# 1

# The Modern Mathematics of Deep Learning

Julius Berner, Philipp Grohs, Gitta Kutyniok and Philipp Petersen

**Abstract:** We describe the new field of the 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, a 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.

## 1.1 Introduction

Deep learning has undoubtedly established itself as the outstanding machine learning technique of recent times. This dominant position has been 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 (LeCun et al., 1998; Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016). 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 (He et al., 2015).

A second famous application area is the training of deep-learning-based agents to play board games or computer games, such as Atari games (Mnih et al., 2013). 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 (Silver et al., 2016, 2017) – a feat that was previously unthinkable owing to the extreme complexity



https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

of this game. Moreover, even in multiplayer, team-based games with incomplete information, deep-learning-based agents nowadays outperform world-class human teams (Berner et al., 2019a; Vinyals et al., 2019).

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 (Ma et al., 2015), molecular dynamics (Faber et al., 2017), and in high-energy physics (Baldi et al., 2014). 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 (Senior et al., 2020). 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, and generating text, impressive advances have been made based on deep learning. Here, we refer to Young et al. (2018) for an overview. One technique that has recently stood out is based on a so-called transformer neural network (Bahdanau et al., 2015; Vaswani et al., 2017). This network structure has given rise to the impressive GPT-3 model (Brown et al., 2020) which not only creates coherent and compelling texts but can also produce code, such as that 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 (Saxton et al., 2018; Lample and Charton, 2019).

In this chapter, 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 *the 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 chapter, we will use the term in the following narrow sense: *deep learning refers to techniques where deep neural networks*[1] *are trained with gradient-based methods*. This narrow definition helps to make this chapter 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 chapter, many practically observed phenomena in this context are not explained

[1] We will define the term *neural network* later but, for now, we can view it as a parametrized family of functions with a differentiable parametrization.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

theoretically. Moreover, theoretical insights and the development of a comprehensive theory often constitute 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 fundamental ingredient of the design of aircraft or cars, and the theory of information which affects and shapes all modern digital communication. In the words of Vladimir Vapnik[2]: "Nothing is more practical than a good theory," (Vapnik, 2013, 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 in order 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 McCulloch and Pitts (1943). 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 Rosenblatt (1958). 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, owing 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 Kelley (1960); Dreyfus (1962); Linnainmaa (1970); Rumelhart et al. (1986). Again, these techniques will be discussed in the upcoming sections. Since then, techniques have been improved and extended. As the rest of the chapter is spent reviewing these methods, we will keep the discussion of literature brief at this point. Instead, we refer to some overviews of the history of deep learning from various perspectives: LeCun et al. (2015); Schmidhuber (2015); Goodfellow et al. (2016); Higham and Higham (2019).

Given the fact that the two main ingredients of deep neural networks have been around for a long time, one might expect that a comprehensive mathematical theory would have been developed that describes why and when deep-learning-based methods will perform well or when they will fail. Statistical learning theory (Anthony and Bartlett, 1999; Vapnik, 1999; Cucker and Smale, 2002; Bousquet et al., 2003; Vapnik, 2013) describes multiple aspects of the performance of general learning methods and in particular deep learning. We will review this theory in the

[2] This claim can be found earlier in a non-mathematical context in the works of Kurt Lewin (1943).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

context of deep learning in §1.1.2 below. Here, we focus on the classical, deep-learning-related results that we consider to be well known in the machine learning community. Nonetheless, the choice of these results is guaranteed to be subjective. We will find that this 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, nonlinearity, 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 the natural sciences?*

The new field of the mathematical analysis of deep learning has emerged around questions like those listed above. In the remainder of this chapter, we will collect some of the main recent advances towards answering these questions. Because this field of the mathematical analysis of deep learning is incredibly active and new material is added at breathtaking speed, a brief survey of recent advances in this area is guaranteed to miss not only 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 readers to familiarize themselves with some exciting concepts and provide a convenient entry-point for further studies.

### *1.1.1 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 $f, g \colon \mathcal{X} \to [0, \infty)$, 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 $\ell^p$-norm, and by $\langle \cdot, \cdot \rangle$ the Euclidean inner product of given vectors. By $\|\cdot\|_{\mathrm{op}}$ 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

Fourier transform[3] of a tempered distribution $g$. For probabilistic statements, we will assume a suitable underlying probability space with probability measure $\mathcal{I}$. For an $\mathcal{X}$-valued random variable $X$, we denote by $\mathbb{E}[X]$ and $\mathbb{V}[X]$ its expectation and variance and by $\mathcal{I}_X$ the image measure of $X$ on $\mathcal{X}$, i.e., $\mathcal{I}_X(A) = \mathcal{I}(X \in A)$ 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 $\mathrm{I}_d$ for the $d$-dimensional identity matrix and, for a set $A$, we write $1_A$ for the indicator function of $A$, i.e., $1_A(x) = 1$ if $x \in A$ and $1_A(x) = 0$ otherwise.

### 1.1.2 Foundations of Learning Theory

Before we describe recent developments in the mathematical analysis of deep learning methods, we will start by providing a concise overview of the classical mathematical and statistical theory underlying machine learning tasks and algorithms that, 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 $\mathcal{L}\colon \mathcal{M}(\mathcal{X},\mathcal{Y}) \times \mathcal{Z} \to \mathbb{R}$. The goal is to choose a hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X},\mathcal{Y})$ and to construct a learning algorithm, i.e., a mapping

$$\mathcal{A}\colon \bigcup_{m\in\mathbb{N}} \mathcal{Z}^m \to \mathcal{F},$$

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 similarly 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 $\mathcal{Z} \coloneqq \mathcal{X} \times \mathcal{Y}$, 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, for binary

[3] Respecting common notation, we will also use the hat symbol to denote the minimizer of the empirical risk $\widehat{f_s}$ in Definition 1.8 but this clash of notation does not involve any ambiguity.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

classification tasks with $\mathcal{Y} = \{-1, 1\}$, we consider the 0–1 loss $\mathcal{L}(f, (x, y)) = 1_{(-\infty,0)}(yf(x))$. 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 $f_s : \mathcal{X} \to \mathcal{Y}$, 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 $T_1 : \mathcal{Z} \to \mathcal{X}, T_2 : \mathcal{Z} \to \mathcal{Y}$. In doing so, one asks for a model $f_s$ approximating the transformation $T_2 \circ T_1^{-1} : \mathcal{X} \to \mathcal{Y}$ which is, for example, made 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 $\sigma$-finite reference measure $\mu$ on $\mathcal{Z}$. One then aims for a probability density $f_s$ that approximates the density of the unseen data $z$ with respect to $\mu$. One can perform $L^2(\mu)$-approximation based on the discretization $\mathcal{L}(f, z) = -2f(z) + \|f\|^2_{L^2(\mu)}$ 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 §1.6, we will review some frequently used architectures; in the other sections, however, we will mostly focus on *fully connected feed-forward* (FC) neural networks as defined below.

**Definition 1.4** (FC neural network)**.** A fully connected feed-forward neural network is given by its architecture $a = (N, \varrho)$, where $L \in \mathbb{N}$, $N \in \mathbb{N}^{L+1}$, and $\varrho : \mathbb{R} \to \mathbb{R}$. We refer to $\varrho$ as the activation function, to $L$ as the number of layers, and to $N_0$, $N_L$, and $N_\ell$, $\ell \in [L-1]$, as the number of neurons in the input, output, and $\ell$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 $\Phi_a : \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \to \mathbb{R}^{N_L}$, which

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

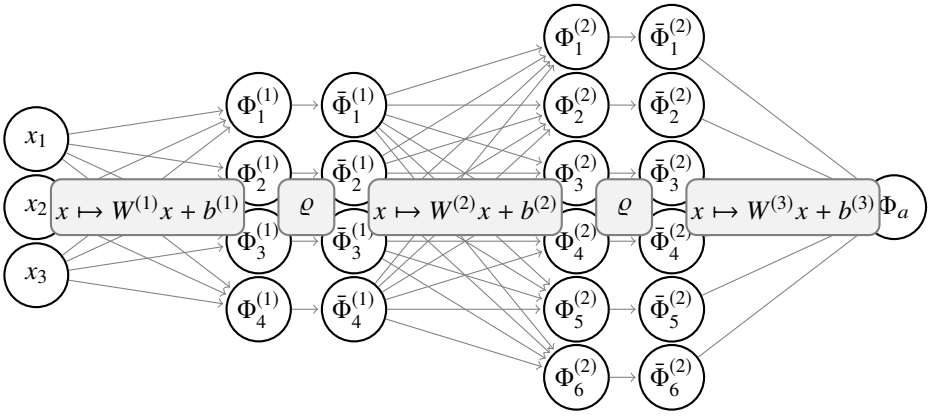



Figure 1.1 Graph (pale gray) and (pre-)activations of the neurons (white) of a deep fully connected feed-forward neural network $\Phi_a : \mathbb{R}^3 \times \mathbb{R}^{53} \mapsto \mathbb{R}$ with architecture $a = ((3, 4, 6, 1), \varrho)$ and parameters $\theta = ((W^{(\ell)}, b^{(\ell)})_{\ell=1}^3$.

satisfies, for every input $x \in \mathbb{R}^{N_0}$ and parameters

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

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

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

and $\varrho$ 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 $N_\ell$ neurons in the $\ell$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 $\varrho$ intertwined; see Figure 1.1. Typical activation 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 $\varrho(x) \to 1$ for $x \to \infty$ and $\varrho(x) \to 0$ for $x \to -\infty$, such as the logistic function $\varrho_\sigma(x) := 1/(1 + e^{-x})$ (often referred to as *the* sigmoid function). See also Table 1.1 for a comprehensive list of widely used activation functions.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

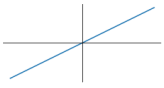
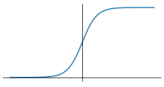
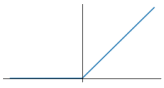
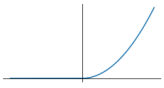
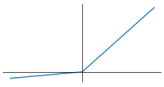
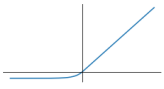
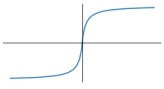
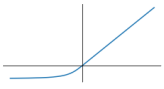
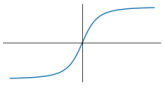
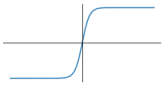
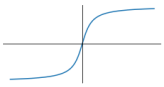
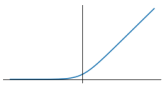
| **Name** | **Given as a function of $x \in \mathbb{R}$ by** | **Plot** |
|---|---|---|
| linear | $x$ |  |
| Heaviside / step function | $1_{(0,\infty)}(x)$ |  |
| logistic / sigmoid | $\frac{1}{1+e^{-x}}$ |  |
| rectified linear unit (ReLU) | $\max\{0, x\}$ |  |
| power rectified linear unit | $\max\{0, x\}^k$ for $k \in \mathbb{N}$ |  |
| parametric ReLU (PReLU) | $\max\{ax, x\}$ for $a \geq 0$, $a \neq 1$ |  |
| exponential linear unit (ELU) | $x \cdot 1_{[0,\infty)}(x) +$ $(e^x - 1) \cdot 1_{(-\infty,0)}(x)$ |  |
| softsign | $\frac{x}{1+\lvert x\rvert}$ |  |
| inverse square root linear unit | $x \cdot 1_{[0,\infty)}(x) +$ $\frac{x}{\sqrt{1+ax^2}} \cdot 1_{(-\infty,0)}(x)$ 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 1.1 *List of commonly used activation functions.*

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

**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 §1.6) can be written as special cases of Definition 1.4 where, for example, 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

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

To enhance readability we will often not specify the underlying architecture $a = (N, \varrho)$ or the parameters $\theta \in \mathbb{R}^{P(N)}$ but use the term NN to refer to the architecture as well as the realization functions $\Phi_a(\cdot, \theta) \colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ or $\Phi_a \colon \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \to \mathbb{R}^{N_L}$. However, we want to emphasize that one cannot infer the underlying architecture or properties such as the 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 $\varrho$. For regression tasks the corresponding hypothesis set is given by

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

and for classification tasks by

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

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 two definitions.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

**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

$$\widehat{\mathcal{R}}_s(f) \coloneqq \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f, z^{(i)}). \tag{1.2}$$

**Definition 1.8** (ERM learning algorithm)**.** Given a hypothesis set $\mathcal{F}$, an empirical risk-minimization algorithm $\mathcal{A}^{\text{erm}}$ chooses[4] 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.,

$$\mathcal{A}^{\text{erm}}(s) \in \operatorname*{argmin}_{f \in \mathcal{F}} \widehat{\mathcal{R}}_s(f). \tag{1.3}$$

**Remark 1.9** (Surrogate loss and regularization)**.** Note that, for classification tasks, one needs to optimize over non-differentiable functions with discrete outputs in (1.3). For an NN hypothesis set $\mathcal{F}_{a,\text{sgn}}$ one typically uses the corresponding hypothesis set for regression tasks $\mathcal{F}_a$ to find an approximate minimizer $\widehat{f_s^{\text{surr}}} \in \mathcal{F}_a$ of

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

where $\mathcal{L}^{\text{surr}} \colon \mathcal{M}(\mathcal{X}, \mathbb{R}) \times \mathcal{Z} \to \mathbb{R}$ is a surrogate loss guaranteeing that $\text{sgn}(\widehat{f_s^{\text{surr}}}) \in \operatorname{argmin}_{f \in \mathcal{F}_{a,\text{sgn}}} \widehat{\mathcal{R}}_s(f)$. A frequently used surrogate loss is the logistic loss,[5] given by

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

In various learning tasks one also adds regularization terms to the minimization problem in (1.3), 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 $\theta$ 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

[4] For simplicity, we assume that the minimum is attained; this is the case, for instance, 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, for example, one chooses a compact parameter set $\mathcal{P} \subset \mathbb{R}^{P(N)}$ and a continuous activation function $\varrho$. One could also work with approximate minimizers: see Anthony and Bartlett (1999).

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 §1.1 for various references.

In this field one usually establishes a connection between the 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 $Z^{(1)}, \dots, Z^{(m)}, Z$.

In this formal setting, we can compute the average out-of-sample performance of a model. Recall from our notation in §1.1.1 that we denote by $\mathcal{I}_Z$ the image measure of $Z$ on $\mathcal{Z}$, which is the underlying distribution of our training data $S = (Z^{(i)})_{i=1}^m \sim \mathcal{I}_Z^m$ and unknown data $Z \sim \mathcal{I}_Z$.

**Definition 1.11** (Risk)**.** For a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, we define[6] the risk by

$$\mathcal{R}(f) \coloneqq \mathbb{E}\big[\mathcal{L}(f, Z)\big] = \int_{\mathcal{Z}} \mathcal{L}(f, z)\, \mathrm{d}\mathcal{I}_Z(z). \tag{1.4}$$

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

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

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

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

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

is called a Bayes-optimal function.

[6] Note 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.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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., Cucker and Zhou (2007, Propositions 1.8 and 9.3).

**Lemma 1.13** (Regression and classification risk)**.** *For a regression task with* $\mathbb{V}[Y] < \infty$*, the risk can be decomposed as follows:*

$$\mathcal{R}(f) = \mathbb{E}\big[(f(X) - \mathbb{E}[Y|X])^2\big] + \mathcal{R}^*, \quad f \in \mathcal{M}(\mathcal{X}, \mathcal{Y}), \tag{1.5}$$

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

$$\mathcal{R}(f) = \mathbb{E}\big[|\mathbb{E}[Y|X]| 1_{(-\infty,0)}(\mathbb{E}[Y|X] f(X))\big] + \mathcal{R}^*, \quad f \in \mathcal{M}(\mathcal{X}, \mathcal{Y}), \tag{1.6}$$

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

As our model $f_S$ depends on the random training data $S$, the risk $\mathcal{R}(f_S)$ is a random variable and we might aim[7] for $\mathcal{R}(f_S)$ to be 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 using only training data, 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 $m$ (Devroye et al., 1996, Theorem 7.2).

**Theorem 1.14** (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* $\mathcal{I}_Z$ *such that for every* $m \in \mathbb{N}$ *and training data* $S \sim \mathcal{I}_Z^m$ *it holds true that*

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

Theorem 1.14 shows the non-existence of a universal learning algorithm for every data distribution $\mathcal{I}_Z$ and shows that useful bounds must necessarily be accompanied by a priori regularity conditions on the underlying distribution $\mathcal{I}_Z$. Such prior knowledge can then be incorporated into the choice of the hypothesis set $\mathcal{F}$. To illustrate this, let $f_{\mathcal{F}}^* \in \operatorname{argmin}_{f \in \mathcal{F}} \mathcal{R}(f)$ be a best approximation in $\mathcal{F}$, such that we can bound the error

$$\begin{aligned}
&\mathcal{R}(f_S) - \mathcal{R}^* \\
&\quad = \mathcal{R}(f_S) - \widehat{\mathcal{R}}_S(f_S) + \widehat{\mathcal{R}}_S(f_S) - \widehat{\mathcal{R}}_S(f_{\mathcal{F}}^*) + \widehat{\mathcal{R}}_S(f_{\mathcal{F}}^*) - \mathcal{R}(f_{\mathcal{F}}^*) + \mathcal{R}(f_{\mathcal{F}}^*) - \mathcal{R}^* \\
&\quad \leq \varepsilon^{\text{opt}} + 2\varepsilon^{\text{gen}} + \varepsilon^{\text{approx}}
\end{aligned} \tag{1.7}$$

[7] In 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, for example, the case if $\mathcal{F}$ constitutes a measurable space and $s \mapsto \mathcal{A}(s)$ and $f \to \mathcal{R}|_{\mathcal{F}}$ are measurable.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

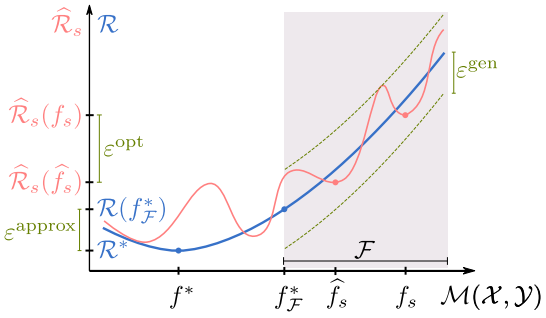



Figure 1.2 Illustration of the errors (A)–(C) in the decomposition (1.7). It shows the exemplary risk $\widehat{\mathcal{R}}$ (blue) and the empirical risk $\widehat{\mathcal{R}}_s$ (red) with respect to the projected space of measurable functions $\mathcal{M}(\mathcal{X}, \mathcal{Y})$. Note that the empirical risk and thus $\varepsilon^{\text{gen}}$ and $\varepsilon^{\text{opt}}$ depend on the realization $s = (z^{(i)})_{i=1}^m$ of the training data $S \sim \mathcal{I}_Z^m$.

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 (uniform[8]) *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 decreases when the hypothesis set is enlarged, but taking $\mathcal{F} = \mathcal{M}(\mathcal{X}, \mathcal{Y})$ prevents control of the generalization error; see also Theorem 1.14. 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 errors with respect to the empirical risk $\widehat{\mathcal{R}}_S$ and its minimizer $\widehat{f}_S$, motivated by the fact that in deep-learning-based applications one typically tries to minimize variants of $\widehat{\mathcal{R}}_S$.

*Optimization*

The first error in the decomposition of (1.7) is the optimization error: $\varepsilon^{\text{opt}}$. 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

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 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* (Kelley, 1960; Dreyfus, 1962; Linnainmaa, 1970; Rumelhart et al., 1986; Griewank and Walther, 2008). This numerical scheme is also applicable in general settings, such as those where 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 $\theta$ according to a variant of *stochastic gradient descent* (SGD), which we shall review below in a general formulation.

**Algorithm 1.1** Stochastic gradient descent

**Input:** Differentiable function $r\colon \mathbb{R}^p \to \mathbb{R}$, 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 for**

If $D^{(k)}$ is chosen deterministically in Algorithm 1.1, 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 $r\colon \mathbb{R}^{P(N)} \to \mathbb{R}$ 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

$$\Theta^{(k)} := \Theta^{(k-1)} - \frac{\eta_k}{m'} \sum_{z \in S'} \nabla_\theta \mathcal{L}(\Phi_a(\cdot,\Theta^{(k-1)}), z), \tag{1.8}$$

where $S'$ is a so-called *mini-batch* of size $|S'| = m'$ chosen uniformly[9] at random from the training data $s$. The sequence of step sizes $(\eta_k)_{k\in\mathbb{N}}$ is often called the *learning rate* in this context. Stopping at step $K$, the output of a deep learning algorithm $\mathcal{A}$ is then given by

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

[9] We remark that in practice one typically picks $S'$ by selecting a subset of training data in such 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 $\Theta^{(k-1)}$.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

where $\bar{\theta}$ can be chosen to be the realization of the last parameter $\Theta^{(K)}$ of (1.8) or a convex combination of $(\Theta^{(k)})_{k=1}^K$ such as the mean.

Algorithm 1.1 was originally introduced in Robbins and Monro (1951) in the context of finding the root of a nondecreasing function from noisy measurements. Shortly afterwards this idea was applied to find the unique global minimum of a Lipschitz-regular function that has no flat regions away from the minimum (Kiefer and Wolfowitz, 1952).

In some regimes, we can guarantee the convergence of SGD at least in expectation. See Nemirovsky and Yudin (1983), Nemirovski et al. (2009), Shalev-Shwartz et al. (2009), Shapiro et al. (2014, Section 5.9), Shalev-Shwartz and Ben-David (2014, Chapter 14). One prototypical convergence guarantee that is found in the aforementioned references in various forms is stated below.

**Theorem 1.15** (Convergence of SGD)**.** *Let $p, K \in \mathbb{N}$ and let $r\colon \mathbb{R}^p \supset B_1(0) \to \mathbb{R}$ be differentiable and convex. Further, let $(\Theta^{(k)})_{k=1}^K$ be the output of Algorithm 1.1 with initialization $\Theta^{(0)} = 0$, step sizes $\eta_k = K^{-1/2}$, $k \in [K]$, and random variables $(D^{(k)})_{k=1}^K$ satisfying $\|D^{(k)}\|_2 \leq 1$ almost surely for all $k \in [K]$. Then*

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

*where $\bar{\Theta} \coloneqq \frac{1}{K}\sum_{k=1}^K \Theta^{(k)}$ and $\theta^* \in \operatorname{argmin}_{\theta \in B_1(0)} r(\theta)$.*

Theorem 1.15 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 cannot in general 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.

Moreover, gradient descent, i.e., the deterministic version of Algorithm 1.1, 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 1.1 is not deterministic then 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 (Nemirovski et al., 2009; Ghadimi and Lan, 2013; Ge et al., 2015; Lee et al., 2016; Jentzen et al., 2020).

In addition, many methods that improve 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 Goodfellow et al. (2016, Chapter 8) for an overview.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

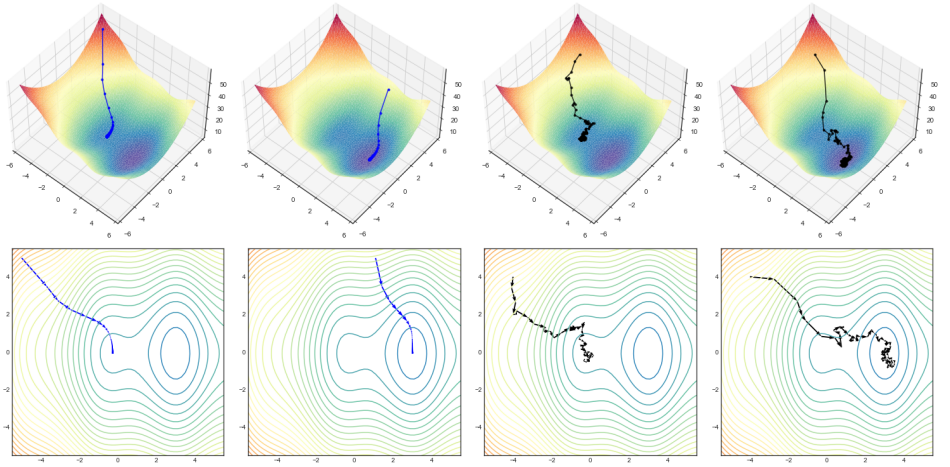


Figure 1.3 Examples of the dynamics of gradient descent (four panels on the left) and stochastic gradient descent (four panels on the right) for an objective function with one non-global minimum next to the global minimum. We see that depending on the initial condition and also on fluctuations in the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

*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 $\varrho$ they can approximate any continuous function on a compact set up to arbitrary precision (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989; Leshno et al., 1993).

