michael I Jordan, *Attractor dynamics and parallelism in a connectionist sequential machine*, Artificial neural networks: concept learning, IEEE Press, 1990, pp. 112–127.
- [JT19a] Ziwei Ji and Matus Telgarsky, *Gradient descent aligns the layers of deep linear networks*, International Conference on Learning Representations, 2019.
- [JT19b] ———, *A refined primal-dual analysis of the implicit bias*, 2019, arXiv preprint arXiv:1906.04540.
- [JT20] ———, *Directional convergence and alignment in deep learning*, Advances in Neural Information Processing Systems, 2020, pp. 17176–17186.
- [Jud90] Stephen J Judd, *Neural network design and the complexity of learning*, MIT Press, 1990.
- [Kel60] Henry J Kelley, *Gradient theory of optimal flight paths*, Ars Journal **30** (1960), no. 10, 947–954.
- [KH09] Alex Krizhevsky and Geoffrey Hinton, *Learning multiple layers of features from tiny images*, Tech. report, University of Toronto, 2009.
- [KL18] Sham M Kakade and Jason D Lee, *Provably correct automatic subdifferentiation for qualified programs*, Advances in Neural Information Processing Systems, 2018, pp. 7125–7135.
- [KL20] Patrick Kidger and Terry Lyons, *Universal approximation with deep narrow networks*, Conference on Learning Theory, 2020, pp. 2306–2327.
- [KM97] Marek Karpinski and Angus Macintyre, *Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks*, Journal of Computer and System Sciences **54** (1997), no. 1, 169–176.
- [KMN⁺17] 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*, International Conference on Learning Representations, 2017.
- [KPRS19] Gitta Kutyniok, Philipp Petersen, Mones Raslan, and Reinhold Schneider, *A theoretical analysis of deep neural networks and parametric PDEs*, 2019, arXiv preprint arXiv:1904.00377.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
- [KW52] Jack Kiefer and Jacob Wolfowitz, *Stochastic estimation of the maximum of a regression function*, The Annals of Mathematical Statistics **23** (1952), no. 3, 462–466.
- [LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.
- [LBD⁺89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural Computation **1** (1989), no. 4, 541–551.
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444.
- [LBN⁺18] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein, *Deep neural networks as Gaussian processes*, International Conference on Learning Representations, 2018.
- [LC19] Guillaume Lample and François Charton, *Deep learning for symbolic mathematics*, International Conference on Learning Representations, 2019.

- [LDS89] Yann LeCun, John S Denker, and Sara A Solla, *Optimal brain damage*, Advances in Neural Information Processing Systems, 1989, pp. 598–605.
- [Lew43] Kurt Lewin, *Psychology and the process of group living*, The Journal of Social Psychology **17** (1943), no. 1, 113–131.
- [Li21] Weilin Li, *Generalization error of minimum weighted norm and kernel interpolation*, SIAM Journal on Mathematics of Data Science **3** (2021), no. 1, 414–438.
- [Lin70] Seppo Linnainmaa, *Alogritmin kumulatiivinen pyöristysvirhe yksittäisten pyöristysvirheiden Taylor-kehitemänä*, Master’s thesis, University of Helsinki, 1970.
- [LL18] Yuanzhi Li and Yingyu Liang, *Learning overparameterized neural networks via stochastic gradient descent on structured data*, Advances in Neural Information Processing Systems, 2018, pp. 8157–8166.
- [LL19] Kaifeng Lyu and Jian Li, *Gradient descent maximizes the margin of homogeneous neural networks*, International Conference on Learning Representations, 2019.
- [LLPS93] 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** (1993), no. 6, 861–867.
- [LLS19] Qianxiao Li, Ting Lin, and Zuowei Shen, *Deep learning via dynamical systems: An approximation perspective*, 2019, arXiv preprint arXiv:1912.10382.
- [LML⁺20] Yiping Lu, Chao Ma, Yulong Lu, Jianfeng Lu, and Lexing Ying, *A mean field analysis of deep ResNet and beyond: Towards provably optimization via overparameterization from depth*, International Conference on Machine Learning, 2020, pp. 6426–6436.
- [LÖS18] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb, *Adversarial regularizers in inverse problems*, Advances in Neural Information Processing Systems, 2018, pp. 8507–8516.
- [LP21] Fabian Laakmann and Philipp Petersen, *Efficient approximation of solutions of parametric linear transport equations by ReLU DNNs*, Advances in Computational Mathematics **47** (2021), no. 1, 1–32.
