universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Scaling Behavior of Physics Informed Neural Networks for Solving Partial Differential Equations“

verfasst von / submitted by

Philipp Maximilian Möhl, B.Sc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 821
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Mathematik
Betreut von / Supervisor:	Univ.-Prof. Dr. Philipp Grohs

Abstract

This thesis provides a thorough introduction to the mathematical theory of learning through finite samples of training data. It then presents neural networks, along with gradient-based optimization methods, as a mathematical framework for solving learning problems. The theory of mathematical learning through neural networks is further extended with the addition of regularization techniques to introduce physics-informed neural networks, an algorithm for the numerical solution of general partial differential equations. The algorithm's validity as an approximator is confirmed through an abstract proof, offering insight into the theoretical underpinnings of the approach.

Numerical studies are empirically conducted to assess the scaling behavior of the algorithm with respect to the amounts of training data samples on three different classes of model problems with varying levels of complexity. These studies provide insight into the effectiveness of the algorithm in approximating solutions to partial differential equations and the role that the amount of training data plays in the approximation process.

Zusammenfassung

Diese Arbeit bietet eine gründliche Einführung in die mathematische Theorie des Lernens durch endliche Proben von Trainingsdaten. Des Weiteren präsentiert sie neuronale Netze zusammen mit Gradientenbasierten Optimierungsmethoden als mathematischen Rahmen zur Lösung von Lernproblemen. Die Theorie des mathematischen Lernens durch neuronale Netze wird durch die Hinzufügung von Regularisierungstechniken erweitert, um physikinformierte neuronale Netze einzuführen, einen Algorithmus für die numerische Lösung allgemeiner partieller Differentialgleichungen. Die Gültigkeit des Algorithmus als Approximator wird durch einen abstrakten Beweis bestätigt und gibt Einblick in die theoretischen Grundlagen des Ansatzes.

Es werden numerische Studien empirisch durchgeführt, um das Skalierungsverhalten des Algorithmus im Bezug auf die Mengen von Trainingsdatenproben auf drei verschiedene Klassen von Modellproblemen mit unterschiedlicher Komplexität zu bewerten. Diese Studien geben einen Einblick in die Effektivität des Algorithmus bei der Annäherung von Lösungen partieller Differentialgleichungen und die Rolle, welche die Menge der Trainingsdaten im Approximationsprozess spielt.

Contents

1 Introduction

A partial differential equation (PDE) is a mathematical model that describes the behavior of a physical system in terms of partial derivatives with respect to the space $x \in U$ and time $t \in (0, T)$ variables. These equations can be expressed in a general form as

$$L(u) = 0, \tag{1.1}$$

where u is the unknown function to be solved for and L is a differential operator that describes the underlying physical laws.
To fully define the problem, auxiliary conditions such as initial and boundary conditions are usually provided. These conditions are of the form

$$\begin{cases} u(\cdot, 0) = \varphi & \text{on } U \\ \mathcal{G}(u) = \psi & \text{on } \partial U \times [0, T], \end{cases} \tag{1.2}$$

where $\mathcal{G}$ is a boundary operator.
PDEs have a wide range of applications in various fields, including engineering, physics, and finance. However, finding solutions can be challenging, especially for high-dimensional and complex systems. In recent years, machine learning techniques such as neural networks have been utilized as a new approach to solve PDEs (cf. for instance Lagaris [15], Beck [63], Mishra [56] and Lye [61]), reformulating them as a learning problem. This involves the representation of the problem stated by a PDE as a minimization task of an error function

$$\mathcal{E}_Z(u) = \mathbb{E}[\mathcal{L}(u(X), Y)], \tag{1.3}$$

where $\mathcal{L}$ is a loss function, and X and Y are random vectors with values in the input and output domains of u, referred to as data $Z = (X, Y)$. By taking m realizations $\mathbf{z} = ((x_i, y_i))_{i=1}^m$ of samples drawn according to the distribution of the data, the empirical learning problem can be stated as the search for a minimizer of the empirical error function