**Theorem 1.16** (Universal approximation theorem)**.** *Let $d \in \mathbb{N}$, let $K \subset \mathbb{R}^d$ be compact, and let $\varrho \in L^\infty_{\mathrm{loc}}(\mathbb{R})$ be an activation function such that the closure of the points of discontinuity of $\varrho$ is a Lebesgue null set. Further let*

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

*be the corresponding set of two-layer NN realizations. Then it follows that $C(K) \subset \mathrm{cl}(\widetilde{\mathcal{F}})$ (where 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 $p\colon \mathbb{R} \to \mathbb{R}$ with $p = \varrho$ almost everywhere.*

The theorem can be proven by the Hahn–Banach theorem, which implies that $\widetilde{\mathcal{F}}$ 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

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

In the case $\mathcal{S} = C(K)$ we have by the Riesz–Markov–Kakutani representation theorem that $\mathcal{S}'$ is the space of signed Borel measures on $K$; see Rudin (2006). Therefore, Theorem 1.16 holds if $\varrho$ is such that, for a signed Borel measure $\mu$,

$$\int_K \varrho(\langle w, x\rangle + b)\,\mathrm{d}\mu(x) = 0 \tag{1.9}$$

for all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ implies that $\mu = 0$. An activation function $\varrho$ satisfying this condition is called *discriminatory*. It is not hard to see that any sigmoidal $\varrho$ is discriminatory. Indeed, assume that $\varrho$ satisfies (1.9) for all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Since for every $x \in \mathbb{R}^d$ it follows that $\varrho(ax + b) \to 1_{(0,\infty)}(x) + \varrho(b)1_{\{0\}}(x)$ for $a \to \infty$, we conclude by superposition and passing to the limit that for all $c_1, c_2 \in \mathbb{R}$ and $w \in \mathbb{R}^d$, $b \in \mathbb{R}$,

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

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

Theorem 1.16 addresses the uniform approximation problem on a general compact set. If we are given a finite number of points and care about good approximation only 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 than or equal to the number of data points, then one can always interpolate, i.e., exactly fit the data to a given finite number of points.

**Proposition 1.17** (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 $\varrho$ 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*

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

We sketch the proof as follows. First, note that Theorem 1.16 also holds for functions $g \in C(K, \mathbb{R}^m)$ with multi-dimensional output if we approximate each one-dimensional component $x \mapsto (g(x))_i$ and stack 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 in 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

$g(x^{(i)}) = e^{(i)}$, $i \in [m]$, where $e^{(i)} \in \mathbb{R}^m$ denotes the $i$th standard basis vector. By the discussion above, there exists a neural network architecture $\tilde{a} = ((d, n, m), \varrho)$ and parameters $\tilde{\theta} = ((\widetilde{W}^{(1)}, \tilde{b}^{(1)}), (\widetilde{W}^{(2)}, 0))$ such that

$$\|\Phi_{\tilde{a}}(\cdot, \tilde{\theta}) - g\|_{L^\infty(K)} < \frac{1}{m}, \tag{1.10}$$

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

$$\widetilde{A} := \left[\varrho(\widetilde{W}^{(1)}(x^{(1)}) + \tilde{b}^{(1)})) \cdots \varrho(\widetilde{W}^{(1)}(x^{(m)}) + \tilde{b}^{(1)}))\right] \in \mathbb{R}^{n \times m}. \tag{1.11}$$

The equivalence of the max and operator norm together with (1.10) establish that

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

where $\mathrm{I}_m$ 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 $\tilde{b}^{(1)}$ and the parameters $\theta^{(2)}$ of the second layer by

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

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., $O(m(d + k))$ parameters.

One can also first project the input features onto a one-dimensional line where they are separated and then apply Proposition 1.17 with $d = 1$. For nearly all activation functions, this argument shows that a three-layer NN with only $O(d + mk)$ parameters can interpolate arbitrary training data.[10]

Beyond interpolation results, one can obtain a quantitative version of Theorem 1.16 if one knows additional regularity properties of the Bayes optimal function $f^*$, such as its smoothness, compositionality, and symmetries. For surveys on such results, we refer the reader to DeVore et al. (2021) and Chapter 3 in this book. For instructive purposes we review one such result, which can be found in Mhaskar (1996, Theorem 2.1), next.

**Theorem 1.18** (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 $\varrho$ 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*

[10] To avoid the $m \times d$ weight matrix (without using shared parameters as in Zhang et al., 2017) one interjects an approximate one-dimensional identity (Petersen and Voigtlaender, 2018, 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 (1.12) below.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

*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 true that*

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

Theorem 1.18 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. The strategy involves the re-approximation of classical approximation methods by the use of NNs, thereby transferring the approximation rates of these methods to NNs. In the example of Theorem 1.18, approximation by polynomials is used. Thanks to the non-vanishing derivatives of the activation function,[11] one can approximate every univariate polynomial via divided differences of the activation function. Specifically, accepting unbounded parameter magnitudes, for any activation function $\varrho\colon \mathbb{R}\to\mathbb{R}$ 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

$$\lim_{h\to 0}\sup_{x\in K}\left|\sum_{i=0}^{p}\frac{(-1)^i\binom{p}{i}}{h^p\,\varrho^{(p)}(\lambda)}\varrho\big((p/2-i)hx+\lambda\big) - x^p\right| = 0. \tag{1.12}$$

Let us end this subsection by clarifying the connection of the approximation results above to the error decomposition of (1.7). Consider, for simplicity, a regression task with quadratic loss. Then, the approximation error $\varepsilon^{\text{approx}}$ equals a common $L^2$-error:

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

where the identities marked by $(*)$ follow from Lemma 1.13. Hence, Theorem 1.16 postulates that $\varepsilon^{\text{approx}} \to 0$ for increasing NN sizes, whereas Theorem 1.18 additionally explains how fast $\varepsilon^{\text{approx}}$ converges to 0.

### *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 $\mathcal{L}(f, Z^{(i)})$, $i \in [m]$,

[11] The 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., Donoghue (1969, Chapter 10).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 boundedness[12] of the loss, Hoeffding's inequality(Hoeffding, 1963) and a union bound directly imply the following generalization guarantee for countable, weighted hypothesis sets $\mathcal{F}$; see, e.g., Bousquet et al. (2003).

**Theorem 1.19** (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 $\mathcal{I}_Z^m$-distributed training data $S$) it holds true 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 1.19 implies that, with high probability,

$$\varepsilon^{\text{gen}} \lesssim \sqrt{\frac{\ln(|\mathcal{F}|)}{m}}. \tag{1.13}$$

Again, one notices that, in line with the bias–variance trade-off, the generalization bound increases 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 to a great extent is derived from the work of Vladimir Vapnik and Alexey Chervonenkis (1971). While in (1.13) we counted the number of functions in $\mathcal{F}$, we now refine this analysis to count the number of functions in $\mathcal{F}$, 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; thus

[12] Note that for our classification tasks in Definition 1.2 it follows 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 $\sup_{f\in\mathcal{F}} |\mathcal{L}(f, Z)| \leq 4c^2$.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

$\text{growth}(m, \mathcal{F}) \leq 2^m$. 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*

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

which defines the largest number of points such that $\mathcal{F}$ can realize any classification pattern; see, e.g.,Anthony and Bartlett (1999), Bousquet et al. (2003). There exist various results on the VC-dimensions of NNs with different activation functions; see, for instance, Baum and Haussler (1989), Karpinski and Macintyre (1997), Bartlett et al. (1998), Sakurai (1999). We present the result of Bartlett et al. (1998) for piecewise polynomial activation functions $\varrho$. 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, linearly in the number of parameters $P(N)$ and quadratically in the number of layers $L$.

**Theorem 1.20** (VC-dimension of neural network hypothesis sets)**.** *Let $\varrho$ be a piecewise polynomial activation function. Then there exists a constant $c \in (0, \infty)$ such that for every $L \in \mathbb{N}$ and $N \in \mathbb{N}^{L+1}$,*

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

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 1.20 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 (Talagrand, 1994; Anthony and Bartlett, 1999):

**Theorem 1.21** (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 $m \in \mathbb{N}$, $\delta \in (0, 1)$, then, with probability $1 - \delta$ (with respect to the repeated sampling of $\mathcal{I}_Z^m$-distributed training data $S$), it follows 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 $\varrho$ for a classification task, with high probability it follows that

$$\varepsilon^{\text{gen}} \lesssim \sqrt{\frac{P(N)\log(P(N))}{m}}. \tag{1.14}$$

In the remainder of this section we will sketch a proof of Theorem 1.21 and, in

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

doing so, present further concepts and complexity measures connected with generalization bounds. We start by observing that McDiarmid's inequality (McDiarmid, 1989) ensures that $\varepsilon^{\text{gen}}$ is sharply concentrated around its expectation, i.e., with probability $1-\delta$ it holds true that[13]

$$\left|\varepsilon^{\text{gen}} - \mathbb{E}\left[\varepsilon^{\text{gen}}\right]\right| \lesssim \sqrt{\frac{\log(1/\delta)}{m}}. \tag{1.15}$$

To estimate the expectation of the uniform generalization error we employ a *symmetrization argument* (Giné and Zinn, 1984). Define $\mathcal{G} \coloneqq \mathcal{L} \circ \mathcal{F} \coloneqq \{\mathcal{L}(f,\cdot)\colon f \in \mathcal{F}\}$, let $\widetilde{S} = (\widetilde{Z}^{(i)})_{i=1}^m \sim \mathcal{I}_Z^m$ be a test data set that is independent of $S$, and note that $\mathcal{R}(f) = \mathbb{E}[\widehat{\mathcal{R}}_{\widetilde{S}}(f)]$. By properties of the conditional expectation and Jensen's inequality it follows that

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

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

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

is called the *Rademacher complexity*[14] of $\mathcal{G}$. One can also prove a corresponding lower bound (van der Vaart and Wellner, 1997), i.e.,

$$\mathfrak{R}_m(\mathcal{G}) - \frac{1}{\sqrt{m}} \lesssim \mathbb{E}\left[\varepsilon^{\text{gen}}\right] \lesssim \mathfrak{R}_m(\mathcal{G}). \tag{1.16}$$

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 (Dudley,

[13] For precise conditions to ensure that the expectation of $\varepsilon^{\text{gen}}$ is well defined, we refer readers to van der Vaart and Wellner (1997), Dudley (2014).

[14] Due to our decomposition in (1.7), we want to uniformly bound the absolute value of the difference between the risk and the empirical risk. It is also common just to 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.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

1967; Ledoux and Talagrand, 1991) implies that

$$\mathfrak{R}_m(\mathcal{G}) \lesssim \mathbb{E}\Big[\int_0^\infty \sqrt{\frac{\log N_\alpha(\mathcal{G}, d_S)}{m}}\, \mathrm{d}\alpha\Big], \tag{1.17}$$

where

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

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

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

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

$$N_\alpha(\mathcal{G}, d_S) = N_{2\alpha}(\mathcal{F}, d_{(X^{(i)})_{i=1}^m}). \tag{1.18}$$

The proof is completed by combining the inequalities in (1.15), (1.16), (1.17) and (1.18) with a result of David Haussler (1995) which shows that, for $\alpha \in (0,1)$, we have

$$\log(N_\alpha(\mathcal{F}, d_{(X^{(i)})_{i=1}^m})) \lesssim \mathrm{VCdim}(\mathcal{F}) \log(1/\alpha). \tag{1.19}$$

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 as $d\log(1/\alpha)$. Finally, note that there exists a bound similar to the one in (1.19) for bounded regression tasks that makes use of the so-called *fat-shattering dimension* (Mendelson and Vershynin, 2003, Theorem 1).

### *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 chapter then collects modern approaches to explain the following issues.

**Why do large neural networks not overfit?** In §1.1.2, we have observed that three-layer NNs with commonly used activation functions and only $O(d + m)$ 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 (Zhang et al., 2017), indeed trained convolutional[15] NNs with ReLU activation function and about 1.6 million

[15] The basic definition of a convolutional NN will be given in §1.6. In Zhang et al. (2017) more elaborate versions such as an *inception* architecture (Szegedy et al., 2015) are employed.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

parameters to achieve zero empirical risk on $m = 50,000$ training images of the CIFAR10 dataset (Krizhevsky and Hinton, 2009) with $32 \times 32$ pixels per image, i.e., $d = 1,024$. For such large NNs, generalization bounds scaling with the number of parameters $P(N)$ as the VC-dimension bound in (1.14) are vacuous. However, these workers observed close to state-of-the-art generalization performance.[16]

Generally speaking, NNs are observed in practice 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 the fit to training data and the 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 Neyshabur et al. (2014, 2017), Zhang et al. (2017), Belkin et al. (2019b), Nakkiran et al. (2020):

(i) *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 $\mathrm{VCdim}(\mathcal{F}) \approx m$ or $\mathfrak{R}_m(\mathcal{F}) \approx 1$, rendering our uniform generalization bounds in Theorem 1.21 and in (1.16) vacuous.

(ii) *Lack of explicit regularization:* The test error depends only mildly on explicit regularization, such as norm-based penalty terms or dropout (see Géron, 2017, 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., $\mathcal{A}(s) \in \widetilde{\mathcal{F}}_s \subsetneq \mathcal{F}$.

(iii) *Dependence on the initialization:* The same NN trained to zero empirical risk but 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.

(iv) *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 the generalization performance.

[16] In practice one usually cannot measure the risk $\mathcal{R}(f_s)$ and instead one 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 $\mathcal{I}_Z$ 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*.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

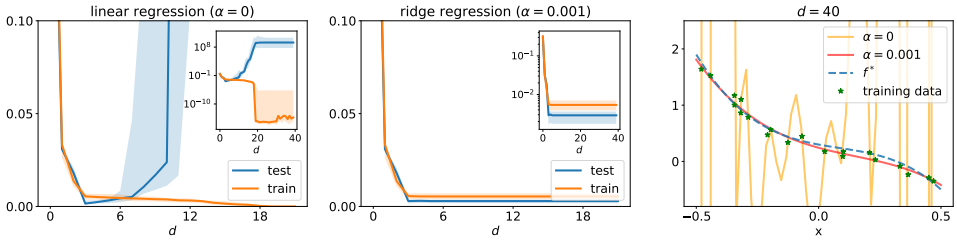



Figure 1.4 The left plot (and its semi-log inset) shows the median and interquartile range of the test and training errors of ten independent linear regressions with $m = 20$ samples, polynomial input features $X = (1, Z, \dots, Z^d)$ of degree $d \in [40]$, and labels $Y = f^*(Z) + \nu$, where $Z \sim \mathcal{U}([-0.5, 0.5])$, $f^*$ is a polynomial of degree three, and $\nu \sim \mathcal{N}(0, 0.01)$. This clearly reflects the classical $\cup$-shaped bias–variance curve with a sweet-spot at $d = 3$ and drastic overfitting beyond the interpolation threshold at $d = 20$. However, the middle 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 $\widehat{f_s}$ are depicted in the right plot.

(v) *Further overparametrization improves generalization performance:* Further increasing the NN size can lead to even lower test error. Together with the previous item, this might require a different treatment of models that are 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., it will have a large test error)", (Hastie et al., 2001). 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, these observations suggest that the generalization performance of NNs depends on an interplay of the data distribution $\mathcal{I}_Z$ 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) on page13 of our error decomposition might deliver insufficient explanation; see also Nagarajan and Kolter (2019). The mismatch between the predictions of classical theory and the practical generalization performance of deep NNs is often referred to as the *generalization puzzle*. In §1.2 we will present possible explanations for this phenomenon.

**What is the role of depth?** We saw in §1.1.2 that NNs can closely approximate every function if they are sufficiently wide (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989). There are additional classical results that even provide a trade-off between the width and the approximation accuracy (Chui et al., 1994; Mhaskar,

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

1996; Maiorov and Pinkus, 1999). 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 (He et al., 2016). After all, the depth of NNs is responsible for the name "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 the effects of depth at many places throughout this chapter. However, one aspects of deep learning that is most clearly affected by deep architectures is the approximation-theoretical aspect. In this framework, we will discuss in §1.3 multiple approaches that describe the effect of depth.

**Why do neural networks perform well in very high-dimensional environments?** We saw in §1.1.2 and will see in §1.3 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* (Bellman, 1952; Novak and Woźniakowski, 2009) 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 $p \to \infty$, when approximating with a continuous scheme[17] of $p$ free parameters (DeVore, 1998).

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 (LeCun et al., 1998) with $28 \times 28$ pixels per image, CIFAR-10/CIFAR-100 (Krizhevsky and Hinton, 2009) with $32 \times 32$ pixels per image, and ImageNet (Deng et al., 2009; Krizhevsky et al., 2012), which contains high-resolution images that are typically down-sampled to $256 \times 256$ 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 $O(\varepsilon^{-784})$ parameters

[17] One 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 (Yarotsky, 2018a; Yarotsky and Zhevnerchuk, 2020).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

to achieve a uniform error of $\varepsilon \in (0,1)$. Even for moderate $\varepsilon$ 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 §1.4, 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 §1.1.2, a convergence guarantee of stochastic gradient descent to a global minimum can typically be given only 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 $\theta$. For a simple intuitive explanation of why this function fails to be convex, it is instructive to consider the following example.

**Example 1.22.** Consider the NN

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

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 function,[18] 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

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

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

Given this non-convexity, Algorithm 1.1 faces serious challenges. First, there may exist multiple suboptimal local minima. Second, the objective function 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.

These issues are not mere theoretical possibilities, but will almost certainly arise in practice. For example, Auer et al. (1996) and Safran and Shamir (2018) showed the existence of many suboptimal local minima in typical learning tasks. Moreover, for fixed-sized NNs, it was shown by Berner et al. (2019b) and Petersen et al. (2020) that, with respect to $L^p$-norms, the set of NNs is generally very non-convex and

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

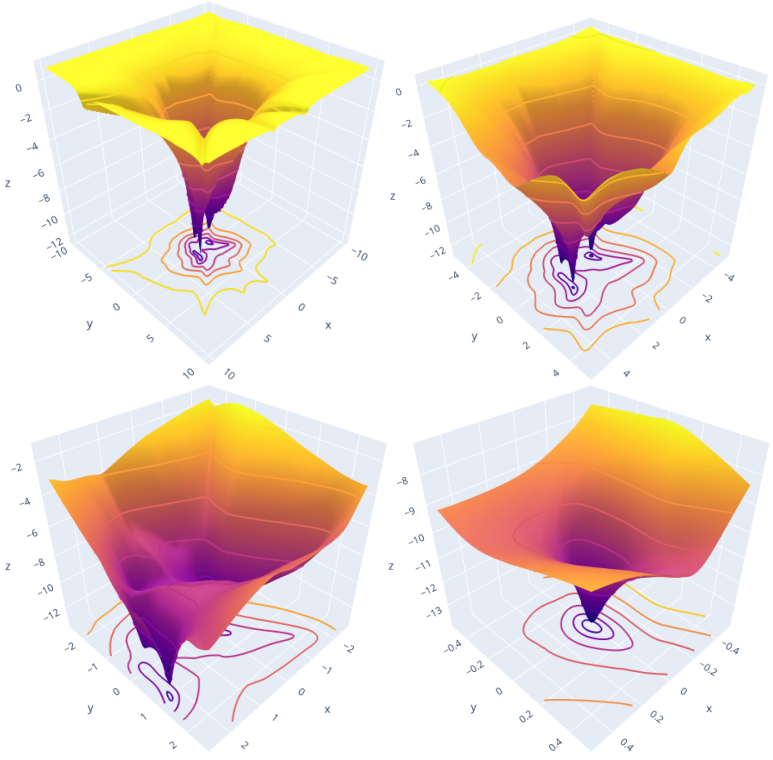


Figure 1.5 Two-dimensional projection of the loss landscape of a neural network with four layers and ReLU activation function on four different scales. From upper left to lower right, we zoom into the global minimum of the landscape.

non-closed. Moreover, the map $\theta \mapsto \Phi_a(\cdot, \theta)$ is not a quotient map, i.e., it is not continuously invertible when its non-injectivity is taken into account. Furthermore, in various situations, finding the global optimum of the minimization problem has been shown to be NP-hard in general (Blum and Rivest, 1989; Judd, 1990; Šíma, 2002). In Figure 1.5 we show the two-dimensional projection of a loss landscape, i.e., a 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 the fact that the loss is only almost everywhere differentiable in the case of piecewise-smooth activation functions, such as the ReLU, although one could resort to subgradient methods (Kakade and Lee, 2018).

In view of these considerations, the classical framework presented in §1.1.2 offers no explanation as to why deep learning works in practice. Indeed, in the survey of Orr and Müller (1998, 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 §1.5 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 saw that, in classical results such as in Theorem 1.18, the effect of only a few aspects of the NN architectures are considered. In Theorem 1.18 only the impact of the width of the NN was studied. In further approximation theorems below, for example, in Theorems 1.23 and 1.25, 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 that some choices are preferable to others. Note that one aspect of the neural network architecture that can have a profound effect on performance, especially regarding the approximation-theoretic aspects of performance, is the choice of the activation function. For example, in Maiorov and Pinkus (1999) and Yarotsky (2021) activation functions were found that allow the uniform approximation of continuous functions to arbitrary accuracy with fixed-size neural networks. In what follows we will focus, however, 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 (LeCun et al., 1998), with skip connections (He et al., 2016), sparse connections (Zhou et al., 2016; Bourely et al., 2017), batch normalization blocks (Ioffe and Szegedy, 2015), and many more. Furthermore, for sequential data, recurrent connections have been used (Rumelhart et al., 1986) and these have often had forget mechanisms (Hochreiter and Schmidhuber, 1997) or other gates (Cho et al., 2014) included in their architectures.

The choice of an appropriate NN architecture is essential to the success of many deep learning tasks. This is so important that frequently an architecture search is applied to find the most suitable one (Zoph and Le, 2017; Pham et al., 2018). 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 would work for a specific problem and those that would yield suboptimal results. In §1.6, 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 at different levels of abstraction that correspond to the layers. This belief is partially grounded in experimental evidence as well as by drawing connections to the human visual cortex; see Goodfellow et al. (2016, Chapter 9.10).

Understanding the features that are learned can be linked, in a way, 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 have evolved in a way that is based on sparse multiscale representations of visual information (Olshausen and Field, 1996). Again, a fascinating question is whether NNs trained in practice can be shown to favor such multiscale representations based on sparsity or whether the architecture is theoretically linked to sparse representations. We will discuss various approaches studying the features learned by neural networks in §1.7.

**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 started to be used also as a numerical ansatz for solving highly complex models from the natural sciences that could be combined with data-driven methods. This is *per se* not very surprising as many such models can be formulated as optimization problems where the commonly used 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 had previously been tackled by highly specialized numerical methods.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

Particular successes include the data-driven solution of ill-posed *inverse problems* (Arridge et al., 2019) which has, for example, led to a fourfold speedup in MRI scantimes (Zbontar et al., 2018) igniting the research project `fastmri.org`. Deep-learning-based approaches have also been very successful in solving a vast array of *partial differential equation* (PDE) models, especially in the high-dimensional regime (E and Yu, 2018; Raissi et al., 2019; Hermann et al., 2020; Pfau et al., 2020) 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 §§1.4.3 and 1.8 we discuss some theoretical and practical aspects of deep learning methods applied to the solution of inverse problems and PDEs.

## 1.2 Generalization of Large Neural Networks

In the following we will shed light on the generalization puzzle of NNs as described in §1.1.3. We focus on four different lines of research which, even so, 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 §1.1.2 that make use of *local* Rademacher complexities (Bartlett et al., 2005) or that drop assumptions on boundedness or rapidly decaying tails (Mendelson, 2014). 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 on the way in which $\mathcal{A}$ chooses its model $f_s$. For instance, one can assume that $f_s$ does not depend too strongly on each individual sample (*algorithmic stability*: Bousquet and Elisseeff, 2002, Poggio et al., 2004), but only on a subset of the samples (*compression bounds*: Arora et al., 2018b), or that it satisfies local properties (*algorithmic robustness*: Xu and Mannor, 2012). Finally, we refer the reader to Jiang et al. (2020) and the references mentioned therein for an empirical study of various measures related to generalization.

Note that many results on the generalization capabilities of NNs can still only be proven in simplified settings, for example 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*.