- [LPRS19] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes, *Fisher–Rao metric, geometry, and complexity of neural networks*, International Conference on Artificial Intelligence and Statistics, 2019, pp. 888–896.
- [LR20] Tengyuan Liang and Alexander Rakhlin, *Just interpolate: Kernel “ridgeless” regression can generalize*, The Annals of Statistics **48** (2020), no. 3, 1329–1347.
- [LRZ20] Tengyuan Liang, Alexander Rakhlin, and Xiyu Zhai, *On the multiple descent of minimum-norm interpolants and restricted lower isometry of kernels*, Conference on Learning Theory, 2020, pp. 2683–2711.
- [LS17] Shiyu Liang and R Srikant, *Why deep neural networks for function approximation?*, International Conference on Learning Representations, 2017.
- [LSAH20] Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier, *NETT: Solving inverse problems with deep neural networks*, Inverse Problems **36** (2020), no. 6, 065005.
- [LSJR16] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht, *Gradient descent only converges to minimizers*, Conference on Learning Theory, 2016, pp. 1246–1257.

- [LT91] Michel Ledoux and Michel Talagrand, *Probability in Banach spaces: Isoperimetry and processes*, vol. 23, Springer Science & Business Media, 1991.
- [LTY19] Bo Li, Shanshan Tang, and Haijun Yu, *Better approximations of high dimensional smooth functions by deep neural networks with rectified power units*, Communications in Computational Physics **27** (2019), no. 2, 379–411.
- [LXS⁺20] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington, *Wide neural networks of any depth evolve as linear models under gradient descent*, Journal of Statistical Mechanics: Theory and Experiment **2020** (2020), no. 12, 124002.
- [Mal12] Stéphane Mallat, *Group invariant scattering*, Communications on Pure and Applied Mathematics **65** (2012), no. 10, 1331–1398.
- [Mal16] ———, *Understanding deep convolutional networks*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374** (2016), no. 2065, 20150203.
- [MAV18] Poorya Mianjy, Raman Arora, and Rene Vidal, *On the implicit bias of dropout*, International Conference on Machine Learning, 2018, pp. 3540–3548.
- [McA99] David A McAllester, *Pac-bayesian model averaging*, Conference on Learning Theory, 1999, pp. 164–170.
- [McD89] Colin McDiarmid, *On the method of bounded differences*, Surveys in Combinatorics **141** (1989), no. 1, 148–188.
- [Men14] Shahar Mendelson, *Learning without concentration*, Conference on Learning Theory, 2014, pp. 25–39.
- [Mha96] Hrushikesh N Mhaskar, *Neural networks for optimal approximation of smooth and analytic functions*, Neural Computation **8** (1996), no. 1, 164–177.
- [MHR⁺18] Alexander G de G Matthews, Jiri Hron, Mark Rowland, Richard E Turner, and Zoubin Ghahramani, *Gaussian process behaviour in wide deep neural networks*, International Conference on Learning Representations, 2018.
- [MKS⁺13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, *Playing atari with deep reinforcement learning*, 2013, arXiv preprint arXiv:1312.5602.
- [MLE21] Vishal Monga, Yuelong Li, and Yonina C Eldar, *Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing*, IEEE Signal Processing Magazine **38** (2021), no. 2, 18–44.
- [MM19] Song Mei and Andrea Montanari, *The generalization error of random features regression: Precise asymptotics and double descent curve*, 2019, arXiv preprint arXiv:1908.05355.
- [MOPS20] Carlo Marcati, Joost Opschoor, Philipp Petersen, and Christoph Schwab, *Exponential ReLU neural network approximation rates for point and edge singularities*, 2020, ETH Zurich SAM Research Report.
- [MP43] Warren S McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*, The Bulletin of Mathematical Biophysics **5** (1943), no. 4, 115–133.
- [MP69] Marvin Minsky and Seymour A Papert, *Perceptrons*, MIT Press, 1969.

- [MP99] Vitaly Maiorov and Allan Pinkus, *Lower bounds for approximation by MLP neural networks*, Neurocomputing **25** (1999), no. 1-3, 81–91.
- [MPCB14] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio, *On the number of linear regions of deep neural networks*, Advances in Neural Information Processing Systems, 2014, pp. 2924–2932.
- [MSL⁺15] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik, *Deep neural nets as a method for quantitative structure–activity relationships*, Journal of chemical information and modeling **55** (2015), no. 2, 263–274.