$$\mathcal{E}_{\mathbf{z}}(u) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(u(x_i), y_i), \tag{1.4}$$

which approximates the error function. This approach opens up the possibility of applying numerical approximation methods over a hypothesis space of solvers.
For our purposes, we focus on neural networks as hypothesis spaces, who restate the empirical learning problem as the search for minimizing network parameters θ. In particular, we study physics-informed neural networks (PINNs) as introduced by Raissi in [53] and [57]. These networks have been shown to be a valid choice of approximators

for PDEs under certain assumptions (cf. for instance Mishra [71], Ryck [73] and Shin [62]). The algorithm utilizes equations (1.1) and (1.2) to extend the empirical learning problem with regularizing error terms towards the objective function

$$\Upsilon_{reg}(\theta) = \frac{1}{m_t}\sum_{i=1}^{m_t}|\delta_i^t|^2 + \frac{1}{m_d}\sum_{i=1}^{m_d}|\delta_i^d|^2 + \frac{1}{m_s}\sum_{i=1}^{m_s}|\delta_i^s|^2, \tag{1.5}$$

with the error terms

$$\begin{aligned}\delta_i^t &= u_\theta(x_i^t, 0) - \varphi(x_i^t)\\ \delta_i^d &= L(u_\theta)(x_i^d)\\ \delta_i^s &= \mathcal{G}(u_\theta)(x_i^s) - \psi(x_i^s),\end{aligned} \tag{1.6}$$

where $(x_i^t)_{i=1}^{m_t}$, $(x_i^d)_{i=1}^{m_d}$ and $(x_i^s)_{i=1}^{m_s}$ are realizations drawn from independent samples of random variables

$$\begin{aligned}X^t &: \Omega \to U\\ X^d &: \Omega \to U \times (0,T)\\ X^s &: \Omega \to \partial U \times [0,T].\end{aligned} \tag{1.7}$$

The inherent challenges in solving complex PDEs and limitations of traditional numerical methods prompt the need for a thorough investigation into the performance and scalability of PINNs. Our aim is to gain insight into the effectiveness of the algorithm in approximating solutions to PDEs when presented with varying levels of complexity and different amounts of training data samples.
In chapter 2, a thorough introduction to the theory of mathematical learning is provided. Neural networks and their properties are briefly introduced in chapter 3. Chapter 4 gives a general definition of PDEs and introduces the algorithm of PINNs as a learning strategy. In the last chapter numerical experiments are presented, which empirically study scaling behaviors of the algorithm.
The prerequisites for this thesis include a good understanding of measure-theoretic probability theory and functional analysis. Familiarity with PDEs is beneficial but not mandatory. Some of the references for these fields include Klenke [43], Ash [22], Billingsley [37], Aliprantis [32], Cannarsa [46], Rudin [10], Brezis [35] and Evans [34].

Notation. *Let $d, n, k \in \mathbb{N}$, let $\mathcal{D} \subseteq \mathbb{R}^d$, let $u : \mathcal{D} \to \mathbb{R}$, let $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}_0^d$ be a multi-index of order $|\alpha|$, let (S, ρ) be a metric space and let $p \in [1, \infty]$. We denote*

- *by $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ the non-negative integers,*
- *by $\langle \cdot, \cdot \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $\langle x, y \rangle = \sum_{i=1}^d x_i y_i$ the inner product on $\mathbb{R}^d$,*
- *by $\| \cdot \| : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, $\|x\| = \sqrt{\langle x, x \rangle}$ the Euclidean norm on $\mathbb{R}^d$,*
- *by $\frac{\partial u}{\partial x_i} = u_{x_i}$, $\frac{\partial^2 u}{\partial x_i \partial x_j} = u_{x_i x_j}$, $\frac{\partial^3 u}{\partial x_i \partial x_j \partial x_k} = u_{x_i x_j x_k}$, etc. the partial derivatives of u,*