### *1.2.1 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 $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

such that

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

For instance, in the case of quadratic loss we have that $\ell(\hat{y},y) = (\hat{y}-y)^2$. Further, let $\Phi$ be a NN with architecture $(N,\varrho) = ((d,N_1,\dots,N_{L-1},1),\varrho)$ and let $\Theta_0$ be a $\mathbb{R}^{P(N)}$-valued random variable. For simplicity, we evolve the parameters of $\Phi$ according to the continuous version of gradient descent, so-called *gradient flow*, given by

$$\frac{\mathrm{d}\Theta(t)}{\mathrm{d}t} = -\nabla_\theta \widehat{\mathcal{R}}_s(\Phi(\cdot,\Theta(t))) = -\frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \Phi(x^{(i)},\Theta(t)) D_i(t), \quad \Theta(0)=\Theta_0, \tag{1.20}$$

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

$$\frac{\mathrm{d}\Phi(\cdot,\Theta(t))}{\mathrm{d}t} = -\frac{1}{m}\sum_{i=1}^{m} K_{\Theta(t)}(\cdot,x^{(i)}) D_i(t) \tag{1.21}$$

and of its empirical risk

$$\frac{\mathrm{d}\widehat{\mathcal{R}}_s(\Phi(\cdot,\Theta(t))}{\mathrm{d}t} = -\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} D_i(t) K_{\Theta(t)}(x^{(i)},x^{(j)}) D_j(t), \tag{1.22}$$

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

$$K_\theta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, \quad K_\theta(x_1,x_2) = \big(\nabla_\theta \Phi(x_1,\theta)\big)^T \nabla_\theta \Phi(x_2,\theta). \tag{1.23}$$

Now let $\sigma_w,\sigma_b \in (0,\infty)$ and assume that the initialization $\Theta_0$ consists of independent entries, where entries corresponding to the weight matrix and bias vector in the $\ell$th layer follow a normal distribution with zero mean and variances $\sigma_w^2/N_\ell$ and $\sigma_b^2$, 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 $N_1,\dots,N_{L-1} \to \infty$; see Lee et al. (2018) and Matthews et al. (2018). Similarly, $K_{\Theta_0}$ also converges to a deterministic kernel $K^\infty$ which stays constant in time and depends only on the activation function $\varrho$, the depth $L$, and the initialization parameters $\sigma_w$ and $\sigma_b$ (Jacot et al., 2018; Arora et al., 2019b; Yang, 2019; Lee et al., 2020). Thus, within the infinite width limit, gradient flow on the NN parameters as in (1.20) is equivalent to functional gradient flow in the *reproducing kernel Hilbert space* $(\mathcal{H}_{K^\infty}, \|\cdot\|_{K^\infty})$ corresponding to $K^\infty$; see (1.21).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

By (1.22), 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., Jacot et al., 2018, Du et al., 2019, for suitable conditions) and the $\ell(\cdot, y^{(i)})$ are convex and lower bounded. For instance, in the case of quadratic loss the solution of (1.21) is then given by

$$\Phi(\cdot, \Theta(t)) = C(t)(y^{(i)})_{i=1}^m + \big(\Phi(\cdot, \Theta_0) - C(t)(\Phi(x^{(i)}, \Theta_0))_{i=1}^m\big), \tag{1.24}$$

where $C(t) := \big((K^\infty(\cdot, x^{(i)}))_{i=1}^m\big)^T (\bar{K}^\infty)^{-1}(\mathrm{I}_m - e^{-2\bar{K}^\infty t/m})$. As the initial realization $\Phi(\cdot, \Theta_0)$ constitutes a centered Gaussian process, the second term in (1.24) follows a normal distribution with zero mean at each input. In the limit $t \to \infty$, 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 $K^\infty$ (Belkin et al., 2018; Liang and Rakhlin, 2020; Liang et al., 2020; Ghorbani et al., 2021; Li, 2021).

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 §1.5.2. For sufficiently overparametrized ($P(N) \gg m$) and suitably initialized models, one can show that $K_{\theta(0)}$ is close to $K^\infty$ at initialization and $K_{\theta(t)}$ stays close to $K_{\theta(0)}$ throughout training; see Du et al. (2018b, 2019), Arora et al. (2019b), and Chizat et al. (2019). The dynamics of the NN under gradient flow in (1.21) and (1.22) can thus be approximated by the dynamics of the linearization of $\Phi$ at initialization $\Theta_0$, given by

$$\Phi^{\mathrm{lin}}(\cdot, \theta) := \Phi(\cdot, \Theta_0) + \langle \nabla_\theta \Phi(\cdot, \Theta_0), \theta - \Theta_0\rangle, \tag{1.25}$$

which motivates studying the behavior of linear models in the overparametrized regime.

### 1.2.2 Norm-Based Bounds and Margin Theory

For piecewise linear activation functions, one can improve upon the VC-dimension bounds in Theorem 1.20 and show that, up to logarithmic factors, the VC-dimension is asymptotically bounded both above and below by $P(N)L$; see Bartlett et al. (2019). The lower bound shows that the generalization bound in Theorem 1.21 can be non-vacuous only if the number of samples $m$ scales at least linearly with the number of NN parameters $P(N)$. However, the heavily overparametrized NNs used in practice seem to generalize well outside of this regime.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 (Bartlett, 1998). 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 Neyshabur et al. (2015). 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 1.23** (Rademacher complexity of neural networks)**.** *Let $d \in \mathbb{N}$, assume that $\mathcal{X} = B_1(0) \subset \mathbb{R}^d$, and let $\varrho$ 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*

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

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

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

The factor $2^{L-1}$, depending exponentially on the depth, can be reduced to $\sqrt{L}$ or completely omitted by invoking the spectral norm of the weight matrices (Golowich et al., 2018). 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 (1.16), this motivates why the regularized linear model in Figure 1.4 did perform well in the overparametrized regime.

The proof of Theorem 1.23 is based on the contraction property of the Rademacher complexity (Ledoux and Talagrand, 1991), which establishes that

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

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 follows that

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

In summary, we have established that

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

which by Jensen's inequality yields the claim.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

Recall that for classification problems one typically minimizes a surrogate loss $\mathcal{L}^{\text{surr}}$; see Remark 1.9. This suggests that there could be a trade-off happening between the complexity of the hypothesis class $\mathcal{F}_a$ and the underlying regression fit, 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 Bartlett et al. (2017), Neyshabur et al. (2018), and Jiang et al. (2019). For simplicity, let us focus on the ramp-function surrogate loss with confidence $\gamma > 0$, i.e., $\mathcal{L}_\gamma^{\text{surr}}(f,z) := \ell_\gamma(M(f,z))$, where

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

Note that the ramp function $\ell_\gamma$ is $1/\gamma$-Lipschitz continuous. Using McDiarmid's inequality and a symmetrization argument similar to the proof of Theorem 1.21, combined with the contraction property of the Rademacher complexity, yields the following bound on the probability of misclassification. With probability $1-\delta$ for every $f \in \mathcal{F}_a$ we have

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

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 classified correctly with a margin of at least $\gamma$. 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 increase the complexity of $\mathcal{F}_a$ further, in order to simultaneously increase the margins by which the training data has been classified correctly and thus obtain a better generalization bound.

### *1.2.3 Optimization and Implicit Regularization*

The optimization algorithm, which is usually a variant of SGD, seems to play an important role in generalization performance. Potential indicators for good generalization performance are high speed of convergence (Hardt et al., 2016) or flatness of the local minimum to which SGD converges, which can be characterized by the magnitude of the eigenvalues of the Hessian (or approximately by the robustness of the minimizer to adversarial perturbations on the parameter space); see Keskar et al.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

(2017). In Dziugaite and Roy (2017) and Neyshabur et al. (2017) 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 1.19; see McAllester (1999). Further, one can also unite flatness and norm-based bounds by the *Fisher–Rao metric* of information geometry (Liang et al., 2019).

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

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

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 $\delta$-robustness in the sense that

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

implies that

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

see also Poggio et al. (2017a). This lower bound on the margin $\gamma(\hat{\theta})$ then ensures generalization guarantees, as described in §1.2.2.

Even without explicit[19] control on the complexity of $\mathcal{F}_a$, there do exist results showing that SGD acts as an implicit regularization Neyshabur et al. (2014). This is motivated by linear models where SGD converges to the minimal Euclidean norm solution for a quadratic loss and in the direction of the hard-margin support vector machine solution for the logistic loss on linearly separable data (Soudry et al., 2018). 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 §1.2.2.

While we have seen this behavior of gradient descent for linear regression already in the more general context of kernel regression in §1.2.1, we want to motivate the corresponding result for classification tasks as follows. We focus on the exponential surrogate loss $\mathcal{L}^{\text{surr}}(f, z) = \ell(M(f, z)) = e^{-yf(x)}$ with $\ell(z) = e^{-z}$, but similar observations can be made for the logistic loss defined in Remark 1.9. We assume

[19] Note also that different architectures can exhibit vastly different inductive biases (Zhang et al., 2020) and also that, within an architecture, different parameters have different degrees of importance; see Frankle and Carbin (2018), Zhang et al. (2019), and Proposition 1.29.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 follows that

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

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

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

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

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

and observe that

$$r_\beta(\theta) = -\frac{\log(r(\beta\theta))}{\beta} \to \gamma(\theta), \quad \beta \to \infty. \tag{1.26}$$

Owing to this property, $r_\beta$ is often referred to as the *smoothed margin* (Lyu and Li, 2019; Ji and Telgarsky, 2019b). We evolve $\theta$ according to gradient flow with respect to the smoothed margin $r_1$, i.e.,

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

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

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

This shows that the normalized parameters perform projected gradient ascent with respect to the function $r_{\beta(t)}$, which converges to the margin thanks to (1.26) and the fact that $\beta(t) = \|\theta(t)\|_2 \to \infty$ when approaching a critical point. Thus, during gradient flow, the normalized parameters implicitly maximize the margin. See Gunasekar et al. (2018a), Gunasekar et al. (2018b), Lyu and Li (2019), Nacson et al. (2019), Chizat and Bach (2020), and Ji and Telgarsky (2020) for a precise analysis and various extensions, for example, to homogeneous or two-layer NNs and other optimization geometries.

To illustrate one particular research direction, we now present a result by way of example. 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

$\nabla_{W^{(\ell)}}\Phi = \nabla_{W^{(\ell)}}\Phi(x,\theta) \in \mathbb{R}^{N_{\ell-1}\times N_\ell}$ with respect to the weight matrix in the $\ell$th layer satisfies that

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

where the pre-activations $(\Phi^{(\ell)})_{\ell=1}^{L}$ are as in (1.1). Evolving the parameters according to gradient flow as in (1.20) and using an activation function $\varrho$ with $\varrho(x) = \varrho'(x)x$, such as the ReLU, this implies that

$$\operatorname{diag}\left(\varrho'(\Phi^{(\ell)})\right)W^{(\ell)}(t)\left(\frac{\mathrm{d}W^{(\ell)}(t)}{\mathrm{d}t}\right)^T = \left(\frac{\mathrm{d}W^{(\ell+1)}(t)}{\mathrm{d}t}\right)^T W^{(\ell+1)}(t)\operatorname{diag}\left(\varrho'(\Phi^{(\ell)})\right). \tag{1.27}$$

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

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

see Du et al. (2018a). Furthermore, for deep linear NNs, i.e., $\varrho(x) = x$, the property in (1.27) implies conservation of alignment of the left and right singular spaces $W^{(\ell)}$ and $W^{(\ell+1)}$. This can then be used to show the implicit preconditioning and convergence of gradient descent (Arora et al., 2018a, 2019a) and that, under additional assumptions, gradient descent converges to a linear predictor that is aligned with the maximum margin solution (Ji and Telgarsky, 2019a).

### *1.2.4 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 (Zhang et al., 2017; Nagarajan and Kolter, 2019). 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 ∪-shaped bias–variance curve when the number of parameters is increased further (Belkin et al., 2019b; Geiger et al., 2020; Nakkiran et al., 2020). This is often referred to as the *double descent curve* or *benign overfitting*; see Figure 1.6. For special cases, for example linear regression or random feature regression, such behavior can even be proven; see Hastie et al. (2019), Mei and Montanari (2019), Bartlett et al. (2020), Belkin et al. (2020), and Muthukumar et al. (2020).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

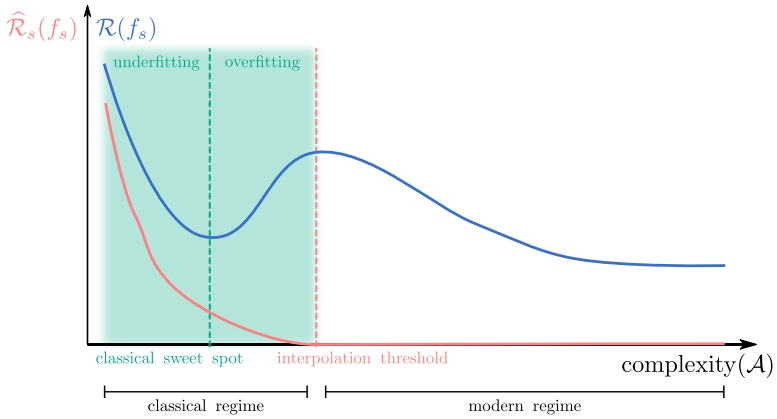



Figure 1.6 This illustration shows the classical, underparametrized regime in green, where the ∪-shaped curve depicts the bias–variance trade-off as explained in §1.1.2. Starting with a complexity of our algorithm $\mathcal{A}$ larger than the interpolation threshold we can achieve zero empirical risk $\widehat{\mathcal{R}}_s(f_s)$ (the training error), where $f_s = \mathcal{A}(s)$. Within this modern interpolation regime, the risk $\mathcal{R}(f_s)$ (the 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 the optimization scheme and the data also influence the complexity, leading to definitions such as complexity($\mathcal{A}$) $:= \max\{m \in \mathbb{N} \colon \mathbb{E}[\widehat{\mathcal{R}}_S(\mathcal{A}(S))] \leq \varepsilon \text{ with } S \sim \mathcal{I}_Z^m\}$, for suitable $\varepsilon > 0$ (Nakkiran et al., 2020). This illustration is based on Belkin et al. (2019b).

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 $\nu$ is a centered random variable that is independent of $X$. For training data $S = ((X^{(i)}, Y^{(i)}))_{i=1}^m$, we consider the empirical risk minimizer $\widehat{f_S} = \langle \hat{\theta}, \cdot \rangle$ with minimum Euclidean norm of its parameters $\hat{\theta}$ or, equivalently, we can consider the limit of gradient flow with zero initialization. Using (1.5) and a bias–variance decomposition we can write

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

where $\Sigma := \sum_{i=1}^m X^{(i)}(X^{(i)})^T$, $\Sigma^+$ denotes the Moore–Penrose inverse of $\Sigma$, and $P := \mathrm{I}_d - \Sigma^+\Sigma$ is the orthogonal projector onto the kernel of $\Sigma$. For simplicity, we focus on the variance $\mathrm{Tr}\big(\Sigma^+ \mathbb{E}[XX^T]\big)$, which can be viewed as the result of 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 $d, m \to \infty$ with $\frac{d}{m} \to \kappa \in (0, \infty)$ can be described via the Marchenko–Pastur law.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

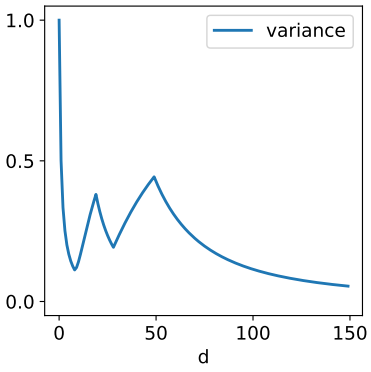



Figure 1.7 The expected variance of the linear regression in (1.29) with $d \in [150]$ and $X_i \sim U(\{-1,1\})$, $i \in [150]$, where $X_i = X_1$ for $i \in \{10,\dots,20\} \cup \{30,\dots,50\}$ and all other coordinates are independent.

Therefore, the asymptotic variance can be computed explicitly as

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

almost surely; see Hastie et al. (2019). This shows that despite interpolating the data we can decrease the risk in the overparametrized regime $\kappa > 1$. In the limit $d,m \to \infty$, 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 (Mei and Montanari, 2019; Montanari and Zhong, 2020; Lin and Dobriban, 2021).

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-asymptomatic regime (Bartlett et al., 2020). Define

$$k^* := \min\left\{k \geq 0 \colon \frac{\sum_{i>k} \lambda_i}{\lambda_{k+1}} \geq cm\right\}, \tag{1.28}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ 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 we have

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

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 a decrease in the risk.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

To motivate this phenomenon, which is considered in much more depth in Chen et al. (2020), let us focus on a single sample $m = 1$ and features $X$ that take values in $\mathcal{X} = \{-1, 1\}^d$. Then it follows that

$$\Sigma^+ = \frac{X^{(1)}(X^{(1)})^T}{\|X^{(1)}\|^4} = \frac{X^{(1)}(X^{(1)})^T}{d^2}$$

and thus

$$\mathbb{E}\big[\mathrm{Tr}\big(\Sigma^+ \mathbb{E}[XX^T]\big)\big] = \frac{1}{d^2}\big\|\mathbb{E}\big[XX^T\big]\big\|_F^2. \tag{1.29}$$

In particular, this shows that by incrementing the input feature dimensions via $d \mapsto d + 1$ 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 1.7.

Generally speaking, overparametrization and the perfect fitting of noisy data does not exclude good generalization performance; see also Belkin et al. (2019a). However, the risk crucially depends on the data distribution and the chosen algorithm.

## 1.3 The Role of Depth in the Expressivity of Neural Networks

The approximation-theoretic aspect of a NN architecture, which is 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 (1.7), is probably one of the most well-studied parts of the deep learning pipe-line. The achievable approximation error of an architecture directly describes the power of the architecture.

As mentioned in §1.1.3, many classical approaches study the approximation theory of NNs with only a few layers, whereas modern architectures are typically very deep. A first observation about the effect of depth is that it can often compensate for insufficient width. For example, in the context of the universal approximation theorem, it has been shown that very narrow NNs are still universal if, instead of increasing the width, the number of layers can be chosen arbitrarily (Hanin and Sellke, 2017; Hanin, 2019; Kidger and Lyons, 2020). However, if the width of a NN falls below a critical number, then the universality will no longer hold.

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.

### *1.3.1 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

significantly more parameters when approximated to the same accuracy by NNs of smaller depth. In the following we present one example of this type of approach, which can be found in Eldan and Shamir (2016).

**Theorem 1.24** (Power of depth)**.** *Let $\varrho \in \{\varrho_R, \varrho_\sigma, 1_{(0,\infty)}\}$ 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 $\mu$ 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 $n \leq ce^{cd}$ we have*

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

In fact, the activation function in Theorem 1.24 is required to satisfy only mild conditions and the result holds, for instance, also for more general sigmoidal functions. The proof of Theorem 1.24 is based on the construction of a suitable radial function $g\colon \mathbb{R}^d \to \mathbb{R}$, i.e., $g(x) = \tilde{g}(\|x\|_2^2)$ for some $\tilde{g}\colon [0,\infty) \to \mathbb{R}$, which can be efficiently approximated by three-layer NNs but for which approximation by only a two-layer NN requires exponentially large complexity, i.e., a width that is exponential in $d$.

The first observation of Eldan and Shamir (2016) was 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 $\varepsilon$ by a two-layer architecture of width $O(C/\varepsilon)$. The same holds for smooth, non-polynomial activation functions, owing to Theorem 1.18. 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 $\varepsilon$ by a two-layer architecture of width $O(d^2/\varepsilon)$. Moreover, this shows that the third layer can efficiently approximate $\tilde{g}$, establishing the approximation of $g$ on a bounded domain up to precision $\varepsilon$ using a three-layer architecture with the number of parameters polynomial in $d/\varepsilon$.

The second step of Eldan and Shamir (2016) was 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 $n$ and not exponential in $d$ is on average (with respect to the probability measure $\mu$) 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 $\varphi$ of the density function associated with the probability measure $\mu$, as well as $\Phi$ and $g$, are well behaved, the Plancherel theorem

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

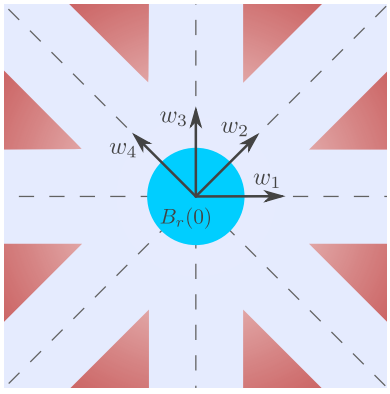



Figure 1.8 This illustration shows the largest possible support (blue) of $\widehat{\Phi\varphi}$, where $\hat{\varphi} = 1_{B_r(0)}$ and $\Phi$ is a shallow neural network with architecture $N = (2, 4, 1)$ and weight matrix $W^{(1)} = [w_1 \cdots w_4]^T$ in the first layer. Any radial function with too much of its $L^2$-mass located at high frequencies (indicated in red) cannot be well approximated by $\Phi\varphi$.

yields

$$\|\Phi - g\|_{L^2(\mu)}^2 = \|\Phi\varphi - g\varphi\|_{L^2(\mathbb{R}^d)}^2 = \left\|\widehat{\Phi\varphi} - \widehat{g\varphi}\right\|_{L^2(\mathbb{R}^d)}^2. \tag{1.30}$$

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 $\varrho_j \colon \mathbb{R} \to \mathbb{R}$ (subsuming the activation function $\varrho$, the norm of $w_j$, and the remaining parameters corresponding to the $j$th neuron in the hidden layer) such that $\Phi$ is of the form

$$\Phi = \sum_{j=1}^{n} \varrho_j(\langle w_j, \cdot \rangle) = \sum_{j=1}^{n} (\varrho_j \otimes 1_{\mathbb{R}^{d-1}}) \circ R_{w_j}. \tag{1.31}$$

The second equality follows by viewing the action of the $j$th neuron as a tensor product of $\varrho_j$ and the indicator function $1_{\mathbb{R}^{d-1}}(x) = 1$, $x \in \mathbb{R}^{d-1}$, composed with a $d$-dimensional rotation $R_{w_j} \in \mathrm{SO}(d)$ 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 $\bigcup_{j=1}^{n} \mathrm{span}\{w_j\}$, which represent lines passing through the origin.

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

density for $\mu$ as

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

Using the convolution theorem, this choice of $\varphi$ yields that

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

Thus the lines passing through the origin are enlarged to tubes. It is this particular shape which allows the construction of some $g$ such that $\|\widehat{\Phi\varphi} - \widehat{g\varphi}\|^2_{L^2(\mathbb{R}^d)}$ can be suitably lower bounded; see also Figure 1.8. Intriguingly, the peculiar behavior of high-dimensional sets now comes into play. Owing to the well-known concentration of measure principle, the variable $n$ needs to be exponentially large for the set $\bigcup_{j=1}^{n} \left(\operatorname{span}\{w_j\} + B_r(0)\right)$ not to be sparse. If it is smaller, one can construct a function $g$ such 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 $\|\Phi - g\|^2_{L^2(\mu)}$; see (1.30). One key technical problem is the fact that such a behavior for $\hat{g}$ does not immediately imply a similar behavior for $\widehat{g\varphi}$, requiring a quite delicate construction of $g$.

### *1.3.2 Deep ReLU Networks*

Perhaps 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 the corresponding NN architectures $(N, \varrho_R)$ 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 $\Phi$ is a continuous piecewise affine linear function. Moreover, $\Phi$ has at most $n + 1$ affine linear pieces. On the other hand, notice that the *hat function*

$$\begin{aligned} h\colon [0,1] &\to [0,1], \\ x &\mapsto 2\varrho_R(x) - 4\varrho_R(x - \tfrac{1}{2}) = \begin{cases} 2x, & \text{if } 0 \le x < \frac{1}{2}, \\ 2(1-x), & \text{if } \frac{1}{2} \le x \le 1, \end{cases} \end{aligned} \tag{1.32}$$

is a NN with two layers and two neurons. Telgarsky (2015) observed that the $n$-fold convolution $h_n(x) \coloneqq h \circ \cdots \circ h$ produces a sawtooth function with $2^n$ spikes. 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.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

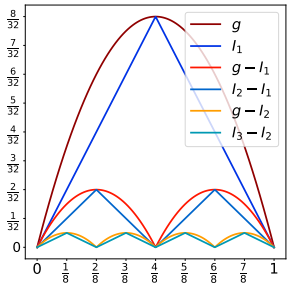



Figure 1.9 Interpolation $I_n$ of $[0,1] \ni x \mapsto g(x) := x - x^2$ on $2^n + 1$ equidistant points, which can be represented as a sum $I_n = \sum_{k=1}^{n} I_k - I_{k-1} = \sum_{k=1}^{n} h_k / 2^{2k}$ of $n$ sawtooth functions. Each sawtooth function $h_k = h_{k-1} \circ h$ in turn can be written as a $k$-fold composition of a hat function $h$. This illustration is based on Elbrächter et al. (2019).

Moreover, it was noted in Yarotsky (2017) that the difference in 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 $h_n / 2^{2n}$; see Figure 1.9. This permits efficient implementation of approximative squaring and, by polarization, also of approximate multiplication using ReLU NNs. Composing these simple functions one can approximate localized Taylor polynomials and thus smooth functions; see Yarotsky (2017). We state below a generalization (Gühring et al., 2020) of Yarotsky's result which includes more general norms, but which for $p = \infty$ and $s = 0$ coincides with his original result.

**Theorem 1.25** (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 $\varrho$ 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 $\|g\|_{W^{k,p}((0,1)^d)} \leq B$ we have*

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

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 Opschoor et al. (2020) and Marcati et al. (2020) 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

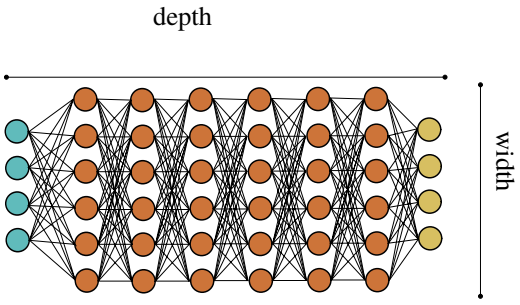



Figure 1.10 Standard feed-forward neural network. For certain approximation results, depth and width need to be in a fixed relationship to achieve optimal results.

of parameters of neural networks needed to approximate such a function in the $W^{1,2}$-norm up to an error of $\varepsilon$ is logarithmic in $\varepsilon$.