- [MV03] Shahar Mendelson and Roman Vershynin, *Entropy and the combinatorial dimension*, Inventiones mathematicae **152** (2003), no. 1, 37–55.
- [MVSS20] Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai, *Harmless interpolation of noisy data in regression*, IEEE Journal on Selected Areas in Information Theory **1** (2020), no. 1, 67–83.
- [MZ20] Andrea Montanari and Yiqiao Zhong, *The interpolation phase transition in neural networks: Memorization and generalization under lazy training*, 2020, arXiv preprint arXiv:2007.12826.
- [NBMS17] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro, *Exploring generalization in deep learning*, Advances in Neural Information Processing Systems, 2017, pp. 5947–5956.
- [NBS18] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro, *A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks*, International Conference on Learning Representations, 2018.
- [NH17] Quynh Nguyen and Matthias Hein, *The loss surface of deep and wide neural networks*, International Conference on Machine Learning, 2017, pp. 2603–2612.
- [NJLS09] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro, *Robust stochastic approximation approach to stochastic programming*, SIAM Journal on Optimization **19** (2009), no. 4, 1574–1609.
- [NK19] Vaishnavh Nagarajan and J Zico Kolter, *Uniform convergence may be unable to explain generalization in deep learning*, Advances in Neural Information Processing Systems, 2019, pp. 11615–11626.
- [NKB⁺20] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever, *Deep double descent: Where bigger models and more data hurt*, International Conference on Learning Representations, 2020.
- [NLG⁺19] Mor Shpigel Nacson, Jason D Lee, Suriya Gunasekar, Pedro Henrique Pamlona Savarese, Nathan Srebro, and Daniel Soudry, *Convergence of gradient descent on separable data*, International Conference on Artificial Intelligence and Statistics, 2019, pp. 3420–3428.
- [NTS14] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro, *In search of the real inductive bias: On the role of implicit regularization in deep learning*, 2014, arXiv preprint arXiv:1412.6614.
- [NTS15] ———, *Norm-based capacity control in neural networks*, Conference on Learning Theory, 2015, pp. 1376–1401.
- [NW09] Erich Novak and Henryk Woźniakowski, *Approximation of infinitely differentiable multivariate functions is intractable*, Journal of Complexity **25** (2009), no. 4, 398–404.

- [NY83] Arkadi Semenovich Nemirovsky and David Borisovich Yudin, *Problem complexity and method efficiency in optimization*, Wiley-Interscience Series in Discrete Mathematics, Wiley, 1983.
- [NZGK21] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang, *Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman Equations*, SIAM Journal on Scientific Computing **43** (2021), no. 2, A1221–A1247.
- [OF96] Bruno A Olshausen and David J Field, *Sparse coding of natural images produces localized, oriented, bandpass receptive fields*, Nature **381** (1996), no. 60, 609.
- [OM98] Genevieve B Orr and Klaus-Robert Müller, *Neural networks: tricks of the trade*, Springer, 1998.
- [OPS20] Joost Opschoor, Philipp Petersen, and Christoph Schwab, *Deep ReLU networks and high-order finite element methods*, Analysis and Applications (2020), no. 0, 1–56.
- [OS19] Kenta Oono and Taiji Suzuki, *Approximation and non-parametric estimation of ResNet-type convolutional neural networks*, International Conference on Machine Learning, 2019, pp. 4922–4931.
- [PGZ⁺18] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean, *Efficient neural architecture search via parameters sharing*, International Conference on Machine Learning, 2018, pp. 4095–4104.
- [PKL⁺17] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh N Mhaskar, *Theory of deep learning III: explaining the non-overfitting puzzle*, 2017, arXiv preprint arXiv:1801.00173.
- [PLR⁺16] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli, *Exponential expressivity in deep neural networks through transient chaos*, Advances in Neural Information Processing Systems, 2016, pp. 3368–3376.
- [PMR⁺17] Tomaso Poggio, Hrushikesh N Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao, *Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review*, International Journal of Automation and Computing **14** (2017), no. 5, 503–519.
- [PP92] Etienne Pardoux and Shige Peng, *Backward stochastic differential equations and quasilinear parabolic partial differential equations*, Stochastic partial differential equations and their applications, Springer, 1992, pp. 200–217.
- [PRE17] Vardan Papyan, Yaniv Romano, and Michael Elad, *Convolutional neural networks analyzed via convolutional sparse coding*, Journal of Machine Learning Research **18** (2017), no. 1, 2887–2938.
- [PRMN04] Tomaso Poggio, Ryan Rifkin, Sayan Mukherjee, and Partha Niyogi, *General conditions for predictivity in learning theory*, Nature **428** (2004), no. 6981, 419–422.
- [PRSE18] Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad, *Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks*, IEEE Signal Processing Magazine **35** (2018), no. 4, 72–89.
- [PRV20] Philipp Petersen, Mones Raslan, and Felix Voigtlaender, *Topological properties of the set of functions generated by neural networks of fixed size*, Foundations of Computational Mathematics (2020), 1–70.
- [PSE17] Vardan Papyan, Jeremias Sulam, and Michael Elad, *Working locally thinking globally: Theoretical guarantees for convolutional sparse coding*, IEEE Transactions on Signal Processing **65** (2017), no. 21, 5687–5701.

- [PSMF20] David Pfau, James S Spencer, Alexander GDG Matthews, and W Matthew C Foulkes, *Ab initio solution of the many-electron schrödinger equation with deep neural networks*, Physical Review Research **2** (2020), no. 3, 033429.
- [PV18] Philipp Petersen and Felix Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks*, Neural Networks **108** (2018), 296–330.
- [PV20] ———, *Equivalence of approximation by convolutional neural networks and fully-connected networks*, Proceedings of the American Mathematical Society **148** (2020), no. 4, 1567–1581.
- [REM17] Yaniv Romano, Michael Elad, and Peyman Milanfar, *The little engine that could: Regularization by denoising (red)*, SIAM Journal on Imaging Sciences **10** (2017), no. 4, 1804–1844.
- [RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, *U-net: Convolutional networks for biomedical image segmentation*, International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234–241.
- [RH19] Lars Ruthotto and Eldad Haber, *Deep neural networks motivated by partial differential equations*, Journal of Mathematical Imaging and Vision (2019), 1–13.
- [RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, *Learning representations by back-propagating errors*, Nature **323** (1986), no. 6088, 533–536.
- [RM51] Herbert Robbins and Sutton Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics (1951), 400–407.
- [RMD⁺06] P López Ríos, Ao Ma, Neil D Drummond, Michael D Towler, and Richard J Needs, *Inhomogeneous backflow transformations in quantum Monte Carlo calculations*, Physical Review E **74** (2006), no. 6, 066701.
- [Ros58] Frank Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain*, Psychological review **65** (1958), no. 6, 386.
- [RPK⁺17] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein, *On the expressive power of deep neural networks*, International Conference on Machine Learning, 2017, pp. 2847–2854.
- [RPK19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics **378** (2019), 686–707.
- [RR⁺07] Ali Rahimi, Benjamin Recht, et al., *Random features for large-scale kernel machines*, Advances in Neural Information Processing Systems, 2007, pp. 1177–1184.
- [Rud06] Walter Rudin, *Real and complex analysis*, McGraw-Hill Series in Higher Mathematics, Tata McGraw-Hill, 2006.
- [RWK⁺20] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari, *What’s hidden in a randomly weighted neural network?*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 11893–11902.
- [Sak99] Akito Sakurai, *Tight bounds for the VC-dimension of piecewise polynomial networks*, Advances in Neural Information Processing Systems, 1999, pp. 323–329.
- [SCC18] Uri Shaham, Alexander Cloninger, and Ronald R Coifman, *Provable approximation properties for deep neural networks*, Applied and Computational Harmonic Analysis **44** (2018), no. 3, 537–557.

- [Sch15] Jürgen Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61** (2015), 85–117.
- [SDR14] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński, *Lectures on stochastic programming: modeling and theory*, SIAM, 2014.
- [SEJ⁺20] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, and Alex Bridgland, *Improved protein structure prediction using potentials from deep learning*, Nature **577** (2020), no. 7792, 706–710.
- [SGHK18] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli, *Analysing mathematical reasoning abilities of neural models*, International Conference on Learning Representations, 2018.
- [SGR⁺21] Michael Scherbela, Leon Gerard, Rafael Reisenhofer, Philipp Marquetand, and Philipp Grohs, *Accelerating ab-initio quantum chemistry using weight-sharing deep neural networks*, 2021.
- [SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber, *Training very deep networks*, Advances in Neural Information Processing Systems, 2015, pp. 2377–2385.
- [SH19] Johannes Schmidt-Hieber, *Deep ReLU network approximation of functions on a manifold*, 2019, arXiv preprint arXiv:1908.00695.