Theorem 1.25 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 Liang and Srikant (2017), Safran and Shamir (2017), Yarotsky (2017), Petersen and Voigtlaender (2018), andElbrächter et al. (2019). We state here Yarotsky's result:

**Theorem 1.26** (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 that 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 follows that*

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

This results 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 as $\prod_{\ell=1}^{L-1} N_\ell$.

In the framework of the aforementioned results, we can speak of a depth–width trade-off; see also Figure 1.10. A fine-grained estimate of achievable rates for freely varying depths was also established in Shen (2020).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

### 1.3.3 Alternative Notions of Expressivity

Conceptual approaches to studying the approximation power of deep NNs beyond 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 was by Montúfar et al. (2014) who described bounds on the number of *affine linear regions* of a ReLU NN $\Phi_{(N,\varrho_R)}(\cdot,\theta)$. In a simplified setting, we already saw estimates on the number of affine linear pieces at the beginning of §1.3.2. Affine linear regions can be defined as the connected components of $\mathbb{R}^{N_0} \setminus H$, where $H$ is the set of non-differentiable parts of the realization[20] $\Phi_{(N,\varrho_R)}(\cdot,\theta)$. A refined analysis on the number of such regions was conducted, for example, by Hinz and van de Geer (2019). It was found that deep ReLU neural networks can exhibit significantly more of such regions than of their shallow counterparts.

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

$$\{x \in \mathbb{R}^d : \langle x, w\rangle + b = 0\}. \tag{1.33}$$

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 (Zaslavsky, 1975), one can bound the number of affine linear regions by $\sum_{j=0}^{d} \binom{n}{j}$. Deepening the 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 regions.[21]

This approach was further developed in Raghu et al. (2017) to a framework to study expressivity that to some extent allows to include the training phase. One central object studied in Raghu et al. (2017) 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 found an exponential dependence of the expected curve length on the depth. Let us motivate this in the

[20] One can also study the potentially larger set of *activation regions* given by the connected components of $\mathbb{R}^{N_0} \setminus \big(\bigcup_{\ell=1}^{L-1} \bigcup_{i=1}^{N_\ell} H_{i,\ell}\big)$, 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-differentiable parts of the activation of the $i$th neuron in the $\ell$th layer. In contrast with the linear regions, the activation regions are necessarily convex (Raghu et al., 2017; Hanin and Rolnick, 2019).

[21] However, to exploit this efficiency with respect to the depth, one requires highly oscillating pre-activations snd this 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 (Hanin and Rolnick, 2019).

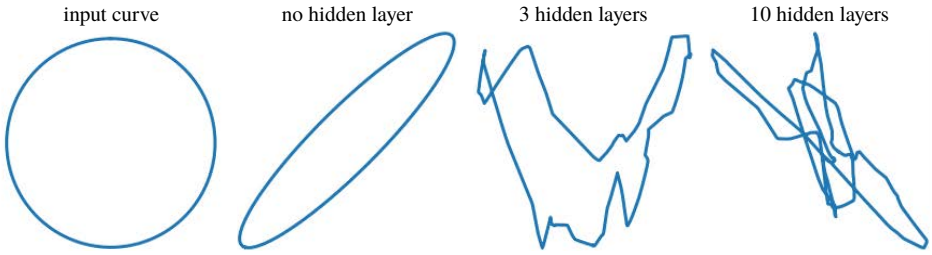



Figure 1.11 Shape of the trajectory $t \mapsto \Phi_{((2,n,\ldots,n,2),\varrho_R)}(\gamma(t),\theta)$ of the output of a randomly initialized network with 0, 3, or 10 hidden layers. The input curve $\gamma$ is the circle given in the leftmost image. The hidden layers have $n = 20$ neurons and the variance of the initialization is taken as $4/n$.

special case of a ReLU NN with architecture $a = ((N_0, n, \ldots, n, N_L), \varrho_R)$ and depth $L \in \mathbb{N}$.

Given a non-constant continuous curve $\gamma\colon [0,1] \to \mathbb{R}^{N_0}$ in the input space, the length of the trajectory in the $\ell$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 $\ell$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 $\Theta_1$ of the NN $\Phi_a$ be initialized independently in such a way 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, for example by Proposition 1.17, that the probability that $\bar{\Phi}^{(\ell)}(\cdot,\Theta_1)$ will map $\gamma$ to a non-constant curve is positive and hence, for fixed $\ell \in [L-1]$,

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

Let $\sigma \in (0,\infty)$ and consider a second initialization $\Theta_\sigma$, where we have changed the variances of the entries corresponding to the weight matrices and bias vectors to $\sigma^2/n$ and $\sigma^2$, 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

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

i.e., the activations corresponding to the two initialization strategies are identically distributed up to the factor $\sigma^\ell$. Therefore, we immediately conclude that

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

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 (Poole

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

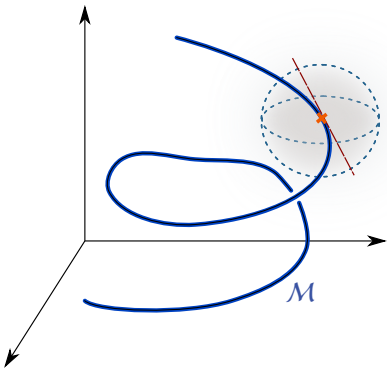



Figure 1.12 Illustration of a one-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$. For every point $x \in \mathcal{M}$ there exists a neighborhood in which the manifold can be linearly projected onto its tangent space at $x$ such that the corresponding inverse function is differentiable.

et al., 2016). In Raghu et al. (2017) this result is also extended to a 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 1.11.

## 1.4 Deep Neural Networks Overcome the Curse of Dimensionality

In §1.1.3, 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 (Bellman, 1952; DeVore, 1998; Novak and Woźniakowski, 2009).

In this section, we present three approaches that offer explanations of this phenomenon. As before, we have 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 Poggio et al. (2017b). Note that also a suitable feature descriptor, factoring out invariances, might lead to a significantly reduced effective dimension; see §1.7.1.

### *1.4.1 Manifold Assumption*

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

function

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

where $d$ is very large. However, we are not seeking to optimize with respect to the uniform norm or a regular $L^p$ space; instead, we consider a measure $\mu$ 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 was fundamental in Chui and Mhaskar (2018), Shaham et al. (2018), Chen et al. (2019), Schmidt-Hieber (2019), Cloninger and Klock (2020), Nakada and Imaizumi (2020). Let us outline an example-based approach, where we consider locally $C^k$-regular functions and NNs with ReLU activation functions below.

(i) *The regularity of g on the manifold is described.* 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 1.12. 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 $\gamma_i$ 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 $i \in [p]$.

(ii) *Localization and charts are constructed via neural networks.* According to the construction of local coordinate charts in Step (i), we can write $g$ as follows:

$$g(x) = \sum_{i=1}^{p} \phi_i(x) \left( g \circ \gamma_i^{-1}(\gamma_i(x)) \right) =: \sum_{i=1}^{p} \tilde{g}_i(\gamma_i(x), \phi_i(x)), \quad x \in \mathcal{M}, \tag{1.34}$$

where $\phi_i$ is a partition of unity such that $\operatorname{supp}(\phi_i) \subset U_i$. Note that $\gamma_i$ 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 $\phi_i$ 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 He et al. (2020) 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.

(iii) *A classical approximation result is used on the localized functions.* By some

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

form of Whitney's extension theorem (Whitney, 1934), 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 $O(\varepsilon^{-(d'+1)/k})$ for $\varepsilon \to 0$; see Mhaskar (1996), Yarotsky (2017), and Shaham et al. (2018).

(iv) *The compositionality of neural networks is used to build the final network.* We have seen that every component in the representation (1.34), i.e., $\tilde{g}_i$, $\gamma_i$, and $\phi_i$, 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 (1.34) 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 of reducing the complexity of a high-dimensional problem to the underlying low-dimensional problem given by the data distribution.

### 1.4.2 Random Sampling

As early as 1992, Andrew Barron showed that, under certain seemingly very natural assumptions on the function to be approximated, a dimension-independent approximation rate by NNs can be achieved (Barron, 1992, 1993). Specifically, the assumption is formulated as a condition on the Fourier transform of a function, and the result is as follows.

**Theorem 1.27** (Approximation of Barron-regular functions)**.** *Let $\varrho\colon \mathbb{R} \to \mathbb{R}$ 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 $\mu$ supported on $B_1(0) \subset \mathbb{R}^d$, and every $g \in L^1(\mathbb{R}^d)$ with $C_g := \int_{\mathbb{R}^d} \|\xi\|_2 |\hat{g}(\xi)| \, \mathrm{d}\xi < \infty$ it follows that*

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

Note that the $L^2$-approximation error can be replaced by an $L^\infty$-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 1.27 is the following application of the law of large numbers. First, we observe that, as per the assumption, $g$ can be represented via the

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

inverse Fourier transform as

$$\begin{aligned} g - g(0) &= \int_{\mathbb{R}^d} \hat{g}(\xi)(e^{2\pi i \langle \cdot, \xi \rangle} - 1)\,\mathrm{d}\xi \\ &= C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2}(e^{2\pi i \langle \cdot, \xi \rangle} - 1)\frac{1}{C_g}\|\xi\|_2 \hat{g}(\xi)\,\mathrm{d}\xi \\ &= C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2}(e^{2\pi i \langle \cdot, \xi \rangle} - 1)\,\mathrm{d}\mu_g(\xi), \end{aligned} \tag{1.35}$$

where $\mu_g$ is a probability measure. Then it was further shown by Barron (1992) that there exist $(\mathbb{R}^d \times \mathbb{R})$-valued random variables $(\Xi, \widetilde{\Xi})$ such that (1.35) can be written as

$$g(x) - g(0) = C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2}(e^{2\pi i \langle x, \xi \rangle} - 1)\,\mathrm{d}\mu_g(\xi) = C_g \mathbb{E}\big[\Gamma(\Xi, \widetilde{\Xi})(x)\big], \quad x \in \mathbb{R}^d, \tag{1.36}$$

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

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

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

$$\begin{aligned} &\mathbb{E}\left[\left\| g - g(0) - \frac{C_g}{n}\sum_{i=1}^{n} \Gamma(\Xi^{(i)}, \widetilde{\Xi}^{(i)})\right\|_{L^2(\mu)}^2\right] \\ &\quad = \int_{B_1(0)} \mathbb{V}\left[\frac{C_g}{n}\sum_{i=1}^{n} \Gamma(\Xi^{(i)}, \widetilde{\Xi}^{(i)})(x)\right] \mathrm{d}\mu(x) \\ &\quad = \frac{C_g^2 \int_{B_1(0)} \mathbb{V}\big[\Gamma(\Xi, \widetilde{\Xi})(x)\big]\,\mathrm{d}\mu(x)}{n} \le \frac{(2\pi C_g)^2}{n}, \end{aligned} \tag{1.37}$$

where the last inequality follows from combining (1.36) with the fact that

$$|e^{2\pi i \langle x, \xi \rangle} - 1|/\|\xi\|_2 \le 2\pi, \qquad x \in B_1(0).$$

This implies that there exists a realization $((\xi^{(i)}, \tilde{\xi}^{(i)}))_{i \in \mathbb{N}}$ of the random variables $((\Xi^{(i)}, \widetilde{\Xi}^{(i)}))_{i \in \mathbb{N}}$ that achieves an $L^2$-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 $1_{(0,\infty)}$ 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 having 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 in the dissertation of Emmanuel Candès (1998) that the

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

approximation rate of NNs for Barron-regular functions is also achievable by $n$-term approximation with complex exponentials, as is apparent by considering (1.35). 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 (Caragea et al., 2020).

In addition, the random sampling idea above was extended in E et al. (2019d, 2020), and E and Wojtowytsch (2020b,c) to facilitate the dimension-independent approximation of vastly more general function spaces. Basically, the idea is to use (1.36) as an inspiration and define a *generalized Barron space* as comprising all functions that may be represented as

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

for any random variable $(\Xi, \widetilde{\Xi})$. In this context, deep and compositional versions of Barron spaces were introduced and studied in Barron and Klusowski (2018), E et al. (2019a), and E and Wojtowytsch (2020a), which considerably extend the original theory.

### 1.4.3 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 do suffer from the curse of dimensionality.

This is not merely an abstract theoretical problem. Very recently, Al-Hamdani et al. (2020) showed 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 represent accurately 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, Grohs et al. (2021), Gonon and Schwab (2020), and Hutzenthaler et al. (2020) for (linear and semilinear) parabolic evolution equations, Elbrächter et al. (2019) for stationary elliptic PDEs, Grohs and Herrmann (2021) for nonlinear Hamilton–Jacobi–Bellman equations, and Kutyniok et al. (2019) 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.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 $g\colon \mathbb{R}^d \times [0,\infty) \to \mathbb{R}$ satisfying[22]

$$\frac{\partial g}{\partial t}(x,t) = \frac{1}{2}\mathrm{Tr}\big(\sigma(x,t)[\sigma(x,t)]^* \nabla_x^2 g(x,t)\big) + \langle \mu(x,t), \nabla_x g(x,t)\rangle, \quad g(x,0) = \varphi(x), \tag{1.38}$$

where the functions

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

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

$$d\mathcal{S}_{x,t} = \mu(\mathcal{S}_{x,t})dt + \sigma(\mathcal{S}_{x,t})dB_t, \quad \mathcal{S}_{x,0} = x, \tag{1.39}$$

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

$$g(x,t) = \mathbb{E}[\varphi(\mathcal{S}_{x,t})], \quad x \in \mathbb{R}^d,\ t \in [0,\infty). \tag{1.40}$$

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

$$g(x,t) = \mathbb{E}[\varphi(\mathcal{S}_{x,t})] \approx \frac{1}{n}\sum_{i=1}^{n} \varphi(\mathcal{S}_{x,t}^{(i)}), \tag{1.41}$$

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 (1.37) for a similar estimate and Grohs et al. (2021) and Gonon and Schwab (2020) for two general approaches to ensure this property. Aside from this issue, (1.41) 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 (1.39) by NNs; this, roughly speaking, can be done without incurring the curse of dimensionality whenever the coefficient functions $\mu, \sigma$ can be approximated by NNs without incurring the curse of dimensionality and when some growth conditions hold true. In a final step one assumes that the initial condition $\varphi$ can be approximated by NNs without incurring the curse

[22] The natural solution concept to this type of PDEs is the viscosity solution concept, a thorough study of which can be found in Hairer et al. (2015).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 (1.41) this implies a corresponding approximation result for the solution of the Kolmogorov PDE $g$ in (1.38).

Informally, we have discovered a regularity result for linear Kolmogorov equations, namely that (modulo some technical conditions on $\mu, \sigma$), *the solution $g$ of* (1.38) *can be approximated by NNs without incurring the curse of dimensionality whenever the same holds true for the initial condition $\varphi$, as well as for the coefficient functions $\mu$ and $\sigma$*. In other words, *the property of being approximable by NNs without curse of dimensionality is preserved under the flow induced by the PDE* (1.38). Some comments are in order.

**Assumption on the initial condition.** One may wonder if the assumption that the initial condition $\varphi$ 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 $\varphi$ typically represents an option pricing formula and (1.38) 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 $\varphi$ can be *exactly* represented by a small ReLU NN.

**Generalization and optimization error.** The Feynman–Kac representation (1.40) 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 (Beck et al., 2021; Berner et al., 2020a).

While it is in principle possible to analyze the approximation and generalization errors, the analysis of the computational cost and/or convergence of the corresponding SGD algorithms is completely open. Some promising numerical results exist – see, for instance, Figure 1.13 – but the stable training of NNs approximating PDEs to very high accuracy (which is needed in several applications such as quantum chemistry) remains very challenging. Recent work (Grohs and Voigtlaender, 2021) has even proved several impossibility results in that direction.

**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 Pardoux and Peng (1992) in place of (1.40) and multilevel Picard sampling algorithms of E et al. (2019c) in place of (1.41).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

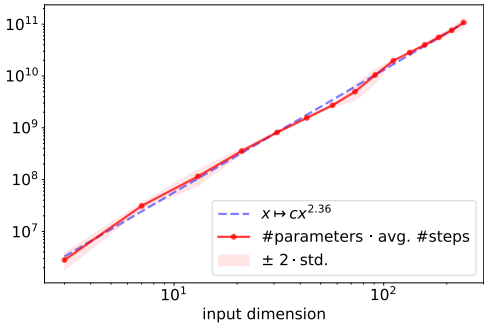



Figure 1.13 Computational complexity as the number of neural network parameters times the number of SGD steps needed to solve heat equations of varying dimensions up to a specified precision. According to the fit above, the scaling is polynomial in the dimension (Berner et al., 2020b).

We can also formulate the underlying idea in an abstract setting (a version of which has also been used in §1.4.2). Assume that a high-dimensional function $g\colon \mathbb{R}^d \to \mathbb{R}$ admits a probabilistic representation of the form

$$g(x) = \mathbb{E}[Y_x], \quad x \in \mathbb{R}^d, \tag{1.42}$$

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 $T_\ell$, $\ell \in [L]$, by NNs and if the numerical scheme is stable enough, then we can also approximate $\mathcal{Y}_x^{(L)}$ using compositionality. Emulating a uniform Monte-Carlo approximator of (1.42) then leads to approximation results for $g$ without the 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, Elbrächter et al. (2018), Li et al. (2019a), and Schwab and Zech (2019). 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 (Laakmann and Petersen, 2021).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

## 1.5 Optimization of Deep Neural Networks

We recall from §§1.1.3 and 1.1.2 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 non-convex; it 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, an excellent performance of SGD is observed. This indicates that the trajectory of the optimization routine somehow misses sub-optimal critical points and other areas that may lead to slow convergence. Clearly, the classical theory does not explain this performance. Below we describe using examples some novel approaches that give partial explanations of this success.

In keeping with the flavor of this chapter, 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 has had to be made and some ground-breaking results have had to be omitted.

### *1.5.1 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 such a 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 Choromanska et al. (2015a) that the loss associated with 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 critical points of the Hamiltonian, and associate with each

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

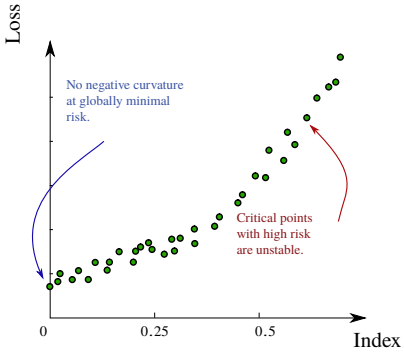



Figure 1.14 The distribution of critical points of the Hamiltonian of a spin glass model.

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 that in Figure 1.14 emerges (Auffinger et al., 2013). 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 (Choromanska et al., 2015b), the theoretical distribution of critical points in Figure 1.14 is visible in many practical applications (Dauphin et al., 2014).

**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. Should there exist a path of non-increasing empirical risk from every point to the global minimum, then we can be certain that no non-global minimum exists, since no such path could escape such a minimum. An even stronger result holds: the existence of such paths shows that the loss landscape has connected level sets (Freeman and Bruna, 2017; Venturi et al., 2019).

A crucial ingredient of the analysis of such paths is *linear substructures*. Consider a biasless two-layer NN $\Phi$ of the form

$$\mathbb{R}^d \ni x \mapsto \Phi(x,\theta) := \sum_{j=1}^{n} \theta_j^{(2)} \varrho\left(\left\langle \theta_j^{(1)}, \begin{bmatrix} x \\ 1 \end{bmatrix} \right\rangle\right), \tag{1.43}$$

where $\theta_j^{(1)} \in \mathbb{R}^{d+1}$ for $j \in [n]$, $\theta^{(2)} \in \mathbb{R}^n$, $\varrho$ is a Lipschitz continuous activation function, and we have augmented the vector $x$ by a constant 1 in the last coordinate

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

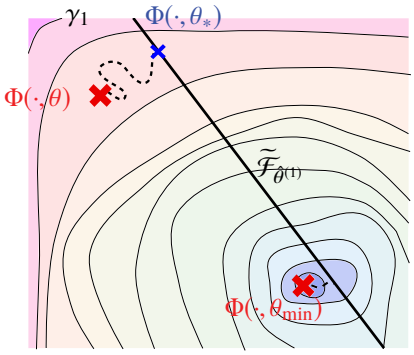



Figure 1.15 Construction of a path from an initial point $\theta$ to the global minimum $\theta_{\min}$ that does not have significantly higher risk than the initial point along the way. We depict here the landscape as a function of the neural network realizations rather than of their parametrizations, so that this landscape is convex.

as outlined in Remark 1.5. If we consider $\theta^{(1)}$ to be fixed then it is clear that the space

$$\widetilde{\mathcal{F}}_{\theta^{(1)}} := \{\Phi(\cdot,\theta)\colon \theta = (\theta^{(1)},\theta^{(2)}),\ \theta^{(2)} \in \mathbb{R}^n\} \tag{1.44}$$

is a linear space. If the risk[23] is convex, as is the case for the widely used quadratic or logistic losses, then this implies that $\theta^{(2)} \mapsto r\big((\theta^{(1)},\theta^{(2)})\big)$ 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 one may travel upon that do not increase the risk above the risk of the start and end points.

This idea was used in, for example, Freeman and Bruna (2017) in a way indicated by the following simple sketch. Assume that, for two parameters $\theta$ and $\theta_{\min}$, there exists a linear subspace of NNs $\widetilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$ such that there are paths $\gamma_1$ and $\gamma_2$ connecting $\Phi(\cdot,\theta)$ and $\Phi(\cdot,\theta_{\min})$ respectively to $\widetilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$. Further, assume that these paths are such that, along them, the risk does not significantly exceed $\max\{r(\theta), r(\theta_{\min})\}$. Figure 1.15 shows a visualization of these paths. In this case, a path from $\theta$ to $\theta_{\min}$ not significantly exceeding $r(\theta)$ along the way is found by concatenating the path $\gamma_1$, a path along $\widetilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$, and the path $\gamma_2$. By the previous discussion, we know that only $\gamma_1$ and $\gamma_2$ determine the extent to which the combined path exceeds $r(\theta)$ along its way. Hence, we need to ask about the existence of an $\widetilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$ that facilitates the construction of appropriate $\gamma_1$ and $\gamma_2$.

To understand why a good choice of $\widetilde{\mathcal{F}}_{\hat{\theta}^{(1)}}$, such that the risk along $\gamma_1$ and $\gamma_2$ will

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

not rise much higher than $r(\theta)$, is likely to be possible we set[24]

$$\hat{\theta}_j^{(1)} := \begin{cases} \theta_j^{(1)} & \text{for } j \in [n/2], \\ (\theta_{\min}^{(1)})_j & \text{for } j \in [n] \setminus [n/2]. \end{cases} \tag{1.45}$$

In other words, the first half of $\hat{\theta}^{(1)}$ is constructed 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., Corollary 4.2.13 of Vershynin, 2018), we expect that, for $\varepsilon > 0$ and a sufficiently large number of neurons $n$, the vectors $(\theta_j^{(1)})_{j=1}^{n/2}$ already $\varepsilon$-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 using 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

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

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 subnetwork 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, we have

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

yielding a path of constant risk between $(\theta_*^{(1)}, \theta_*^{(2)})$ and $(\hat{\theta}^{(1)}, \theta_*^{(2)})$. Connecting these paths completes the construction of $\gamma_1$ and shows that the risk along $\gamma_1$ does not exceed that at $\theta$ by more than $\mathcal{O}(\varepsilon)$. Of course, $\gamma_2$ can be constructed in the same way. The entire construction is depicted in Figure 1.15.

Overall, this derivation shows that for sufficiently wide NNs (appropriately randomly initialized) it is very likely possible to 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 Venturi et al. (2019), a similar approach is taken and the convexity in the last layer is used. However, the authors invoke the concept of intrinsic dimension to

[24] We assume, without loss of generality, that $n$ is a multiple of 2.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

solve elegantly the nonlinearity of $r((\theta^{(1)}, \theta^{(2)}))$ with respect to $\theta^{(1)}$. Also Safran and Shamir (2016) had already constructed 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 1 will decrease the risk, whereas choosing a number larger than 1 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, Nguyen and Hein (2017) showed 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 1.17. As in the case of two-layer NNs, the previous discussion on linear paths shows immediately that in this situation a monotone path to the global minimum exists.

### *1.5.2 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 1.16, we show the relative distance traveled by the parameters 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 has been explained theoretically: see e.g.,Du et al. (2018b, 2019), Li and Liang (2018), Allen-Zhu et al. (2019), and Zou et al. (2020). In §1.2.1, we have already given a high-level overview and, in particular, we discussed 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 the provable convergence of gradient descent for sufficiently overparametrized deep NNs.