- [She20] Zuwei Shen, *Deep network approximation characterized by number of neurons*, Communications in Computational Physics **28** (2020), no. 5, 1768–1811.
- [SHK⁺14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958.
- [SHM⁺16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot, *Mastering the game of go with deep neural networks and tree search*, Nature **529** (2016), no. 7587, 484–489.
- [SHN⁺18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro, *The implicit bias of gradient descent on separable data*, 2018.
- [Ším02] Jiří Šíma, *Training a single sigmoidal neuron is hard*, Neural Computation **14** (2002), no. 11, 2709–2728.
- [SKS⁺17] Kristof T Schütt, Pieter-Jan Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller, *Schnet: A continuous-filter convolutional neural network for modeling quantum interactions*, Advances in Neural Information Processing Systems, 2017, pp. 992–1002.
- [SLJ⁺15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, *Going deeper with convolutions*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
- [SO12] Attila Szabo and Neil S Ostlund, *Modern quantum chemistry: introduction to advanced electronic structure theory*, Courier Corporation, 2012.
- [SPRE18] Jeremias Sulam, Vardan Papayan, Yaniv Romano, and Michael Elad, *Multilayer convolutional sparse modeling: Pursuit and dictionary learning*, IEEE Transactions on Signal Processing **66** (2018), no. 15, 4090–4104.
- [SS16] Itay Safran and Ohad Shamir, *On the quality of the initial basin in overspecified neural networks*, International Conference on Machine Learning, 2016, pp. 774–782.

- [SS17] ———, *Depth-width tradeoffs in approximating natural functions with neural networks*, International Conference on Machine Learning, 2017, pp. 2979–2987.
- [SS18] ———, *Spurious local minima are common in two-layer ReLU neural networks*, International Conference on Machine Learning, 2018, pp. 4433–4441.
- [SSBD14] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms*, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.
- [SSS⁺17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, and Adrian Bolton, *Mastering the game of go without human knowledge*, Nature **550** (2017), no. 7676, 354–359.
- [SSSSS09] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan, *Stochastic convex optimization*, Conference on Learning Theory, 2009.
- [STIM18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry, *How does batch normalization help optimization?*, Advances in Neural Information Processing Systems, 2018, pp. 2488–2498.
- [SZ19] Christoph Schwab and Jakob Zech, *Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq*, Analysis and Applications **17** (2019), no. 01, 19–55.
- [Tal94] Michel Talagrand, *Sharper bounds for Gaussian and empirical processes*, The Annals of Probability (1994), 28–76.
- [Tel15] Matus Telgarsky, *Representation benefits of deep feedforward networks*, 2015, arXiv preprint arXiv:1509.08101.
- [TvG18] Matthew Thorpe and Yves van Gennip, *Deep limits of residual neural networks*, 2018, arXiv preprint arXiv:1810.11741.
- [UVL18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky, *Deep image prior*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9446–9454.
- [Vap99] Vladimir Vapnik, *An overview of statistical learning theory*, IEEE Transactions on Neural Networks **10** (1999), no. 5, 988–999.
- [Vap13] ———, *The nature of statistical learning theory*, Springer science & business media, 2013.
- [VBB19] Luca Venturi, Afonso S Bandeira, and Joan Bruna, *Spurious valleys in one-hidden-layer neural network optimization landscapes*, Journal of Machine Learning Research **20** (2019), no. 133, 1–34.
- [VBC⁺19] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, and Petko Georgiev, *Grandmaster level in StarCraft II using multi-agent reinforcement learning*, Nature **575** (2019), no. 7782, 350–354.
- [VC71] Vladimir Vapnik and Alexey Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability & Its Applications **16** (1971), no. 2, 264–280.
- [vdVW97] Aad W van der Vaart and Jon A Wellner, *Weak convergence and empirical processes with applications to statistics*, Journal of the Royal Statistical Society-Series A Statistics in Society **160** (1997), no. 3, 596–608.

- [Ver18] Roman Vershynin, *High-dimensional probability: An introduction with applications in data science*, vol. 47, Cambridge University Press, 2018.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
- [Wer88] Paul J Werbos, *Generalization of backpropagation with application to a recurrent gas market model*, Neural Networks **1** (1988), no. 4, 339–356.
- [WGB17] Thomas Wiatowski, Philipp Grohs, and Helmut Bölcskei, *Energy propagation in deep convolutional neural networks*, IEEE Transactions on Information Theory **64** (2017), no. 7, 4819–4842.