**A simple learning model.** We consider again the simple NN model of (1.43) with a smooth activation function $\varrho$ which is not affine linear. For a 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

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

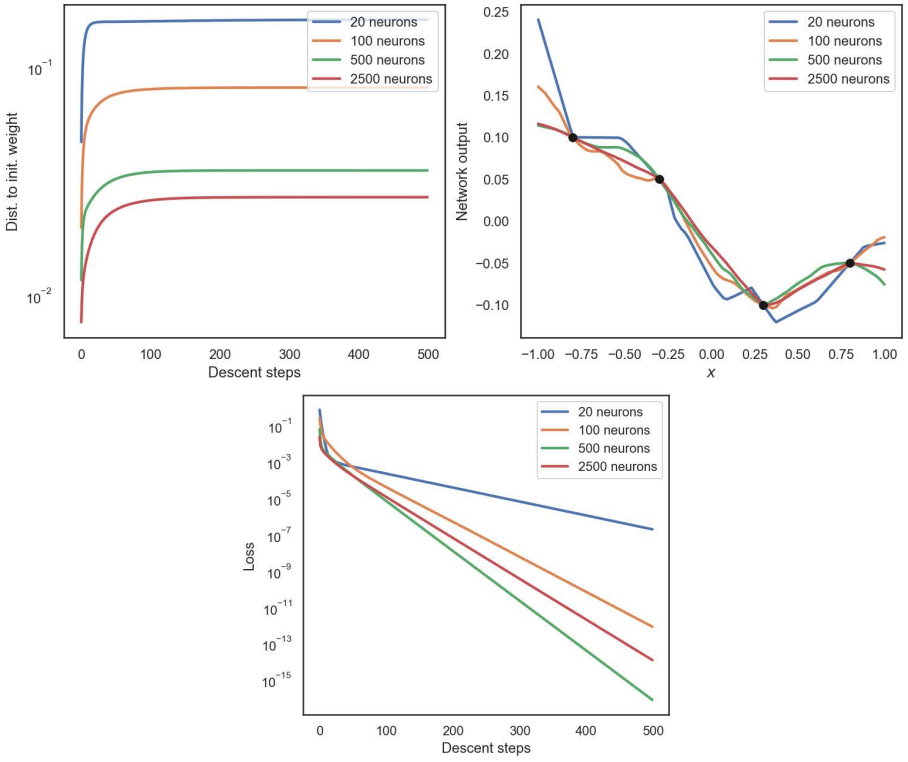



Figure 1.16 Four networks with architecture $((1, n, n, 1), \varrho_R)$ and $n \in \{20, 100, 500, 2500\}$ neurons per hidden layer were trained by gradient descent to fit the four points shown in the top right figure as black dots. We depict on the top left the relative Euclidean distance of the parameters from the initialization through the training process. In the top right, we show the final trained NNs. On the bottom we show the behavior of the training error.

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.

**A peculiar kernel.** Next we would like to understand what the gradient $\nabla_\theta r(\Theta)$ looks like, with high probability, over the initialization $\Theta = (\Theta^{(1)}, \Theta^{(2)})$. As with Equation (1.22), by restricting the gradient to $\theta^{(2)}$ and applying the chain rule, we have that

$$\begin{aligned} \|\nabla_\theta r(\Theta)\|_2^2 &\geq \frac{4}{m^2} \left\| \sum_{i=1}^m \nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta)(\Phi(x^{(i)}, \Theta) - y^{(i)}) \right\|_2^2 \\ &= \frac{4}{m^2} \left( (\Phi(x^{(i)}, \Theta) - y^{(i)})_{i=1}^m \right)^T \bar{K}_\Theta (\Phi(x^{(j)}, \Theta) - y^{(j)})_{j=1}^m, \end{aligned} \tag{1.46}$$

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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

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

This kernel is closely related to the neural tangent kernel in (1.23) evaluated at the features $(x^{(i)})_{i=1}^m$ and the random initialization $\Theta$. It is a slightly simplified version thereof because, in (1.23), the gradient is taken with respect to the full vector $\theta$. This can also be regarded as the kernel associated with a *random features model* (Rahimi et al., 2007).

Note that for our two-layer NN we have that

$$\left(\nabla_{\theta^{(2)}}\Phi(x,\Theta)\right)_k = \varrho\left(\left\langle \Theta_k^{(1)}, \begin{bmatrix} x \\ 1 \end{bmatrix}\right\rangle\right), \quad x\in\mathbb{R}^d,\ k\in[n]. \tag{1.47}$$

Thus, we can write $\bar{K}_\Theta$ as the following sum of (random) rank-1 matrices:

$$\bar{K}_\Theta = \sum_{k=1}^n v_k v_k^T \qquad \text{with} \quad v_k = \left(\varrho\left(\left\langle \Theta_k^{(1)}, \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix}\right\rangle\right)\right)_{i=1}^m \in\mathbb{R}^m, \quad k\in[n]. \tag{1.48}$$

The kernel $\bar{K}_\Theta$ is 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 1.17 shows that for $n=m$ the probability of that event is non-zero, say $\delta$, 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 (1.48) that $\mathbb{E}[\bar{K}_\Theta]$ scales linearly with $n$.

From this intuitive derivation we conclude that, for sufficiently large $n$, with high probability $\bar{K}_\Theta$ is a positive definite kernel with smallest eigenvalue $\lambda_{\min}(\bar{K}_\Theta)$ scaling linearly with $n$. The properties of $\bar{K}_\Theta$, in particular its positive definiteness, have been studied much more rigorously, as already described in §1.2.1.

**Control of the gradient.** Applying the expected behavior of the smallest eigenvalue $\lambda_{\min}(\bar{K}_\Theta)$ of $\bar{K}_\Theta$ to (1.46), we conclude that with high probability

$$\|\nabla_\theta r(\Theta)\|_2^2 \geq \frac{4}{m^2}\lambda_{\min}(\bar{K}_\Theta)\|(\Phi(x^{(i)},\Theta)-y^{(i)})_{i=1}^m\|_2^2 \gtrsim \frac{n}{m} r(\Theta). \tag{1.49}$$

To understand what will happen when applying gradient descent, we first need to understand how the situation changes in a neighborhood of $\Theta$. 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

$$\left\|\nabla_\theta\Phi(x,\Theta)-\nabla_\theta\Phi(x,\Theta+\bar{\theta})\right\|_2^2 \lesssim \sup_{\hat{\theta}\in B_1(0)} \left\|\nabla_\theta^2\Phi(x,\Theta+\hat{\theta})\right\|_{\mathrm{op}}^2, \tag{1.50}$$

where $\|\nabla_\theta^2\Phi(x,\Theta+\hat{\theta})\|_{\mathrm{op}}$ denotes the operator norm of the Hessian of $\Phi(x,\cdot)$ at $\Theta+\hat{\theta}$.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

By inspecting (1.43), it is not hard to see that, for all $i, j \in [n]$ and $k, \ell \in [d+1]$,

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

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 bandwidth $d+1$. Therefore we conclude that $\mathbb{E}\big[\|\nabla_\theta^2 \Phi(x,\Theta)\|_{\mathrm{op}}^2\big] \lesssim 1$. Hence we obtain by the concentration of Gaussian random variables that with high probability, $\|\nabla_\theta^2 \Phi(x,\Theta)\|_{\mathrm{op}}^2 \lesssim 1$. By the block-banded form of $\nabla_\theta^2 \Phi(x,\Theta)$ we have that, even after perturbation of $\Theta$ by a vector $\hat{\theta}$ with norm bounded by 1, the term $\|\nabla_\theta^2 \Phi(x,\Theta+\hat{\theta})\|_{\mathrm{op}}^2$ is still bounded, which yields that the right-hand side of (1.50) is bounded with high probability.

Using (1.50), we can extend (1.49), which holds with high probability, to a neighborhood of $\Theta$ by the following argument. Let $\bar{\theta} \in B_1(0)$; then

$$\begin{aligned}
\|\nabla_\theta r(\Theta+\bar{\theta})\|_2^2 &\geq \frac{4}{m^2}\Big\| \sum_{i=1}^m \nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta+\bar{\theta})(\Phi(x^{(i)}, \Theta+\bar{\theta}) - y^{(i)})\Big\|_2^2 \\
&\underset{(1.50)}{=} \frac{4}{m^2}\Big\| \sum_{i=1}^m (\nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta) + O(1))(\Phi(x^{(i)}, \Theta+\bar{\theta}) - y^{(i)})\Big\|_2^2 \\
&\underset{(*)}{\gtrsim} \frac{1}{m^2}(\lambda_{\min}(\bar{K}_\Theta) + O(1))\|(\Phi(x^{(i)}, \Theta+\bar{\theta}) - y^{(i)})_{i=1}^m\|_2^2 \\
&\gtrsim \frac{n}{m} r(\Theta+\bar{\theta}),
\end{aligned} \tag{1.51}$$

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

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

**Exponential convergence of gradient descent.** For sufficiently small step sizes $\eta$, the observation in the previous paragraph yields the following convergence rate for gradient descent as in Algorithm 1.1, specifically (1.8), with $m' = m$ and $\Theta^{(0)} = \Theta$:

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

if $\|\Theta^{(k)} - \Theta\| \leq 1$ for all $k \in [K+1]$, then[25]

$$r(\Theta^{(K+1)}) \approx r(\Theta^{(K)}) - \eta \|\nabla_\theta r(\Theta^{(K)})\|_2^2 \leq \left(1 - \frac{c\eta n}{m}\right) r(\Theta^{(K)}) \lesssim \left(1 - \frac{c\eta n}{m}\right)^K, \quad (1.52)$$

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

Let us assume without proof that the estimate (1.51) 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 have been shown with tools similar to those 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., $\lim_{t \to \infty}(1 - x/t)^t = e^{-x}$, yields, for sufficiently small $\eta$, that (1.52) is bounded by $e^{-c\sqrt{n/m}}$.

We conclude that, with high probability over the initialization, *gradient descent converges at an exponential rate to an arbitrarily 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 (Chizat et al., 2019).

Ideas similar to those above have led to groundbreaking convergence results of SGD for overparametrized NNs in much more complex and general settings; see, e.g., Du et al. (2018b), Li and Liang (2018), and Allen-Zhu et al. (2019).

In the infinite-width limit, NN training is practically equivalent to kernel regression; see §1.2.1. If we look at Figure 1.16 we see that the most overparametrized NN interpolates the data in the same way as a kernel-based interpolator would. In a sense, which was also highlighted in Chizat et al. (2019), this shows that, while overparametrized NNs in the lazy training regime have very nice properties, they essentially act like linear methods.

## 1.6 Tangible Effects of Special Architectures

In this section we describe results that isolate the effects of certain aspects of NN architectures. As discussed in §1.1.3, typically only either the depth or the number of parameters is used to study theoretical aspects of NNs. We have seen instances of this throughout §§1.3 and 1.4. Moreover, in §1.5, 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 §1.6.2 we introduce *skip connections* and in §1.6.3 we discuss a specific class

[25] Note that the step size $\eta$ needs to be small enough to facilitate the approximation step in (1.52). Hence, we cannot simply put $\eta = m/(cn)$ in (1.52) and have convergence after one step.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

of CNNs equipped with an encoder–decoder structure that is frequently used in image processing techniques. We introduce the *batch normalization block* in §1.6.4. Then, in §1.6.5, we discuss the *sparsely connected* NNs that typically result as an extraction from fully connected NNs. Finally, we briefly comment on recurrent neural networks in §1.6.6.

As we have noted repeatedly throughout this chapter, 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 have 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 by Bronstein et al. (2017) and Wu et al. (2021) for a discussion. Another highly successful type of architecture comprises *(variational) autoencoders* (Ackley et al., 1985; Hinton and Zemel, 1994). These are neural networks with a bottleneck that enforce a more efficient representation of the data. Similarly, *generative adversarial networks* (Goodfellow et al., 2014), which are composed of two neural networks – one generator and one discriminator – could not be discussed here. Yet 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 chapter. Instead, we refer to Wan et al. (2013), Srivastava et al. (2014), Haeffele and Vidal (2017), and Mianjy et al. (2018). Finally, the very successful *attention mechanism* (Bahdanau et al., 2015; Vaswani et al., 2017), which 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 that will be presented below have been developed on the basis of 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 building blocks thus simplified do not accurately reflect the practical applications of these tools in all use cases.

### *1.6.1 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. From this observation, it appears reasonable to have NNs that have local receptive fields in

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 the 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, the 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 LeCun et al. (1989a). 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 $\widetilde{G}$ be a subgroup of $G$, let $p\colon G \to \widetilde{G}$ 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

$$\mathbb{R}^G \ni x \mapsto x' := (p(x * \kappa_i))_{i=1}^C \in (\mathbb{R}^{\widetilde{G}})^C. \tag{1.53}$$

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

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

Popular alternatives are average pooling or max pooling. These operations then either compute 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 non-zero entries.

As explained earlier, a convolutional NN is built by stacking multiple convolutional blocks one after another.[26] At some point, the output can be *flattened*, i.e., mapped to a vector, and is then fed into an FC NN (see Definition 1.4). We depict this set-up in Figure 1.17.

[26] We 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 (1.53) acting on each coordinate. Moreover, one may also interject an activation function between the blocks.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

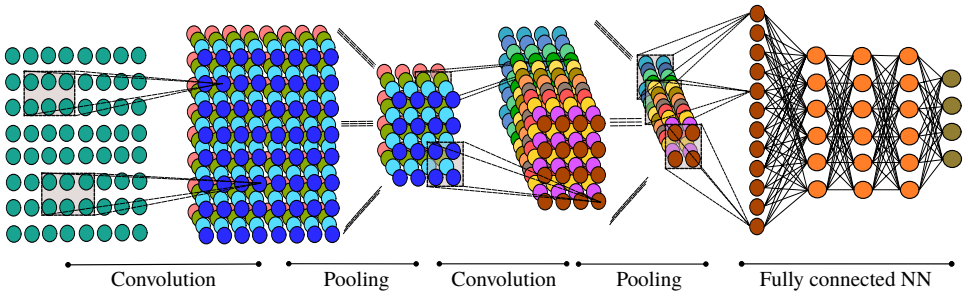



Figure 1.17 Illustration of a convolutional neural network with two-dimensional convolutional blocks and $2 \times 2$ subsampling as the pooling operation.

Owing to the fact that convolution is a linear operation, depending on the pooling operation, one may write a convolutional block (1.53) as an FC NN. For example, if $G = (\mathbb{Z}/(2d\mathbb{Z}))^2$ and the $2 \times 2$ 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., one 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 1.16. 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 (Oono and Suzuki, 2019; Zhou, 2020b). 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 (Yarotsky, 2018b; Petersen and Voigtlaender, 2020). Furthermore, Yarotsky (2018b) illuminates the effect of subsample pooling by showing that if no pooling is applied then universality cannot be achieved, whereas if pooling is applied then universality is possible. The effect of subsampling in CNNs from the viewpoint of approximation theory is further discussed in Zhou (2020a). The role of other types of pooling in enhancing invariances of the hypothesis set will be discussed in §1.7.1 below.

### *1.6.2 Residual Neural Networks*

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

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

It is clear that

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

Therefore, for sufficiently large $L$, we expect that $\left|\frac{\partial \pi}{\partial x_k}\right|$ 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. Owing to the aforementioned phenomenon, it is often observed that training the NNs suffers from either an exploding-gradient or a vanishing-gradient problem, which may prevent the 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 to the efficient training of deep NNs.

In addition to the exploding- and vanishing-gradient problems, there is an empirically observed *degradation problem* (He et al., 2016). This phrase describes the fact 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 $\varrho_R$ 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 $\varrho$ 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 (1.12). 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 He et al. (2016). Instead of defining an FC NN $\Phi$ as in (1.1), one can insert a residual block in the $\ell$th layer by redefining[27]

$$\bar{\Phi}^{(\ell)}(x, \theta) = \varrho(\Phi^{(\ell)}(x, \theta)) + \bar{\Phi}^{(\ell-1)}(x, \theta), \tag{1.54}$$

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 and is referred to as a skip connection or *residual connection*. A schematic diagram of a NN with residual blocks is shown in Figure 1.18. Inserting a residual block in all layers leads to a so-called *residual NN*.

A prominent approach to analyzing residual NNs is to establish a connection with optimal control problems and dynamical systems (E, 2017; Thorpe and van

[27] One can also skip multiple layers – e.g., in He et al. (2016) two or three layers were skipped – use a simple transformation instead of the identity (Srivastava et al., 2015), or randomly drop layers (Huang et al., 2016).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

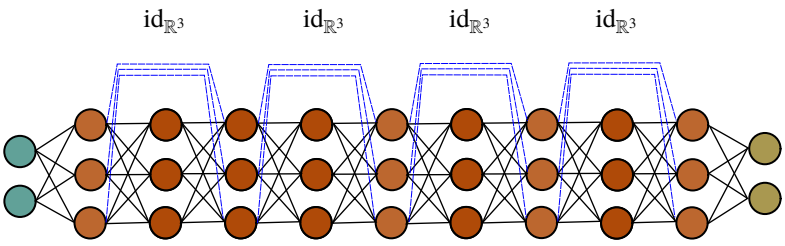



Figure 1.18 Illustration of a neural network with residual blocks.

Gennip, 2018; E et al., 2019b; Li et al., 2019b; Ruthotto and Haber, 2019; Lu et al., 2020). Concretely, if each layer of a NN $\Phi$ is of the form (1.54) then we have that

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

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

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

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.

### *1.6.3 Framelets and U-Nets*

One of the most prominent application areas of deep NNs is inverse problems, particularly those in the field of imaging science; see also §1.8.1. A specific architectural design of CNNs, namely so-called *U-nets*, introduced in Ronneberger et al. (2015), seems to perform best for this range of problems. We sketch a U-net in Figure 1.19. However, a theoretical understanding of the success of this architecture was lacking.

Recently, an innovative approach called *deep convolutional framelets* was suggested in Ye et al. (2018), which we now briefly explain. The core idea is to take a frame-theoretic viewpoint, see, e.g., Casazza et al. (2012), 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 §1.7.2 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

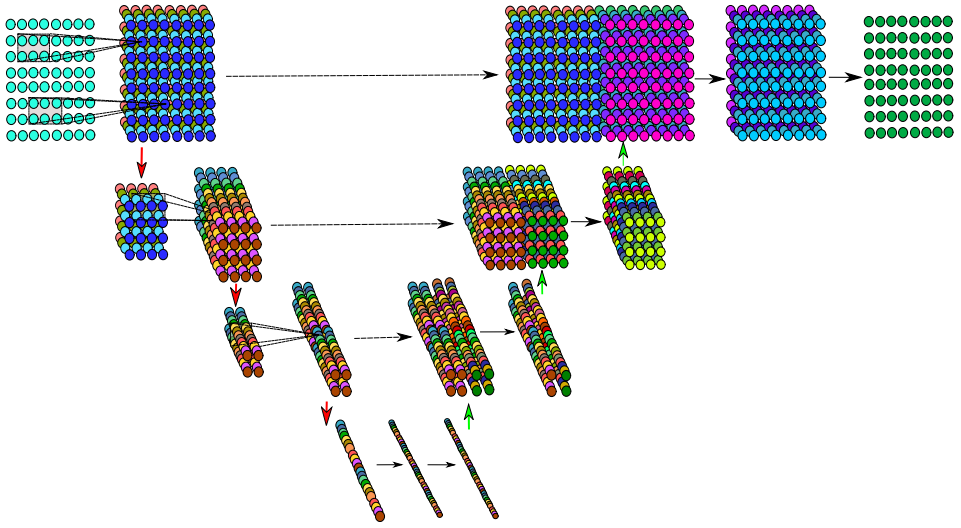


Figure 1.19 Illustration of a simplified U-net neural network. Down arrows stand for pooling, up arrows for deconvolution or upsampling, right-pointing arrows for convolution or fully connected steps. Lines without arrows are skip connections.

$G := \mathbb{Z}/(d\mathbb{Z})$ with kernels defined on the subgroup $H := \mathbb{Z}/(n\mathbb{Z})$, where $d, n \in \mathbb{N}$ with $n < d$; see also §1.6.1. 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 via[28]

$$\langle e^{(i)}, \mathbb{H}_n(x) v \rangle = \sum_{j \in H} x_{i+j} v_j = \langle x, e^{(i)} *_\circ v \rangle, \quad i \in G, \tag{1.55}$$

where $e^{(i)} := 1_{\{i\}} \in \mathbb{R}^G$ and $v \in \mathbb{R}^H$; see Yin et al. (2017). 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.,

$$\frac{1}{n} \mathrm{Tr}\big(\mathbb{H}_n(e^{(k)})^T \mathbb{H}_n(x)\big) = \frac{1}{n} \sum_{j \in H} \sum_{i \in G} e^{(k)}_{i+j} x_{i+j} = \frac{1}{n} |H| x_k = x_k. \tag{1.56}$$

This allows us to construct global and local bases as follows. Let $p, q \in \mathbb{N}$, let $U = \begin{bmatrix} u_1 \cdots u_p \end{bmatrix} \in \mathbb{R}^{d \times p}$, $V = \begin{bmatrix} v_1 \cdots v_q \end{bmatrix} \in \mathbb{R}^{n \times q}$, $\widetilde{U} = \begin{bmatrix} \tilde{u}_1 \cdots \tilde{u}_p \end{bmatrix} \in \mathbb{R}^{d \times p}$, and

[28] Here 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 $\mathbb{R}^n$.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

$\widetilde{V} = \left[\tilde{v}_1 \cdots \tilde{v}_q\right] \in \mathbb{R}^{n\times q}$, and assume that

$$\mathbb{H}_n(x) = \widetilde{U}U^T\mathbb{H}_n(x)V\widetilde{V}^T. \tag{1.57}$$

For $p \geq d$ and $q \geq n$, this is satisfied if, for instance, $U$ and $V$ constitute frames whose dual frames are respectively $\widetilde{U}$ and $\widetilde{V}$, i.e., $\widetilde{U}U^T = \mathrm{I}_d$ and $V\widetilde{V}^T = \mathrm{I}_n$. As a special case, one can consider orthonormal bases $U = \widetilde{U}$ and $V = \widetilde{V}$ 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 (1.57) by choosing the left and right singular vectors of $\mathbb{H}_n(x)$ as $U = \widetilde{U}$ and $V = \widetilde{V}$, respectively. The identity in (1.57) in turn ensures the following decomposition:

$$x = \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{q}\langle x, u_i *_\circ v_j\rangle \tilde{u}_i *_\circ \tilde{v}_j. \tag{1.58}$$

Observing that the vector $v_j \in \mathbb{R}^H$ interacts locally with $x \in \mathbb{R}^G$ owing 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 a local basis and $(u_i)_{i=1}^p$ as a global basis. In the context of CNNs, $v_i$ can be interpreted as a local convolutional kernel and $u_i$ as a pooling operation.[29] The proof of (1.58) follows directly from properties (1.55), (1.56), and (1.57):

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

The decomposition in (1.58) can now be interpreted as the composition of an encoder and a decoder,

$$x \mapsto C = (\langle x, u_i *_\circ v_j\rangle)_{i\in[p],j\in[q]} \quad \text{and} \quad C \mapsto \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{q} C_{i,j}\tilde{u}_i *_\circ \tilde{v}_j, \tag{1.59}$$

which relates it to CNNs equipped with an encoder–decoder structure such as U-nets; see Figure 1.19. Generalizing this approach to multiple channels, it is possible to stack such encoders and decoders leading to a layered version of (1.58). Ye et al. (2018) show that one can make an informed decision on the number of layers on the basis of the rank of $\mathbb{H}_n(x)$, i.e., the complexity of the input features $x$. Moreover, an activation function such as the ReLU or bias vectors can also be included. The key question one can then ask is how the kernels can be chosen to obtain sparse coefficients $C$ in (1.59) and a decomposition such as (1.58), i.e., perfect

[29] Note 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, for example TensorFlow and PyTorch, where typically cross-correlations are used instead of convolutions.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

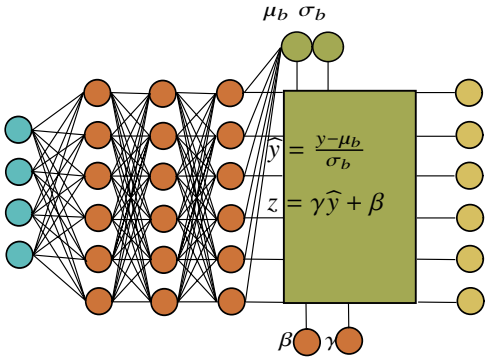



Figure 1.20 A batch normalization block after a fully connected neural network. The parameters $\mu_b, \sigma_b$ are the mean and the standard deviation of the output of the fully connected network computed over a batch $s$, i.e., a set of inputs. The parameters $\beta, \gamma$ are learnable parts of the batch normalization block.

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 (1.58) owing 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 in obtaining a perfect reconstruction.