- [Whi34] Hassler Whitney, *Analytic extensions of differentiable functions defined in closed sets*, Transactions of the American Mathematical Society **36** (1934), no. 1, 63–89.
- [WPC⁺21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip, *A comprehensive survey on graph neural networks*, IEEE Transactions on Neural Networks and Learning Systems **32** (2021), no. 1, 4–24.
- [WZ95] Ronald J Williams and David Zipser, *Gradient-based learning algorithms for recurrent*, Backpropagation: Theory, Architectures, and Applications **433** (1995), 17.
- [WZZ⁺13] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus, *Regularization of neural networks using dropconnect*, International Conference on Machine Learning, 2013, pp. 1058–1066.
- [XM12] Huan Xu and Shie Mannor, *Robustness and generalization*, Machine learning **86** (2012), no. 3, 391–423.
- [Yan19] Greg Yang, *Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation*, 2019, arXiv preprint arXiv:1902.04760.
- [Yar17] Dmitry Yarotsky, *Error bounds for approximations with deep ReLU networks*, Neural Networks **94** (2017), 103–114.
- [Yar18a] ———, *Optimal approximation of continuous functions by very deep ReLU networks*, Conference on Learning Theory, 2018, pp. 639–649.
- [Yar18b] ———, *Universal approximations of invariant maps by neural networks*, 2018, arXiv preprint arXiv:1804.10306.
- [Yar21] ———, *Elementary superexpressive activations*, 2021, arXiv preprint arXiv:2102.10911.
- [YGLD17] Rujie Yin, Tingran Gao, Yue M Lu, and Ingrid Daubechies, *A tale of two bases: Local-nonlocal regularization on image patches with convolution framelets*, SIAM Journal on Imaging Sciences **10** (2017), no. 2, 711–750.
- [YHC18] Jong Chul Ye, Yoseob Han, and Eunju Cha, *Deep convolutional framelets: A general deep learning framework for inverse problems*, SIAM Journal on Imaging Sciences **11** (2018), no. 2, 991–1048.
- [YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria, *Recent trends in deep learning based natural language processing*, IEEE Computational Intelligence Magazine **13** (2018), no. 3, 55–75.
- [Yse10] Harry Yserentant, *Regularity and approximability of electronic wave functions*, Springer, 2010.

- [YZ20] Dmitry Yarotsky and Anton Zhevnerchuk, *The phase diagram of approximation rates for deep neural networks*, Advances in Neural Information Processing Systems, vol. 33, 2020.
- [ZAP16] Hao Zhou, Jose M Alvarez, and Fatih Porikli, *Less is more: Towards compact CNNs*, European Conference on Computer Vision, 2016, pp. 662–677.
- [Zas75] Thomas Zaslavsky, *Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes: Face-count formulas for partitions of space by hyperplanes*, Memoirs of the American Mathematical Society, American Mathematical Society, 1975.
- [ZBH⁺17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, *Understanding deep learning requires rethinking generalization*, International Conference on Learning Representations, 2017.
- [ZBH⁺20] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Michael C Mozer, and Yoram Singer, *Identity crisis: Memorization and generalization under extreme overparameterization*, International Conference on Learning Representations, 2020.
- [ZBS19] Chiyuan Zhang, Samy Bengio, and Yoram Singer, *Are all layers created equal?*, 2019, arXiv preprint arXiv:1902.01996.
- [ZCZG20] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanguan Gu, *Gradient descent optimizes over-parameterized deep ReLU networks*, Machine Learning **109** (2020), no. 3, 467–492.
- [Zho20a] Ding-Xuan Zhou, *Theory of deep convolutional neural networks: Downsampling*, Neural Networks **124** (2020), 319–327.
- [Zho20b] ———, *Universality of deep convolutional neural networks*, Applied and Computational Harmonic Analysis **48** (2020), no. 2, 787–794.
- [ZKS⁺18] Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew J Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, Marc Parente, Krzysztof J Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdal, Adriana Romero, Michael Rabbat, Pascal Vincent, Nafissa Yakubova, James Pinkerton, Duo Wang, Erich Owens, C Lawrence Zitnick, Michael P Recht, Daniel K Sodickson, and Yvonne W Lui, *fastMRI: An open dataset and benchmarks for accelerated MRI*, 2018, arXiv preprint arXiv:1811.08839.
- [ZL17] Barret Zoph and Quoc V Le, *Neural architecture search with reinforcement learning*, International Conference on Learning Representations, 2017.