### 1.6.4 Batch Normalization

Batch normalization involves a building block of NNs that was invented in Ioffe and Szegedy (2015) with the goal of reducing so-called *internal covariance shift*. In essence, this phrase describes the (undesirable) situation where, during training, each layer receives inputs with different distributions. 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

$$\begin{aligned} &\mathrm{BN}_b^{(\beta,\gamma)}(y) := \gamma \frac{y - \mu_b}{\sigma_b} + \beta, \quad y \in \mathbb{R}^n, \\ &\text{with} \quad \mu_b = \frac{1}{m}\sum_{i=1}^m y^{(i)} \quad \text{and} \quad \sigma_b^2 = \frac{1}{m}\sum_{i=1}^m (y^{(i)} - \mu_b)^2, \end{aligned} \tag{1.60}$$

where all operations are to be understood componentwise; see Figure 1.20.

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 data.[30]

[30] In practice, one typically uses a moving average to estimate the mean $\mu$, and the standard deviation $\sigma$ of the output of the previous layer, over the whole training data by using only batches.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

Furthermore, the parameters $\beta, \gamma$ are variable and can be learned during training. Note that if one sets $\beta = \mu_b$ and $\gamma = \sigma_b$ then $\mathrm{BN}_b^{(\beta,\gamma)}(y) = y$ 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 Santurkar et al. (2018, Theorem 4.1), the following observation was made.

**Proposition 1.28** (Smoothening effect of batch normalization)**.** *Let $m \in \mathbb{N}$ with $m \geq 2$, and for every $\beta, \gamma \in \mathbb{R}$ define $\mathcal{B}^{(\beta,\gamma)} \colon \mathbb{R}^m \to \mathbb{R}^m$ by*

$$\mathcal{B}^{(\beta,\gamma)}(b) = (\mathrm{BN}_b^{(\beta,\gamma)}(y^{(1)}), \dots, \mathrm{BN}_b^{(\beta,\gamma)}(y^{(m)})), \quad b = (y^{(i)})_{i=1}^m \in \mathbb{R}^m, \tag{1.61}$$

*where $\mathrm{BN}_b^{(\beta,\gamma)}$ is as given in* (1.60)*. Let $\beta, \gamma \in \mathbb{R}$ and let $r \colon \mathbb{R}^m \to \mathbb{R}$ be a differentiable function. Then, for every $b \in \mathbb{R}^m$, we have*

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

*where $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^m$ and $\sigma_b^2$ is as given in* (1.60)*.*

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 1.28 follows from a convenient representation of the Jacobian of the mapping $\mathcal{B}^{(\beta,\gamma)}$, given by

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

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

Choosing $r$ to mimic the empirical risk of a learning task, Proposition 1.28 shows that, in certain situations – for instance, if $\gamma$ is smaller than $\sigma_b$ 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 (Santurkar et al., 2018, 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 $\beta$ and $\gamma$ were introduced, including a batch normalization block also increases the dimension of the optimization problem by 2.

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

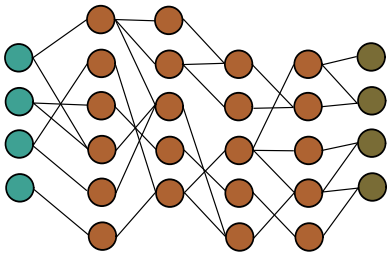


Figure 1.21 A neural network with sparse connections.

### 1.6.5 Sparse Neural Networks and Pruning

For deep FC NNs, the number of trainable parameters usually scales as the square of the number of neurons. For reasons of computational complexity and memory efficiency, it appears sensible to seek for techniques that reduce the number of parameters or extract *sparse subnetworks* (see Figure 1.21) without much affecting the output of a NN. One way to do it is by *pruning* (LeCun et al., 1989b; Han et al., 2016). Here, certain parameters of a NN are removed after training. This is done by, for example, setting these parameters to zero.

In this context, the *lottery ticket hypothesis* was formulated in Frankle and Carbin (2018). It states that "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 Ramanujan et al. (2020) a similar hypothesis was made and empirically studied. There, it was 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., it already does so at initialization.

Under certain simplifying assumptions, the existence of favorable subnetworks is quite easy to prove. We can use a technique that was indirectly used in §1.4.2 – the Carathéodory lemma. This result states the following. Let $n \in \mathbb{N}$, $C \in (0, \infty)$, and $(\mathcal{H}, \|\cdot\|)$ 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., Vershynin (2018, Theorem 0.0.2).

**Proposition 1.29** (Carathéodory pruning)**.** *Let $d, n \in \mathbb{N}$ with $n \geq 100$ and let $\mu$ 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*

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

*parameters such that*

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

*where* $\big(w_i^{(1)}, b_i^{(1)}\big) \in \mathbb{R}^d \times \mathbb{R}$, $i \in [n]$, *and* $w^{(2)} \in \mathbb{R}^n$. *Assume that for every* $i \in [n]$ *it holds true 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 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*

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

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

$$f_i := \|w^{(2)}\|_1 \varrho_R(\langle w_i^{(1)}, \cdot\rangle + b_i^{(1)}), \quad i \in [n], \tag{1.62}$$

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, we have

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

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

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

which yields the result.

Proposition 1.29 shows that certain, very wide NNs can be approximated very well by sparse subnetworks in which only the output weight matrix needs to be changed. The argument of Proposition 1.29 was inspired by Barron and Klusowski (2018), where a much more refined result is shown for deep NNs.

### *1.6.6 Recurrent Neural Networks*

Recurrent NNs are NNs where the underlying graph is allowed to exhibit cycles, as in Figure 1.22; see Hopfield (1982), Rumelhart et al. (1986), Elman (1990), and Jordan (1990). Above, we 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 the previous layers, the order

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

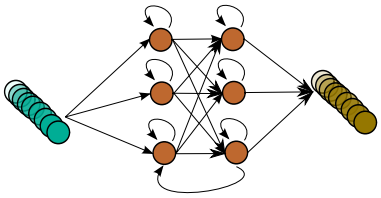


Figure 1.22 Sketch of a recurrent neural network. The cycles in the computational graph incorporate the sequential structure of the input and output.

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 in a way similar to regular feed-forward NNs by an algorithm called *backpropagation through time* (Minsky and Papert, 1969; Werbos, 1988; Williams and Zipser, 1995). This procedure essentially unfolds the recurrent structure to yield a classical NN structure. However, the algorithm may lead to very deep structures. Owing to the vanishing- and exploding-gradient problem discussed earlier, very deep NNs are often hard to train. Because of this, special recurrent structures have been introduced that include gates that prohibit too many recurrent steps; these include the widely used long short-term memory gates, LSTMs, (Hochreiter and Schmidhuber, 1997).

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 Bohn and Feischl (2019, 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 1.9.

On the other hand, with feed-forward NNs, it transpires from Theorem 1.26 that

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

the approximation error of fixed-sized ReLU NNs for any non-affine function is greater than a positive lower bound.

## 1.7 Describing the Features that a Deep Neural Network Learns

This section presents two viewpoints which help in understanding the nature of the features that can be described by NNs. Section 1.7.1 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 1.7.2 relates NN features to the current paradigm of *sparse coding*.

### *1.7.1 Invariances and the Scattering Transform*

One of the first theoretical contributions to the understanding of the mathematical properties of CNNs was by Mallat (2012). Their approach was 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 $\Psi$ 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 $\Psi\colon L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$, $u \mapsto |\hat{u}|$. 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 (Grohs et al., 2020), $u$ can be reconstructed up to a translation from its Fourier modulus (Grohs et al., 2020) 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 (Mallat, 2016). This program was carried out by Mallat (2012) for additional invariances with respect to deformations $u \mapsto u_\tau := u(\cdot - \tau(\cdot))$, where $\tau\colon \mathbb{R}^d \to \mathbb{R}^d$ is a smooth mapping. Such transformations may occur in practice,

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

for instance, as image warpings. In particular, a feature descriptor $\Psi$ is designed so that, with a suitable norm $\|\cdot\|$ on the image of $\Psi$, it

(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 $\tau$ 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. Thus, a deformation invariant feature descriptor represents a big potential for dimensionality reduction. Roughly speaking, the feature descriptor $\Psi$ of Mallat (2012) (also coined the *scattering transform*) is defined by collecting features that are computed by iteratively applying a wavelet transform followed by a pointwise modulus nonlinearity and a subsequent low-pass filtering step, i.e.,

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

where $\psi_j$ refers to a wavelet at scale $j$ and $\varphi_J$ 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 listed above in a suitable (asymptotic) sense. The proof of this result relies on a subtle interplay between the "deformation covariance" property of the wavelet transform and the "regularizing" property of the operation of convolution with the modulus of a wavelet. For a much more detailed exposition of the resulting scattering transform, we refer to Chapter 8 in this book. We remark that similar results can be shown also for different systems, such as Gabor frames (Wiatowski et al., 2017; Czaja and Li, 2019).

### 1.7.2 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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

system. For a vector $x \in \mathbb{R}^d$, a sparsifying representation system $D \in \mathbb{R}^{d\times p}$ – also called a *dictionary* – is such that $x = D\phi$ where the coefficients $\phi \in \mathbb{R}^p$ are sparse in the sense that $\|\phi\|_0 := |\operatorname{supp}(\phi)| = |\{i \in [p] \colon \phi_i \neq 0\}|$ is small compared with $p$. 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 that 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* (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 §1.6.1. The corresponding CSC model is based on the decomposition of a global signal $x \in (\mathbb{R}^G)^c$ with $c \in \mathbb{N}$ channels as

$$x_i = \sum_{j=1}^{C} \kappa_{i,j} * \phi_j, \quad i \in [c], \tag{1.63}$$

where $\phi \in (\mathbb{R}^G)^C$ is taken 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 $\operatorname{supp}(\kappa_{i,j}) \subset H$. 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 (1.63) imply that this patch is affected only by a stripe of at most $(2n-1)$ entries in each channel of $\phi$. The local, patch-based sparsity of the representation $\phi$ 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 Papyan et al. (2017b). Furthermore, note that we can naturally identify $x$ and $\phi$ 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 (Papyan et al., 2017a; Sulam et al., 2018; Papyan et al., 2018). 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* (ML-CSC) *model* if there exist coefficients $\phi^{(\ell)} \in \mathbb{R}^{dC_\ell}$ with

$$\phi^{(\ell-1)} = D^{(\ell)}\phi^{(\ell)} \quad \text{and} \quad \|\phi^{(\ell)}\|_{0,\infty}^{(n_\ell)} \leq k_\ell, \quad \ell \in [L]. \tag{1.64}$$

We now consider the problem of reconstructing the sparse coefficients $(\phi^{(\ell)})_{\ell=1}^L$

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

from a noisy signal $\tilde{x} := x + \nu$, where the noise $\nu \in \mathbb{R}^{dC_0}$ is assumed to have small $\ell^2$-norm and $x$ is assumed to follow the ML-CSC model in (1.64). In general, this problem is NP-hard. However, under suitable conditions on the ML-CSC model, it can be solved approximately, 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

$$\mathcal{T}_{D,b}(x) := \varrho_R(D^T x - b) - \varrho_R(-D^T x - b), \quad x \in \mathbb{R}^{dc}, \tag{1.65}$$

where $\varrho_R(x) = \max\{0, x\}$ is applied componentwise. If $x = D\phi$ as in (1.63), we obtain $\phi \approx \mathcal{T}_{D,b}(x)$ roughly under the following conditions. The distance of $\phi$ from $\psi := D^T x = D^T D\phi$ can be bounded using the local sparsity of $\phi$ 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 $\phi$ by nullifying those entries of $\psi$ with $\psi_i \leq |b_i|$. Utilizing the soft-thresholding operator (1.65) 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 to 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 an ReLU nonlinearity in the $\ell$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 then have the following flavor. Given a suitable reconstruction procedure such as thresholding or $\ell_1$-minimization, the sparse coefficients $(\phi^{(\ell)})_{\ell=1}^L$ of a signal $x$ following an ML-CSC model can be stably recovered from the noisy signal $\tilde{x}$ under certain hypotheses on the ingredients of the ML-CSC model.

## 1.8 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 (Adcock and Dexter, 2020).

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

In this section, we consider concrete applications which have been very successfully solved with deep-learning-based methods. In §1.8.1 we present an overview of deep-learning-based algorithms applied to inverse problems. Section 1.8.2 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.

### *1.8.1 Deep Neural Networks Meet Inverse Problems*

The area of inverse problems, predominantly in imaging, was probably the first class of mathematical methods embracing deep learning with overwhelming success. Let us consider a forward operator $K\colon \mathcal{Y} \to \mathcal{X}$ where $\mathcal{X}, \mathcal{Y}$ are 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 the well-posedness of the approximation and its 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 (Jin et al., 2017a) – 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

$$\|Ky - x\|^2 + \alpha R(y), \tag{1.66}$$

where $\alpha \in (0, \infty)$ is a regularization parameter, $R\colon \mathcal{Y} \to [0, \infty)$ is a regularization term, and $\|\cdot\|$ denotes the norm on $\mathcal{Y}$. As already stated, 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 allow, for instance, 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 that are based on deep learning techniques can be roughly categorized into three classes:

(i) *Supervised approaches*. The most straightforward approach is to train a NN $\Phi(\cdot, \theta)\colon \mathcal{X} \to \mathcal{Y}$ 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 as in Adler and Öktem (2017), Gilton et al. (2019), and Monga et al. (2021). Yet another type of approach aims to combine

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

deep NNs with classical model-based approaches. The first suggestion in this realm was that one should start by applying a standard solver and then use a deep NN, $\Phi(\cdot,\theta)\colon \mathcal{Y} \to \mathcal{Y}$, which serves as a denoiser for specific reconstruction artifacts; e.g., Jin et al. (2017b). This approach was followed by more sophisticated methods such as plug-and-play frameworks for coupling inversion and denoising (Romano et al., 2017).

(ii) *Semi-supervised approaches*. This type of approach aims to encode the regularization by a deep NN $\Phi(\cdot,\theta)\colon \mathcal{Y} \to [0,\infty)$. The underlying idea often requires stronger regularization on those 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 requires only a set of labels $(y^{(i)})_{i=1}^m$ as well as a method for assessing 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 Lunz et al. (2018), and Li et al. (2020). Taking a Bayesian viewpoint, one can also learn prior distributions as deep NNs; this was done in Barbano et al. (2020).

(iii) *Unsupervised approaches*. One highlight of what we might call unsupervised approaches in our problem setting has been the introduction of deep image priors in Dittmer et al. (2020), and Ulyanov et al. (2018). The key idea is to parametrize the solutions $y$ as the output of a NN $\Phi(\xi,\cdot)\colon \mathcal{P} \to \mathcal{Y}$ with parameters in a suitable space $\mathcal{P}$ applied to a fixed input $\xi$. 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 early stopping is often 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 the availability and accuracy of training data, properties of the forward operator, and requirements for the solution. And each of the three classes of hybrid solvers follows a different strategy.

Let us now discuss the 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 use of synthetic data as a significant part of the training data. Thus the learned NN will mainly perform as well as the algorithm which generated the data, but will not significantly improve on it – except 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 $K$, 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 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.

### *1.8.2 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 examples from three problem classes.

(i) *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 multi-level Picard algorithms E et al., 2019c) no general algorithm exists that provably scales well with the dimension.

(ii) *Schrödinger equation*. The electronic Schrödinger equation describes the stationary non-relativistic behavior of a quantum mechanical electron system in the electric field generated by the nuclei of a molecule. A 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,

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

when applied to large delocalized molecules, rendering these methods useless for those problems.

(iii) *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. This 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.

Thanks to the favorable approximation results of NNs for high-dimensional functions (see especially §§1.4.3), it might not come as a surprise that a NN ansatz has proven to be quite successful in solving the aforementioned PDE models. Pioneering work in this direction was by Han et al. (2018) who used the backwards SDE reformulation of semi-linear 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 Nakamura-Zimmerer et al. (2021). In this work, the Pontryagin principle was used to generate samples of the PDE solution along solutions of the corresponding boundary value problem. Other numerical approaches include the *deep Ritz method* (E and Yu, 2018), where a Dirichlet energy is minimized over a set of NNs; or so-called *physics informed neural networks* (Raissi et al., 2019), 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 non-zero eigenvalue of the eigenvalue problem

$$\mathcal{H}_R\psi = \lambda_\psi\psi, \tag{1.67}$$

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

for $\psi : \mathbb{R}^{3\times n} \to \mathbb{R}$, 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 the Coulomb attraction force between electrons and nuclei (second and third terms) and the Coulomb repulsion force between different electrons (fourth term). Here, the coordinates $R = \left[R_1, \ldots, R_p\right] \in \mathbb{R}^{3\times p}$ 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 $r = \left[r_1, \ldots, r_n\right] \in \mathbb{R}^{3\times n}$ refer to the positions of the electrons. The associated eigenfunction $\psi$ describes the so-called *wave function*, 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 $\lambda_\psi$ of (1.67) 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 (1.67) is complicated by the *Pauli principle*, which states that the wave function $\psi$ must be antisymmetric in all coordinates representing electrons of equal spin. We need to clarify that every electron is defined not only 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 considerably. This is reflected in the Pauli principle mentioned above. Suppose that electrons $i$ and $j$ have equal spin; then the wave function must satisfy

$$P_{i,j}\psi = -\psi, \tag{1.68}$$

where $P_{i,j}$ denotes the operator that swaps $r_i$ and $r_j$, i.e.,

$$(P_{i,j}\psi)(r) = \psi(r_1, \ldots, r_j, \ldots, r_i, \ldots, r_n).$$

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 of Paul Dirac (1929):

"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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

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 non-zero eigenvalue of (1.67) can be reformulated as minimizing the Rayleigh quotient

$$\frac{\int_{\mathbb{R}^{3\times n}} \overline{\psi(r)}(\mathcal{H}_R\psi)(r)\,\mathrm{d}r}{\|\psi\|_{L^2}^2} = \mathbb{E}\left[\frac{(\mathcal{H}_R\psi)(X)}{\psi(X)}\right] \tag{1.69}$$

over all $\psi$'s satisfying the Pauli principle; see Szabo and Ostlund (2012). 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 sampling.[31] Since the wave function $\psi$ will be parametrized as a NN, the minimization of (1.69) will require the computation of the gradient of (1.69) with respect to the NN parameters (the method in Pfau et al., 2020, 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 $\psi$; see, for example Hermann et al. (2020, Equation (8)).

Compared with 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 of electrons close to a nucleus into the NN architecture. A first attempt in this direction was made by Han et al. (2019), and significantly improved NN architectures have been developed in Hermann et al. (2020), Pfau et al. (2020), and Scherbela et al. (2021) 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 Hermann et al. (2020); Scherbela et al. (2021). Standard numerical approaches (notably the multireference Hartree–Fock method; see Szabo and Ostlund, 2012) use a low-rank approach to minimize (1.69). Such an approach would approximate $\psi$ by sums of products of *one-electron orbitals* $\prod_{i=1}^n \varphi_i(r_i)$ but clearly this would not satisfy the Pauli principle (1.68). In order to accommodate 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

$$\det\left(\left(\varphi_i(r_j)\right)_{i,j=1}^{n_+}\right) \times \det\left(\left(\varphi_i(r_j)\right)_{i,j=n_++1}^{n}\right) \tag{1.70}$$

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

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

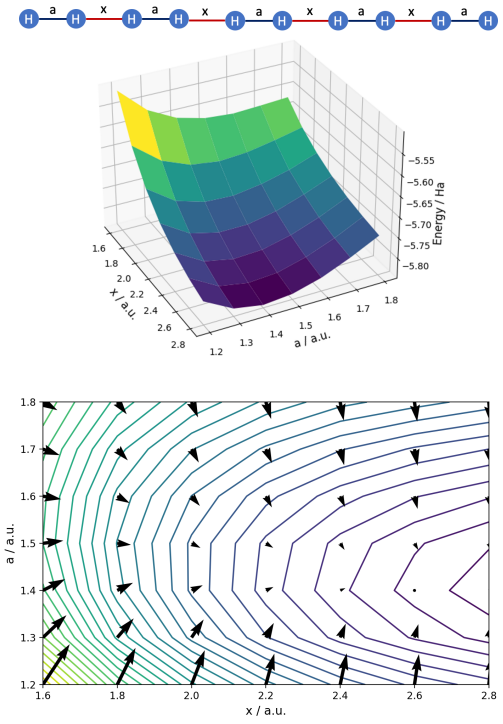



Figure 1.23 By sharing layers across different nuclear geometries one can efficiently compute different geometries in one single training step (Scherbela et al., 2021). Top: potential energy surface of an $H_{10}$ chain computed by the deep-learning-based algorithm from Scherbela et al. (2021). The lowest energy is achieved when pairs of H atoms enter into a covalent bond to form five $H_2$ molecules. Bottom: the method of Scherbela et al. (2021) is capable of accurately computing forces between nuclei, which allows for molecular dynamics simulations from first principles.

satisfies (1.68) and is typically called a Slater determinant. While the Pauli principle establishes a (non-classical) interaction between electrons of equal spin, the so-called *exchange correlation*, in the representation (1.70) electrons with opposite spins are uncorrelated. In particular, (1.70) ignores interactions between electrons that arise through Coulomb forces, implying that no non-trivial wave function can be accurately represented by a single Slater determinant. To capture the 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 our knowledge the best currently known approximation results are contained in Yserentant (2010), 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 Hermann et al. (2020), whose main

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

idea was to use NNs to incorporate interactions into Slater determinants of the form (1.70) using what is called the *backflow trick* (Ríos et al., 2006). The basic building blocks would now consist of functions of the form

$$\det\left(\left(\varphi_i(r_j)\Psi_j(r,\theta_j)\right)_{i,j=1}^{n_+}\right)\times\det\left(\left(\varphi_i(r_j)\Psi_j(r,\theta_j)\right)_{i,j=n_++1}^{n}\right), \tag{1.71}$$

where the $\Psi_k(\cdot,\theta_k)$, $k\in[n]$, are NNs. If these are arbitrary NNs, it is easy to see that the Pauli principle (1.68) 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 true that

$$P_{i,j}\Psi_k(\cdot,\theta_k)=\begin{cases}\Psi_k(\cdot,\theta_k), & \text{if } k\notin\{i,j\},\\ \Psi_i(\cdot,\theta_i), & \text{if } k=j,\\ \Psi_j(\cdot,\theta_j), & \text{if } k=i,\end{cases} \tag{1.72}$$

and analogous conditions hold for $i,j,k\in[n]\setminus[n_+]$, the expression (1.71) does actually satisfy (1.68). The construction of such symmetric NNs can be achieved by using a modification of the so-called *SchNet architecture* (Schütt et al., 2017) which can be considered as a specific residual NN.

We describe a simplified construction inspired by Han et al. (2019) and used in a slightly more complex form in Scherbela et al. (2021). We restrict ourselves to the case of positive spin (for example, 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

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

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

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

where $\Gamma_k(\cdot,\theta_{k,\text{fc}})$ denotes a standard FC NN with input dimension equal to the output dimension of $\Psi^+$ plus the dimension of the negative-spin electrons. The networks $\Psi_k$, $k\in[n]\setminus[n_+]$, are defined analogously using different parameters $\theta^-_{\text{emb}}$ for the embeddings. It is straightforward to check that the NNs $\Psi_k$, $k\in[n]$, satisfy (1.72) so that the backflow determinants (1.71) satisfy the Pauli principle (1.68).

In Hermann et al. (2020) the backflow determinants (1.71) 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 Hermann et al. (2020) show that this ansatz (namely using linear combinations of backflow determinants (1.71) instead of plain Slater determinants (1.70)) is vastly more efficient in terms of the number of determinants needed to obtain chemical accuracy. The full architecture provides

https://doi.org/10.1017/9781009025096.002 Published online by Cambridge University Press

a general purpose NN architecture to represent complicated wave functions. A distinct advantage of this approach is that some parameters (for example, regarding the 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 (Scherbela et al., 2021); see Figure 1.23.

Finally, we would like to highlight the need for 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.

## Acknowledgments

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 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 chapter. In particular, Dennis Elbrächter provided help for several theoretical results.

## References

Ackley, David H., Hinton, Geoffrey E., and Sejnowski, Terrence J. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science*, **9**(1), 147–169.

Adcock, Ben, and Dexter, Nick. 2020. The gap between theory and practice in function approximation with deep neural networks. ArXiv preprint arXiv:2001.07523.

Adler, Jonas, and Öktem, Ozan. 2017. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, **33**(12), 124007.

Al-Hamdani, Yasmine S., Nagy, Péter R., Barton, Dennis, Kállay, Mihály, Brandenburg, Jan Gerit, and Tkatchenko, Alexandre. 2020. Interactions between large molecules: Puzzle for reference quantum-mechanical methods. ArXiv preprint arXiv:2009.08927.

Allen-Zhu, Zeyuan, Li, Yuanzhi, and Song, Zhao. 2019. A convergence theory for deep learning via over-parameterization. Pages 242–252 of: *Proc. International Conference on Machine Learning*.

Anthony, Martin, and Bartlett, Peter L. 1999. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.

Arora, Sanjeev, Cohen, Nadav, and Hazan, Elad. 2018a. On the optimization of

deep networks: Implicit acceleration by overparameterization. Pages 372–389 of: *Proc. International Conference on Machine Learning*.

Arora, Sanjeev, Ge, Rong, Neyshabur, Behnam, and Zhang, Yi. 2018b. Stronger generalization bounds for deep nets via a compression approach. Pages 254–263 of: *Proc. International Conference on Machine Learning*.

Arora, Sanjeev, Cohen, Nadav, Golowich, Noah, and Hu, Wei. 2019a. A convergence analysis of gradient descent for deep linear neural networks. In: *International Conference on Learning Representations*.

Arora, Sanjeev, Du, Simon S., Hu, Wei, Li, Zhiyuan, Salakhutdinov, Ruslan, and Wang, Ruosong. 2019b. On exact computation with an infinitely wide neural net. Pages 8139–8148 of: *Advances in Neural Information Processing Systems*.

Arridge, Simon, Maass, Peter, Öktem, Ozan, and Schönlieb, Carola-Bibiane. 2019. Solving inverse problems using data-driven models. *Acta Numerica*, **28**, 1–174.

Auer, Peter, Herbster, Mark, and Warmuth, Manfred K. 1996. Exponentially many local minima for single neurons. Page 316–322 of: *Advances in Neural Information Processing Systems*.

Auffinger, Antonio, Arous, Gérard Ben, and Černỳ, Jiří. 2013. Random matrices and complexity of spin glasses. *Communications on Pure and Applied Mathematics*, **66**(2), 165–201.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. 2015. Neural machine translation by jointly learning to align and translate. In: *Proc. International Conference on Learning Representations*.

Baldi, Pierre, Sadowski, Peter, and Whiteson, Daniel. 2014. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, **5**(1), 1–9.

Barbano, Riccardo, Zhang, Chen, Arridge, Simon, and Jin, Bangti. 2020. Quantifying model uncertainty in inverse problems via Bayesian deep gradient descent. ArXiv preprint arXiv:2007.09971.

Barron, Andrew R. 1992. Neural net approximation. Pages 69–72 of: *Proc. Yale Workshop on Adaptive and Learning Systems*, vol. 1.

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

Barron, Andrew R., and Klusowski, Jason M. 2018. Approximation and estimation for high-dimensional deep learning networks. ArXiv preprint arXiv:1809.03090.

Bartlett, Peter L. 1998. 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**(2), 525–536.

Bartlett, Peter L, Maiorov, Vitaly, and Meir, Ron. 1998. Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural Computation*, **10**(8), 2159–2173.

Bartlett, Peter L., Bousquet, Olivier, and Mendelson, Shahar. 2005. Local Rademacher complexities. *Annals of Statistics*, **33**(4), 1497–1537.

Bartlett, Peter L., Foster, Dylan J., and Telgarsky, Matus. 2017. Spectrally-normalized margin bounds for neural networks. Pages 6240–6249 of: *Advances in Neural Information Processing Systems*.

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

Bartlett, Peter L., Long, Philip M., Lugosi, Gábor, and Tsigler, Alexander. 2020. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, **117**(48), 30063–30070.

Baum, Eric B., and Haussler, David. 1989. What size net gives valid generalization? *Neural Computation*, **1**(1), 151–160.

Beck, Christian, Becker, Sebastian, Grohs, Philipp, Jaafari, Nor, and Jentzen, Arnulf. 2021. Solving the Kolmogorov PDE by means of deep learning. *Journal of Scientific Computing*, **83**(3), 1–28.

Belkin, Mikhail, Ma, Siyuan, and Mandal, Soumik. 2018. To understand deep learning we need to understand kernel learning. Pages 541–549 of: *Proc. International Conference on Machine Learning*.

Belkin, Mikhail, Rakhlin, Alexander, and Tsybakov, Alexandre B. 2019a. Does data interpolation contradict statistical optimality? Pages 1611–1619 of: *Proc. International Conference on Artificial Intelligence and Statistics*.

Belkin, Mikhail, Hsu, Daniel, Ma, Siyuan, and Mandal, Soumik. 2019b. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, **116**(32), 15849–15854.

Belkin, Mikhail, Hsu, Daniel, and Xu, Ji. 2020. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, **2**(4), 1167–1180.

Bellman, Richard. 1952. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, **38**(8), 716.

Berner, Christopher, Brockman, Greg, Chan, Brooke, Cheung, Vicki, Debiak, Przemyslaw, Dennison, Christy, Farhi, David, Fischer, Quirin, Hashme, Shariq, and Hesse, Chris. 2019a. Dota 2 with large scale deep reinforcement learning. ArXiv preprint arXiv:1912.06680.

Berner, Julius, Elbrächter, Dennis, and Grohs, Philipp. 2019b. How degenerate is the parametrization of neural networks with the ReLU activation function? Pages 7790–7801 of: *Advances in Neural Information Processing Systems*.

Berner, Julius, Grohs, Philipp, and Jentzen, Arnulf. 2020a. 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**(3), 631–657.

Berner, Julius, Dablander, Markus, and Grohs, Philipp. 2020b. Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. Pages 16615–16627 of: *Advances in Neural Information Processing Systems*.

Blum, Avrim, and Rivest, Ronald L. 1989. Training a 3-node neural network is NP-complete. Pages 494–501 of: *Advances in Neural Information Processing Systems*.

Bohn, Jan, and Feischl, Michael. 2019. Recurrent neural networks as optimal mesh refinement strategies. ArXiv preprint arXiv:1909.04275.

Bourely, Alfred, Boueri, John Patrick, and Choromonski, Krzysztof. 2017. Sparse neural networks topologies. ArXiv preprint arXiv:1706.05683.

Bousquet, Olivier, and Elisseeff, André. 2002. Stability and generalization. *Journal of Machine Learning Research*, **2**(March), 499–526.

Bousquet, Olivier, Boucheron, Stéphane, and Lugosi, Gábor. 2003. Introduction to statistical learning theory. Pages 169–207 of: *Proc. Summer School on Machine Learning*.

Bronstein, Michael M, Bruna, Joan, LeCun, Yann, Szlam, Arthur, and Vandergheynst, Pierre. 2017. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, **34**(4), 18–42.

Brown, Tom, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared D, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, Agarwal, Sandhini, Herbert-Voss, Ariel, Krueger, Gretchen, Henighan, Tom, Child, Rewon, Ramesh, Aditya, Ziegler, Daniel, Wu, Jeffrey, Winter, Clemens, Hesse, Chris, Chen, Mark, Sigler, Eric, Litwin, Mateusz, Gray, Scott, Chess, Benjamin, Clark, Jack, Berner, Christopher, McCandlish, Sam, Radford, Alec, Sutskever, Ilya, and Amodei, Dario. 2020. Language models are few-shot learners. Pages 1877–1901 of: *Advances in Neural Information Processing Systems*.

Candès, Emmanuel J. 1998. *Ridgelets: Theory and Applications*. Ph.D. thesis, Stanford University.

Caragea, Andrei, Petersen, Philipp, and Voigtlaender, Felix. 2020. Neural network approximation and estimation of classifiers with classification boundary in a Barron class. ArXiv preprint arXiv:2011.09363.

Casazza, Peter G., Kutyniok, Gitta, and Philipp, Friedrich. 2012. Introduction to finite frame theory. Pages 1–53 of: *Finite Frames: Theory and Applications*. Birkhäuser Boston.

Chen, Lin, Min, Yifei, Belkin, Mikhail, and Karbasi, Amin. 2020. Multiple descent: Design your own generalization curve. ArXiv preprint arXiv:2008.01036.

Chen, Minshuo, Jiang, Haoming, Liao, Wenjing, and Zhao, Tuo. 2019. Efficient approximation of deep ReLU networks for functions on low dimensional manifolds. Pages 8174–8184 of: *Advances in Neural Information Processing Systems*.

Chizat, Lenaic, and Bach, Francis. 2020. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. Pages 1305–1338 of: *Proc. Conference on Learning Theory*.

Chizat, Lenaic, Oyallon, Edouard, and Bach, Francis. 2019. On lazy training in differentiable programming. Pages 2937–2947 of: *Advances in Neural Information Processing Systems*.

Cho, Kyunghyun, van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. Pages 1724–1734 of: *Proc. 2014 Conference on Empirical Methods in Natural Language Processing*.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Gérard Ben, and LeCun, Yann. 2015a. The loss surfaces of multilayer networks. Pages 192–204 of: *Proc. International Conference on Artificial Intelligence and Statistics*.

Choromanska, Anna, LeCun, Yann, and Arous, Gérard Ben. 2015b. Open problem: rhe landscape of the loss surfaces of multilayer networks. Pages 1756–1760 of: *Proc. Conference on Learning Theory*.

Chui, Charles K., and Mhaskar, Hrushikesh N. 2018. Deep nets for local manifold learning. *Frontiers in Applied Mathematics and Statistics*, **4**, 12.

Chui, Charles K., Li, Xin, and Mhaskar, Hrushikesh N. 1994. Neural networks for localized approximation. *Mathematics of Computation*, **63**(208), 607–623.

Cloninger, Alexander, and Klock, Timo. 2020. ReLU nets adapt to intrinsic dimensionality beyond the target domain. ArXiv preprint arXiv:2008.02545.

Cucker, Felipe, and Smale, Steve. 2002. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, **39**(1), 1–49.

Cucker, Felipe, and Zhou, Ding-Xuan. 2007. *Learning Theory: An Approximation Theory Viewpoint*. Cambridge University Press.

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

Czaja, Wojciech, and Li, Weilin. 2019. Analysis of time–frequency scattering transforms. *Applied and Computational Harmonic Analysis*, **47**(1), 149–171.

Dauphin, Yann N., Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, Ganguli, Surya, and Bengio, Yoshua. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. Pages 2933–2941 of: *Advances in Neural Information Processing Systems*.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. 2009. Imagenet: A large-scale hierarchical image database. Pages 248–255 of: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

DeVore, Ronald A. 1998. Nonlinear approximation. *Acta Numerica*, **7**, 51–150.

DeVore, Ronald, Hanin, Boris, and Petrova, Guergana. 2021. Neural network approximation. *Acta Numerica*, **30**, 327–444.

Devroye, Luc, Györfi, László, and Lugosi, Gábor. 1996. *A Probabilistic Theory of Pattern Recognition*. Springer.

Dirac, Paul Adrien Maurice. 1929. Quantum mechanics of many-electron systems. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, **123**(792), 714–733.

Dittmer, Sören, Kluth, Tobias, Maass, Peter, and Baguer, Daniel Otero. 2020. Regularization by architecture: A deep prior approach for inverse problems. *Journal of Mathematical Imaging and Vision*, **62**(3), 456–470.

Donoghue, William F. 1969. *Distributions and Fourier Transforms*. Academic Press.

Dreyfus, Stuart. 1962. The numerical solution of variational problems. *Journal of Mathematical Analysis and Applications*, **5**(1), 30–45.

Du, Simon S., Hu, Wei, and Lee, Jason D. 2018a. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. Pages 384–395 of: *Advances in Neural Information Processing Systems*.

Du, Simon S., Zhai, Xiyu, Poczos, Barnabas, and Singh, Aarti. 2018b. Gradient descent provably optimizes over-parameterized neural networks. In: *Proc. International Conference on Learning Representations*.

Du, Simon S., Lee, Jason D., Li, Haochuan, Wang, Liwei, and Zhai, Xiyu. 2019. Gradient descent finds global minima of deep neural networks. Pages 1675–1685 of: *Proc. International Conference on Machine Learning*.

Dudley, Richard M. 1967. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *Journal of Functional Analysis*, **1**(3), 290–330.

Dudley, Richard M. 2014. *Uniform Central Limit Theorems*. Cambridge University Press.

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

E, Weinan. 2017. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, **5**(1), 1–11.

E, Weinan, and Wojtowytsch, Stephan. 2020a. On the Banach spaces associated with multi-layer ReLU networks: Function representation, approximation theory and gradient descent dynamics. ArXiv preprint arXiv:2007.15623.

E, Weinan, and Wojtowytsch, Stephan. 2020b. A priori estimates for classification problems using neural networks. ArXiv preprint arXiv:2009.13500.

E, Weinan, and Wojtowytsch, Stephan. 2020c. Representation formulas and pointwise properties for Barron functions. ArXiv preprint arXiv:2006.05982.

E, Weinan, and Yu, Bing. 2018. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, **6**(1), 1–12.

E, Weinan, Ma, Chao, and Wu, Lei. 2019a. Barron spaces and the compositional function spaces for neural network models. ArXiv preprint arXiv:1906.08039.

E, Weinan, Han, Jiequn, and Li, Qianxiao. 2019b. A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences*, **6**(1), 1–41.

E, Weinan, Hutzenthaler, Martin, Jentzen, Arnulf, and Kruse, Thomas. 2019c. 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**(3), 1534–1571.

E, Weinan, Ma, Chao, and Wu, Lei. 2019d. A priori estimates of the population risk for two-layer neural networks. *Communications in Mathematical Sciences*, **17**(5), 1407–1425.

E, Weinan, Ma, Chao, Wojtowytsch, Stephan, and Wu, Lei. 2020. Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. ArXiv preprint arXiv:2009.10713.

Elbrächter, Dennis, Grohs, Philipp, Jentzen, Arnulf, and Schwab, Christoph. 2018. DNN expression rate analysis of high-dimensional PDEs: Application to option pricing. ArXiv preprint arXiv:1809.07669.

Elbrächter, Dennis, Perekrestenko, Dmytro, Grohs, Philipp, and Bölcskei, Helmut. 2019. Deep neural network approximation theory. ArXiv preprint arXiv:1901.02220.

Eldan, Ronen, and Shamir, Ohad. 2016. The power of depth for feedforward neural networks. Pages 907–940 of: *Proc. Conference on Learning Theory*, vol. 49.

Elman, Jeffrey L. 1990. Finding structure in time. *Cognitive Science*, **14**(2), 179–211.

Faber, Felix A., Hutchison, Luke, Huang, Bing, Gilmer, Justin, Schoenholz, Samuel S., Dahl, George E., Vinyals, Oriol, Kearnes, Steven, Riley, Patrick F., and Von Lilienfeld, O. Anatole. 2017. Prediction errors of molecular machine learning models lower than hybrid DFT error. *Journal of Chemical Theory and Computation*, **13**(11), 5255–5264.

Frankle, Jonathan, and Carbin, Michael. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: *proc. International Conference on Learning Representations*.

Freeman, Daniel C., and Bruna, Joan. 2017. Topology and geometry of half-rectified network optimization. In: *Proc. International Conference on Learning Representations*.

Funahashi, Ken-Ichi. 1989. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, **2**(3), 183–192.

Ge, Rong, Huang, Furong, Jin, Chi, and Yuan, Yang. 2015. Escaping from saddle points – online stochastic gradient for tensor decomposition. Pages 797–842 of: *Proc. Conference on Learning Theory*.

Geiger, Mario, Jacot, Arthur, Spigler, Stefano, Gabriel, Franck, Sagun, Levent, d'Ascoli, Stéphane, Biroli, Giulio, Hongler, Clément, and Wyart, Matthieu. 2020. Scaling description of generalization with number of parameters in

deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, **2**(2), 023401.

Géron, Aurelien. 2017. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Ghadimi, Saeed, and Lan, Guanghui. 2013. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, **23**(4), 2341–2368.

Ghorbani, Behrooz, Mei, Song, Misiakiewicz, Theodor, and Montanari, Andrea. 2021. Linearized two-layers neural networks in high dimension. *Annals of Statistics*, **49**(2), 1029–1054.

Gilton, Davis, Ongie, Greg, and Willett, Rebecca. 2019. Neumann networks for linear inverse problems in imaging. *IEEE Transactions on Computational Imaging*, **6**, 328–343.

Giné, Evarist, and Zinn, Joel. 1984. Some limit theorems for empirical processes. *Annals of Probability*, 929–989.

Golowich, Noah, Rakhlin, Alexander, and Shamir, Ohad. 2018. Size-independent sample complexity of neural networks. Pages 297–299 of: *Proc. Conference On Learning Theory*.

Gonon, Lukas, and Schwab, Christoph. 2020. Deep ReLU network expression rates for option prices in high-dimensional, exponential Lévy models. ETH Zurich SAM Research Report.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. 2014. Generative adversarial nets. Pages 2672–2680 of: *Advances in Neural Information Processing Systems*.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. 2016. *Deep Learning*. MIT Press.

Griewank, Andreas, and Walther, Andrea. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM.

Grohs, Philipp, and Herrmann, Lukas. 2021. Deep neural network approximation for high-dimensional parabolic Hamilton–Jacobi–Bellman equations. ArXiv preprint arXiv:2103.05744.

Grohs, Philipp, and Voigtlaender, Felix. 2021. Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces. ArXiv preprint arXiv:2104.02746.

Grohs, Philipp, Koppensteiner, Sarah, and Rathmair, Martin. 2020. Phase retrieval: Uniqueness and stability. *SIAM Review*, **62**(2), 301–350.

Grohs, Philipp, Hornung, Fabian, Jentzen, Arnulf, and von Wurstemberger, Philippe. 2021. 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*, to appear.

Gühring, Ingo, Kutyniok, Gitta, and Petersen, Philipp. 2020. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *Analysis and Applications*, **18**(05), 803–859.

Gunasekar, Suriya, Lee, Jason D., Soudry, Daniel, and Srebro, Nathan. 2018a. Characterizing implicit bias in terms of optimization geometry. Pages 1832–1841 of: *Proc. International Conference on Machine Learning*.

Gunasekar, Suriya, Lee, Jason D., Soudry, Daniel, and Srebro, Nathan. 2018b. Implicit bias of gradient descent on linear convolutional networks. Pages 9461–9471 of: *Advances in Neural Information Processing Systems*.

Haeffele, Benjamin D., and Vidal, René. 2017. Global optimality in neural network training. Pages 7331–7339 of: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

Hairer, Martin, Hutzenthaler, Martin, and Jentzen, Arnulf. 2015. Loss of regularity for Kolmogorov equations. *Annals of Probability*, **43**(2), 468–527.

Han, Song, Mao, Huizi, and Dally, William J. 2016. Deep compression: compressing deep neural network with pruning, trained quantization and Huffman coding. In: *Proc. International Conference on Learning Representations*.

Han, Jiequn, Jentzen, Arnulf, and E, Weinan. 2018. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, **115**(34), 8505–8510.

Han, Jiequn, Zhang, Linfeng, and E, Weinan. 2019. Solving many-electron Schrödinger equation using deep neural networks. *Journal of Computational Physics*, **399**, 108929.

Hanin, Boris. 2019. Universal function approximation by deep neural nets with bounded width and ReLU activations. *Mathematics*, **7**(10), 992.

Hanin, Boris, and Rolnick, David. 2019. Deep ReLU networks have surprisingly few activation patterns. Pages 359–368 of: *Advances in Neural Information Processing Systems*.

Hanin, Boris, and Sellke, Mark. 2017. Approximating continuous functions by ReLU nets of minimal width. ArXiv preprint arXiv:1710.11278.

Hardt, Moritz, Recht, Ben, and Singer, Yoram. 2016. Train faster, generalize better: Stability of stochastic gradient descent. Pages 1225–1234 of: *Proc. International Conference on Machine Learning*.

Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

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

Haussler, David. 1995. Sphere packing numbers for subsets of the Boolean $n$-cube with bounded Vapnik–Chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, **2**(69), 217–232.

He, Juncai, Li, Lin, Xu, Jinchao, and Zheng, Chunyue. 2020. ReLU deep neural

networks and linear finite elements. *Journal of Computational Mathematics*, **38**(3), 502–527.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. 2015. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. Pages 1026–1034 of: *Proc. IEEE International Conference on Computer Vision*.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. 2016. Deep residual learning for image recognition. Pages 770–778 of: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

Hermann, Jan, Schätzle, Zeno, and Noé, Frank. 2020. Deep-neural-network solution of the electronic Schrödinger equation. *Nature Chemistry*, **12**(10), 891–897.

Higham, Catherine F., and Higham, Desmond J. 2019. Deep learning: An introduction for applied mathematicians. *SIAM Review*, **61**(4), 860–891.

Hinton, Geoffrey E., and Zemel, Richard S. 1994. Autoencoders, minimum description length, and Helmholtz free energy. *Advances in Neural Information Processing Systems*, **6**, 3–10.

Hinz, Peter, and van de Geer, Sara. 2019. 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**, 7304–7324.

Hochreiter, Sepp, and Schmidhuber, Jürgen. 1997. Long short-term memory. *Neural Computation*, **9**(8), 1735–1780.

Hoeffding, Wassily. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, **58**(301), 13–30.

Hopfield, John J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, **79**(8), 2554–2558.

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

Huang, Gao, Sun, Yu, Liu, Zhuang, Sedra, Daniel, and Weinberger, Kilian Q. 2016. Deep networks with stochastic depth. Pages 646–661 of: *Proc. European Conference on Computer Vision*.

Hutzenthaler, Martin, Jentzen, Arnulf, Kruse, Thomas, and Nguyen, Tuan Anh. 2020. 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**(2), 1–34.

Ioffe, Sergey, and Szegedy, Christian. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Pages 448–456 of: *Proc. International Conference on Machine Learning*.

Jacot, Arthur, Gabriel, Franck, and Hongler, Clément. 2018. Neural tangent kernel: Convergence and generalization in neural networks. Pages 8571–8580 of: *Advances in Neural Information Processing Systems*.

Jentzen, Arnulf, Kuckuck, Benno, Neufeld, Ariel, and von Wurstemberger, Philippe.

2020. Strong error analysis for stochastic gradient descent optimization algorithms. *IMA Journal of Numerical Analysis*, **41**(1), 455–492.

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

Ji, Ziwei, and Telgarsky, Matus. 2019b. A refined primal–dual analysis of the implicit bias. ArXiv preprint arXiv:1906.04540.

Ji, Ziwei, and Telgarsky, Matus. 2020. Directional convergence and alignment in deep learning. Pages 17176–17186 of: *Advances in Neural Information Processing Systems*.

Jiang, Yiding, Krishnan, Dilip, Mobahi, Hossein, and Bengio, Samy. 2019. Predicting the generalization gap in deep networks with margin distributions. In: *Proc. International Conference on Learning Representations*.

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

Jin, Bangti, Maaß, Peter, and Scherzer, Otmar. 2017a. Sparsity regularization in inverse problems. *Inverse Problems*, **33**(6), 060301.

Jin, Kyong Hwan, McCann, Michael T., Froustey, Emmanuel, and Unser, Michael. 2017b. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, **26**(9), 4509–4522.

Jordan, Michael I. 1990. Attractor dynamics and parallelism in a connectionist sequential machine. Pages 112–127 of: *Artificial Neural Networks: Concept Learning*. IEEE Press.

Judd, Stephen J. 1990. *Neural Network Design and the Complexity of Learning*. MIT Press.

Kakade, Sham M., and Lee, Jason D. 2018. Provably correct automatic subdifferentiation for qualified programs. Pages 7125–7135 of: *Advances in Neural Information Processing Systems*.

Karpinski, Marek, and Macintyre, Angus. 1997. Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks. *Journal of Computer and System Sciences*, **54**(1), 169–176.

Kelley, Henry J. 1960. Gradient theory of optimal flight paths. *Ars Journal*, **30**(10), 947–954.

Keskar, Nitish Shirish, Mudigere, Dheevatsa, Nocedal, Jorge, Smelyanskiy, Mikhail, and Tang, Ping Tak Peter. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In: *Proc. International Conference on Learning Representations*.

Kidger, Patrick, and Lyons, Terry. 2020. Universal approximation with deep narrow networks. Pages 2306–2327 of: *Proc. Conference on Learning Theory*.

Kiefer, Jack, and Wolfowitz, Jacob. 1952. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, **23**(3), 462–466.

Krizhevsky, Alex, and Hinton, Geoffrey. 2009. Learning multiple layers of features from tiny images. Technical Report. University of Toronto.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. 2012. ImageNet classification with deep convolutional neural networks. Pages 1097–1105 of: *Advances in Neural Information Processing Systems*.

Kutyniok, Gitta, Petersen, Philipp, Raslan, Mones, and Schneider, Reinhold. 2019. A theoretical analysis of deep neural networks and parametric PDEs. ArXiv preprint arXiv:1904.00377.

Laakmann, Fabian, and Petersen, Philipp. 2021. Efficient approximation of solutions of parametric linear transport equations by ReLU DNNs. *Advances in Computational Mathematics*, **47**(1), 1–32.

Lample, Guillaume, and Charton, François. 2019. Deep learning For symbolic mathematics. In: *Proc. International Conference on Learning Representations*.

LeCun, Yann, Boser, Bernhard, Denker, John S., Henderson, Donnie, Howard, Richard E., Hubbard, Wayne, and Jackel, Lawrence D. 1989a. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**(4), 541–551.

LeCun, Yann, Denker, John S., and Solla, Sara A. 1989b. Optimal brain damage. Pages 598–605 of: *Advances in Neural Information Processing Systems*.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. 2015. Deep learning. *Nature*, **521**(7553), 436–444.

Ledoux, Michel, and Talagrand, Michel. 1991. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer Science & Business Media.

Lee, Jason D., Simchowitz, Max, Jordan, Michael I., and Recht, Benjamin. 2016. Gradient descent only converges to minimizers. Pages 1246–1257 of: *Proc. Conference on Learning Theory*.

Lee, Jaehoon, Bahri, Yasaman, Novak, Roman, Schoenholz, Samuel S., Pennington, Jeffrey, and Sohl-Dickstein, Jascha. 2018. Deep neural networks as Gaussian processes. In: *Proc. International Conference on Learning Representations*.

Lee, Jaehoon, Xiao, Lechao, Schoenholz, Samuel S., Bahri, Yasaman, Novak, Roman, Sohl-Dickstein, Jascha, and Pennington, Jeffrey. 2020. Wide neural networks of any depth evolve as linear models under gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, **2020**(12), 124002.

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

Lewin, Kurt. 1943. Psychology and the process of group living. *The Journal of Social Psychology*, **17**(1), 113–131.

Li, Bo, Tang, Shanshan, and Yu, Haijun. 2019a. Better approximations of high-dimensional smooth functions by deep neural networks with rectified power units. *Communications in Computational Physics*, **27**(2), 379–411.

Li, Housen, Schwab, Johannes, Antholzer, Stephan, and Haltmeier, Markus. 2020. NETT: Solving inverse problems with deep neural networks. *Inverse Problems*, **36**(6), 065005.

Li, Qianxiao, Lin, Ting, and Shen, Zuowei. 2019b. Deep learning via dynamical systems: An approximation perspective. ArXiv preprint arXiv:1912.10382.

Li, Weilin. 2021. Generalization error of minimum weighted norm and kernel interpolation. *SIAM Journal on Mathematics of Data Science*, **3**(1), 414–438.

Li, Yuanzhi, and Liang, Yingyu. 2018. Learning overparameterized neural networks via stochastic gradient descent on structured data. Pages 8157–8166 of: *Advances in Neural Information Processing Systems*.

Liang, Shiyu, and Srikant, R. 2017. Why deep neural networks for function approximation? In: *Proc. International Conference on Learning Representations*.

Liang, Tengyuan, and Rakhlin, Alexander. 2020. Just interpolate: Kernel "ridgeless" regression can generalize. *Annals of Statistics*, **48**(3), 1329–1347.

Liang, Tengyuan, Poggio, Tomaso, Rakhlin, Alexander, and Stokes, James. 2019. Fisher–Rao metric, geometry, and complexity of neural networks. Pages 888–896 of: *Proc. International Conference on Artificial Intelligence and Statistics*.

Liang, Tengyuan, Rakhlin, Alexander, and Zhai, Xiyu. 2020. On the multiple descent of minimum-norm interpolants and restricted lower isometry of kernels. Pages 2683–2711 of: *Proc. Conference on Learning Theory*.

Lin, Licong, and Dobriban, Edgar. 2021. What causes the test error? Going beyond bias-variance via anova. *Journal of Machine Learning Research*, **22**(155), 1–82.

Linnainmaa, Seppo. 1970. *Alogritmin Kumulatiivinen Pyöristysvirhe Yksittäisten Pyöristysvirheiden Taylor-Kehitelmänä*. M.Phil. thesis, University of Helsinki.

Lu, Yiping, Ma, Chao, Lu, Yulong, Lu, Jianfeng, and Ying, Lexing. 2020. A mean field analysis of deep ResNet and beyond: Towards provable optimization via overparameterization from depth. Pages 6426–6436 of: *Proc. International Conference on Machine Learning*.

Lunz, Sebastian, Öktem, Ozan, and Schönlieb, Carola-Bibiane. 2018. Adversarial regularizers in inverse problems. Pages 8507–8516 of: *Advances in Neural Information Processing Systems*.

Lyu, Kaifeng, and Li, Jian. 2019. Gradient descent maximizes the margin of homogeneous neural networks. In: *Proc. International Conference on Learning Representations*.

Ma, Junshui, Sheridan, Robert P., Liaw, Andy, Dahl, George E., and Svetnik, Vladimir. 2015. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, **55**(2), 263–274.

Maiorov, Vitaly, and Pinkus, Allan. 1999. Lower bounds for approximation by MLP neural networks. *Neurocomputing*, **25**(1-3), 81–91.

Mallat, Stéphane. 2012. Group invariant scattering. *Communications on Pure and Applied Mathematics*, **65**(10), 1331–1398.

Mallat, Stéphane. 2016. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **374**(2065), 20150203.

Marcati, Carlo, Opschoor, Joost, Petersen, Philipp, and Schwab, Christoph. 2020. Exponential ReLU neural network approximation rates for point and edge singularities. ETH Zurich SAM Research Report.

Matthews, Alexander G. de G., Hron, Jiri, Rowland, Mark, Turner, Richard E., and Ghahramani, Zoubin. 2018. Gaussian process behaviour in wide deep neural networks. In: *Proc. International Conference on Learning Representations*.

McAllester, David A. 1999. PAC-Bayesian model averaging. Pages 164–170 of: *Prc. Conference on Learning Theory*.

McCulloch, Warren S., and Pitts, Walter. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**(4), 115–133.

McDiarmid, Colin. 1989. On the method of bounded differences. Pages 148–188 of: *Surveys in Combinatorics*. London Mathematical Society Lecture Notes, vol. 141. Cambridge University Press.

Mei, Song, and Montanari, Andrea. 2019. The generalization error of random features regression: Precise asymptotics and double descent curve. ArXiv preprint arXiv:1908.05355.

Mendelson, Shahar. 2014. Learning without concentration. Pages 25–39 of: *Proc. Conference on Learning Theory*.

Mendelson, Shahar, and Vershynin, Roman. 2003. Entropy and the combinatorial dimension. *Inventiones Mathematicae*, **152**(1), 37–55.

Mhaskar, Hrushikesh N. 1996. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, **8**(1), 164–177.

Mianjy, Poorya, Arora, Raman, and Vidal, Rene. 2018. On the implicit bias of dropout. Pages 3540–3548 of: *Proc. International Conference on Machine Learning*.

Minsky, Marvin, and Papert, Seymour A. 1969. *Perceptrons*. MIT Press.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. 2013. Playing Atari with deep reinforcement learning. ArXiv preprint arXiv:1312.5602.

Monga, Vishal, Li, Yuelong, and Eldar, Yonina C. 2021. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, **38**(2), 18–44.

Montanari, Andrea, and Zhong, Yiqiao. 2020. The interpolation phase transition in neural networks: Memorization and generalization under lazy training. ArXiv preprint arXiv:2007.12826.

Montúfar, Guido, Pascanu, Razvan, Cho, Kyunghyun, and Bengio, Yoshua. 2014. On the number of linear regions of deep neural networks. Pages 2924–2932 of: *Advances in Neural Information Processing Systems*.

Muthukumar, Vidya, Vodrahalli, Kailas, Subramanian, Vignesh, and Sahai, Anant. 2020. Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, **1**(1), 67–83.

Nacson, Mor Shpigel, Lee, Jason D., Gunasekar, Suriya, Savarese, Pedro Henrique Pamplona, Srebro, Nathan, and Soudry, Daniel. 2019. Convergence of gradient descent on separable data. Pages 3420–3428 of: *International Conference on Artificial Intelligence and Statistics*.

Nagarajan, Vaishnavh, and Kolter, J. Zico. 2019. Uniform convergence may be unable to explain generalization in deep learning. Pages 11615–11626 of: *Advances in Neural Information Processing Systems*.

Nakada, Ryumei and Imaizumi, Masaaki. 2020. Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. *Journal of Machine Learning Research*, **21**(174), 1–38.

Nakamura-Zimmerer, Tenavi, Gong, Qi, and Kang, Wei. 2021. Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations. *SIAM Journal on Scientific Computing*, **43**(2), A1221–A1247.

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

Nemirovski, Arkadi, Juditsky, Anatoli, Lan, Guanghui, and Shapiro, Alexander. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, **19**(4), 1574–1609.

Nemirovsky, Arkadi Semenovich, and Yudin, David Borisovich. 1983. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics. Wiley.

Neyshabur, Behnam, Tomioka, Ryota, and Srebro, Nathan. 2014. In search of the real inductive bias: On the role of implicit regularization in deep learning. ArXiv preprint arXiv:1412.6614.

Neyshabur, Behnam, Tomioka, Ryota, and Srebro, Nathan. 2015. Norm-based capacity control in neural networks. Pages 1376–1401 of: *Proc. Conference on Learning Theory*.

Neyshabur, Behnam, Bhojanapalli, Srinadh, McAllester, David, and Srebro, Nati. 2017. Exploring generalization in deep learning. Pages 5947–5956 of: *Advances in Neural Information Processing Systems*.

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

Nguyen, Quynh, and Hein, Matthias. 2017. The loss surface of deep and wide neural networks. Pages 2603–2612 of: *Proc. International Conference on Machine Learning*.

Novak, Erich, and Woźniakowski, Henryk. 2009. Approximation of infinitely differentiable multivariate functions is intractable. *Journal of Complexity*, **25**(4), 398–404.

Olshausen, Bruno A., and Field, David J. 1996. Sparse coding of natural images produces localized, oriented, bandpass receptive fields. *Nature*, **381**(60), 609.

Oono, Kenta, and Suzuki, Taiji. 2019. Approximation and non-parametric estimation of ResNet-type convolutional neural networks. Pages 4922–4931 of: *Proc. International Conference on Machine Learning*.

Opschoor, Joost, Petersen, Philipp, and Schwab, Christoph. 2020. Deep ReLU networks and high-order finite element methods. *Analysis and Applications*, 1–56.

Orr, Genevieve B, and Müller, Klaus-Robert. 1998. *Neural Networks: Tricks of the Trade*. Springer.

Papyan, Vardan, Romano, Yaniv, and Elad, Michael. 2017a. Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, **18**(1), 2887–2938.

Papyan, Vardan, Sulam, Jeremias, and Elad, Michael. 2017b. Working locally thinking globally: Theoretical guarantees for convolutional sparse coding. *IEEE Transactions on Signal Processing*, **65**(21), 5687–5701.

Papyan, Vardan, Romano, Yaniv, Sulam, Jeremias, and Elad, Michael. 2018. Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks. *IEEE Signal Processing Magazine*, **35**(4), 72–89.

Pardoux, Etienne, and Peng, Shige. 1992. Backward stochastic differential equations and quasilinear parabolic partial differential equations. Pages 200–217 of: *Stochastic Partial Differential Equations and Their Applications*. Springer.

Petersen, Philipp, and Voigtlaender, Felix. 2018. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, **108**, 296–330.

Petersen, Philipp, and Voigtlaender, Felix. 2020. Equivalence of approximation by convolutional neural networks and fully-connected networks. *Proceedings of the American Mathematical Society*, **148**(4), 1567–1581.

Petersen, Philipp, Raslan, Mones, and Voigtlaender, Felix. 2020. Topological properties of the set of functions generated by neural networks of fixed size. *Foundations of Computational Mathematics*, **21**, 375–444.

Pfau, David, Spencer, James S., Matthews, Alexander G. D. G., and Foulkes, W. M. C. 2020. Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Physical Review Research*, **2**(3), 033429.

Pham, Hieu, Guan, Melody, Zoph, Barret, Le, Quoc, and Dean, Jeff. 2018. Efficient neural architecture search via parameters sharing. Pages 4095–4104 of: *Proc. International Conference on Machine Learning*.

Poggio, Tomaso, Rifkin, Ryan, Mukherjee, Sayan, and Niyogi, Partha. 2004. General conditions for predictivity in learning theory. *Nature*, **428**(6981), 419–422.

Poggio, Tomaso, Kawaguchi, Kenji, Liao, Qianli, Miranda, Brando, Rosasco, Lorenzo, Boix, Xavier, Hidary, Jack, and Mhaskar, Hrushikesh N. 2017a. Theory of deep learning III: explaining the non-overfitting puzzle. ArXiv preprint arXiv:1801.00173.

Poggio, Tomaso, Mhaskar, Hrushikesh N., Rosasco, Lorenzo, Miranda, Brando, and Liao, Qianli. 2017b. Why and when can deep – but not shallow – networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, **14**(5), 503–519.

Poole, Ben, Lahiri, Subhaneil, Raghu, Maithra, Sohl-Dickstein, Jascha, and Ganguli, Surya. 2016. Exponential expressivity in deep neural networks through transient chaos. Pages 3368–3376 of: *Advances in Neural Information Processing Systems*.

Raghu, Maithra, Poole, Ben, Kleinberg, Jon, Ganguli, Surya, and Sohl-Dickstein, Jascha. 2017. On the expressive power of deep neural networks. Pages 2847–2854 of: *Proc. International Conference on Machine Learning*.

Rahimi, Ali, Recht, Benjamin, et al. 2007. Random features for large-scale kernel machines. Pages 1177–1184 of: *Advances in Neural Information Processing Systems*.

Raissi, Maziar, Perdikaris, Paris, and Karniadakis, George E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, **378**, 686–707.

Ramanujan, Vivek, Wortsman, Mitchell, Kembhavi, Aniruddha, Farhadi, Ali, and Rastegari, Mohammad. 2020. What's hidden in a randomly weighted neural network? Pages 11893–11902 of: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

Ríos, P. López, Ma, Ao, Drummond, Neil D., Towler, Michael D., and Needs, Richard J. 2006. Inhomogeneous backflow transformations in quantum Monte Carlo calculations. *Physical Review E*, **74**(6), 066701.

Robbins, Herbert, and Monro, Sutton. 1951. A stochastic approximation method. *Annals of Mathematical Statistics*, 400–407.

Romano, Yaniv, Elad, Michael, and Milanfar, Peyman. 2017. The little engine that could: Regularization by denoising (RED). *SIAM Journal on Imaging Sciences*, **10**(4), 1804–1844.

Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. 2015. U-net: convolutional networks for biomedical image segmentation. Pages 234–241 of: *Proc. International Conference on Medical image Computing and Computer-Assisted Intervention*.

Rosenblatt, Frank. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**(6), 386.

Rudin, Walter. 2006. *Real and Complex Analysis*. McGraw-Hill.

Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. 1986. Learning representations by back-propagating errors. *Nature*, **323**(6088), 533–536.

Ruthotto, Lars, and Haber, Eldad. 2019. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 1–13.

Safran, Itay, and Shamir, Ohad. 2016. On the quality of the initial basin in overspecified neural networks. Pages 774–782 of: *Proc. International Conference on Machine Learning*.

Safran, Itay, and Shamir, Ohad. 2017. Depth–width tradeoffs in approximating natural functions with neural networks. Pages 2979–2987 of: *Proc. International Conference on Machine Learning*.

Safran, Itay, and Shamir, Ohad. 2018. Spurious local minima are common in two-layer ReLU neural networks. Pages 4433–4441 of: *Proc. International Conference on Machine Learning*.

Sakurai, Akito. 1999. Tight bounds for the VC-dimension of piecewise polynomial networks. Pages 323–329 of: *Advances in Neural Information Processing Systems*.

Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, and Madry, Aleksander. 2018. How does batch normalization help optimization? Pages 2488–2498 of: *Advances in Neural Information Processing Systems*.

Saxton, David, Grefenstette, Edward, Hill, Felix, and Kohli, Pushmeet. 2018. Analysing mathematical reasoning abilities of neural models. In: *Proc. International Conference on Learning Representations*.

Scherbela, Michael, Reisenhofer, Rafael, Gerard, Leon, Marquetand Philipp, and Grohs, Philipp. 2021. Solving the electronic Schrödinger equation for multiple nuclear geometries with weight-sharing deep neural network. ArXiv preprint arXiv:2105.08351.

Schmidhuber, Jürgen. 2015. Deep learning in neural networks: An overview. *Neural Networks*, **61**, 85–117.

Schmidt-Hieber, Johannes. 2019. Deep ReLU network approximation of functions on a manifold. ArXiv preprint arXiv:1908.00695.

Schütt, Kristof T., Kindermans, Pieter-Jan, Sauceda, Huziel E., Chmiela, Stefan, Tkatchenko, Alexandre, and Müller, Klaus-Robert. 2017. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. Pages 992–1002 of: *Advances in Neural Information Processing Systems*.

Schwab, Christoph, and Zech, Jakob. 2019. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Analysis and Applications*, **17**(01), 19–55.

Senior, Andrew W., Evans, Richard, Jumper, John, Kirkpatrick, James, Sifre, Laurent, Green, Tim, Qin, Chongli, Žídek, Augustin, Nelson, Alexander W. R., and Bridgland, Alex. 2020. Improved protein structure prediction using potentials from deep learning. *Nature*, **577**(7792), 706–710.

Shaham, Uri, Cloninger, Alexander, and Coifman, Ronald R. 2018. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, **44**(3), 537–557.

Shalev-Shwartz, Shai, and Ben-David, Shai. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Shalev-Shwartz, Shai, Shamir, Ohad, Srebro, Nathan, and Sridharan, Karthik. 2009. Stochastic convex optimization. In: *Proc. Conference on Learning Theory*.

Shapiro, Alexander, Dentcheva, Darinka, and Ruszczyński, Andrzej. 2014. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM.

Shen, Zuowei. 2020. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, **28**(5), 1768–1811.

Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, and Lanctot, Marc. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, **529**(7587), 484–489.

Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, and Bolton, Adrian. 2017. Mastering the game of Go without human knowledge. *Nature*, **550**(7676), 354–359.

Šíma, Jiří. 2002. Training a single sigmoidal neuron is hard. *Neural Computation*, **14**(11), 2709–2728.

Soudry, Daniel, Hoffer, Elad, Nacson, Mor Shpigel, Gunasekar, Suriya, and Srebro, Nathan. 2018. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, **19**, 1–57.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**(1), 1929–1958.

Srivastava, Rupesh Kumar, Greff, Klaus, and Schmidhuber, Jürgen. 2015. Training very deep networks. Pages 2377–2385 of: *Advances in Neural Information Processing Systems*.

Sulam, Jeremias, Papyan, Vardan, Romano, Yaniv, and Elad, Michael. 2018. Multilayer convolutional sparse modeling: Pursuit and dictionary learning. *IEEE Transactions on Signal Processing*, **66**(15), 4090–4104.

Szabo, Attila, and Ostlund, Neil S. 2012. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Courier Corporation.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. 2015. Going deeper with convolutions. Pages 1–9 of: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

Talagrand, Michel. 1994. Sharper bounds for Gaussian and empirical processes. *Annals of Probability*, 28–76.

Telgarsky, Matus. 2015. Representation benefits of deep feedforward networks. ArXiv preprint arXiv:1509.08101.

Thorpe, Matthew, and van Gennip, Yves. 2018. Deep limits of residual neural networks. ArXiv preprint arXiv:1810.11741.

Ulyanov, Dmitry, Vedaldi, Andrea, and Lempitsky, Victor. 2018. Deep image prior. Pages 9446–9454 of: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

van der Vaart, Aad W., and Wellner, Jon A. 1997. Weak convergence and empirical processes with applications to statistics. *Journal of the Royal Statistical Society Series A: Statistics in Society*, **160**(3), 596–608.

Vapnik, Vladimir. 1999. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, **10**(5), 988–999.

Vapnik, Vladimir. 2013. *The Nature of Statistical Learning Theory*. Springer Science & Business Media.

Vapnik, Vladimir, and Chervonenkis, Alexey. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & its Applications*, **16**(2), 264–280.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Łukasz, and Polosukhin, Illia. 2017. Attention is all you need. Pages 5998–6008 of: *Advances in Neural Information Processing Systems*.

Venturi, Luca, Bandeira, Alfonso S., and Bruna, Joan. 2019. Spurious valleys in one-hidden-layer neural network optimization landscapes. *Journal of Machine Learning Research*, **20**(133), 1–34.

Vershynin, Roman. 2018. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press.

Vinyals, Oriol, Babuschkin, Igor, Czarnecki, Wojciech M., Mathieu, Michaël, Dudzik, Andrew, Chung, Junyoung, Choi, David H., Powell, Richard, Ewalds, Timo, and Georgiev, Petko. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, **575**(7782), 350–354.

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Le Cun, Yann, and Fergus, Rob. 2013. Regularization of neural networks using dropconnect. Pages 1058–1066 of: *Proc. International Conference on Machine Learning*.

Werbos, Paul J. 1988. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, **1**(4), 339–356.

Whitney, Hassler. 1934. Analytic extensions of differentiable functions defined in closed sets. *Transactions of the American Mathematical Society*, **36**(1), 63–89.

Wiatowski, Thomas, Grohs, Philipp, and Bölcskei, Helmut. 2017. Energy propagation in deep convolutional neural networks. *IEEE Transactions on Information Theory*, **64**(7), 4819–4842.

Williams, Ronald J., and Zipser, David. 1995. Gradient-based learning algorithms for recurrent networks and their computational complexity. Pages 433–486 of: *Backpropagation: Theory, Architectures, and Applications*, Psychology Press.

Wu, Zonghan, Pan, Shirui, Chen, Fengwen, Long, Guodong, Zhang, Chengqi, and Philip, S. Yu. 2021. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, **32**(1), 4–24.

Xu, Huan, and Mannor, Shie. 2012. Robustness and generalization. *Machine learning*, **86**(3), 391–423.

Yang, Greg. 2019. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. ArXiv preprint arXiv:1902.04760.

Yarotsky, Dmitry. 2017. Error bounds for approximations with deep ReLU networks. *Neural Networks*, **94**, 103–114.

Yarotsky, Dmitry. 2018a. Optimal approximation of continuous functions by very deep ReLU networks. Pages 639–649 of: *Proc. Conference on Learning Theory*.

Yarotsky, Dmitry. 2018b. Universal approximations of invariant maps by neural networks. ArXiv preprint arXiv:1804.10306.

Yarotsky, Dmitry. 2021. Elementary superexpressive activations. ArXiv preprint arXiv:2102.10911.

Yarotsky, Dmitry, and Zhevnerchuk, Anton. 2020. The phase diagram of approximation rates for deep neural networks. In: *Advances in Neural Information Processing Systems*, vol. 33.

Ye, Jong Chul, Han, Yoseob, and Cha, Eunju. 2018. Deep convolutional framelets: A general deep learning framework for inverse problems. *SIAM Journal on Imaging Sciences*, **11**(2), 991–1048.

Yin, Rujie, Gao, Tingran, Lu, Yue M., and Daubechies, Ingrid. 2017. A tale of two bases: Local–nonlocal regularization on image patches with convolution framelets. *SIAM Journal on Imaging Sciences*, **10**(2), 711–750.

Young, Tom, Hazarika, Devamanyu, Poria, Soujanya, and Cambria, Erik. 2018. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, **13**(3), 55–75.

Yserentant, Harry. 2010. *Regularity and Approximability of Electronic Wave Functions*. Springer.

Zaslavsky, Thomas. 1975. *Facing up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes*. Memoirs of the American Mathematical Society. American Mathematical Society.

Zbontar, Jure, Knoll, Florian, Sriram, Anuroop, Murrell, Tullie, Huang, Zhengnan, Muckley, Matthew J., Defazio, Aaron, Stern, Ruben, Johnson, Patricia, Bruno, Mary, Parente, Marc, Geras, Krzysztof J., Katsnelson, Joe, Chandarana, Hersh, Zhang, Zizhao, Drozdzal, Michal, Romero, Adriana, Rabbat, Michael, Vincent, Pascal, Yakubova, Nafissa, Pinkerton, James, Wang, Duo, Owens, Erich, Zitnick, C. Lawrence, Recht, Michael P., Sodickson, Daniel K., and Lui, Yvonne W. 2018. fastMRI: An open dataset and benchmarks for accelerated MRI. ArXiv preprint arXiv:1811.08839.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. 2017. Understanding deep learning requires rethinking generalization. In: *Proc. International Conference on Learning Representations*.

Zhang, Chiyuan, Bengio, Samy, and Singer, Yoram. 2019. Are all layers created equal? ArXiv preprint arXiv:1902.01996.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Mozer, Michael C., and Singer, Yoram. 2020. Identity crisis: Memorization and generalization under extreme overparameterization. In: *Proc. International Conference on Learning Representations*.

Zhou, Ding-Xuan. 2020a. Theory of deep convolutional neural networks: Downsampling. *Neural Networks*, **124**, 319–327.

Zhou, Ding-Xuan. 2020b. Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, **48**(2), 787–794.

Zhou, Hao, Alvarez, Jose M., and Porikli, Fatih. 2016. Less is more: Towards compact CNNs. Pages 662–677 of: *Proc. European Conference on Computer Vision*.

Zoph, Barret, and Le, Quoc V. 2017. Neural architecture search with reinforcement learning. In: *Proc.Dobriban International Conference on Learning Representations*.

Zou, Difan, Cao, Yuan, Zhou, Dongruo, and Gu, Quanquan. 2020. Gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*, **109**(3), 467–492.