- *by* $D^\alpha u(x) = \frac{\partial^{|\alpha|} u(x)}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}$ *the multi-index derivative of* u,
- *by* $D^k u(x) = \{D^\alpha u(x) \big| |\alpha| = k\} \in \mathbb{R}^{d^k}$ *the set of all partial derivatives of order* k *of* u,
- *by* $\mathcal{B}(S)$ *the Borel* σ*-algebra of* S,
- *by* $\mathcal{B}(\mathcal{D}, \mathbb{R}^n)$ *the space of* $\mathcal{B}(\mathcal{D})/\mathcal{B}(\mathbb{R}^n)$*-measurable functions,*
- *and by* $L^p(\mathcal{D}, \mathbb{R}^n)$ *the Lebesgue spaces of* $\mathcal{B}(\mathcal{D})/\mathcal{B}(\mathbb{R}^n)$*-measurable functions, where we follow the convention of writing* $L^p(\mathcal{D})$*, if* $n = 1$.

Further, let $p \in [1, \infty)$*, let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space and let* $X : \Omega \to \mathcal{D}$ *be a random vector, then we denote*

- *by* $\mathbb{P}_X = \mathbb{P} \circ X^{-1}$ *the push forward measure of* X *induced on the measurable space* $(\mathcal{D}, \mathcal{B}(\mathcal{D}))$,
- *by* $L^p(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$ *the space of* $\mathcal{B}(\mathcal{D})/\mathcal{B}(\mathbb{R}^n)$*-measurable functions* $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$*, which are integrable in the* p*-th power w.r.t.* $\mathbb{P}_X$

$$\int_{\mathcal{D}} \|f\|^p d\mathbb{P}_X < \infty, \tag{1.8}$$

- *and by* $L^p(\Omega, \mathbb{R}^n; \mathbb{P})$ *the space of random vectors* $Y : \Omega \to \mathbb{R}^n$*, which are integrable in the* p*-th power w.r.t.* $\mathbb{P}$

$$\int_{\Omega} \|Y\|^p d\mathbb{P} < \infty. \tag{1.9}$$

2 Foundations of Learning Theory

The following chapter presents a theoretical framework for the study of learning and discusses some of its main properties. We might explain learning as a change in behavior influenced by certain factors such as data and experience. Various scientific fields, including cognitive psychology, neuroscience, computer science, engineering, and mathematics, are involved in studying the underlying processes and laws to gain more insight into this phenomenon. Our context will be specifically machine learning, a sub-field of mathematics and computer science, which is devoted to develop artificial learning methods and algorithms. Although there is no universally accepted definition, we consider learning to be the acquisition of knowledge and new skills in our context. In machine learning, we are typically facing one of three primary paradigms. Either we learn from a given source of knowledge, what we wish to replicate, or we are confronted with unlabeled data, where underlying structures or relationships must be discovered within, or we learn from direct interactions with a given environment, while following a set of predefined laws. We refer to these three situations as supervised learning, unsupervised learning and reinforcement learning, which all differ in strategies of solving the underlying learning task. Ongoing, we will work exclusively with the first situation, supervised learning, as our matter is presented in the described format. However, one might not ignore the others, as they lead to interesting and promising results themselves (cf. for instance Goodfellow [41] and Fawzi [70]).
Typically, the source of knowledge in supervised learning tasks is presented in the form of given data $Z = (X, Y)$, where Z is a real-valued random vector on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We refer to X as the input, predictor or feature and Y as the output, response or label. We are aiming for a way to understand the relationship between both, where we have to keep in mind that their joint probability distribution $\mathbb{P}(X, Y)$ might be not fully understood. In general, we are trying to generate a function f via some algorithm such that

$$f(X) \approx Y. \tag{2.1}$$

The form of Y determines whether we are talking of a classification task or regression task. A discrete random vector leads naturally to a need to classify inputs into the resulting discrete values or classes. On the other hand, a continuous real-valued random vector seeks an understanding of how Y continuously changes w.r.t changes in X, which is called regression analysis. We will focus on the later case, where we measure performance of our approach with the quadrature loss.
In this chapter, we attempt to develop a mathematical framework for supervised learning. While there is a variety of different approaches, we will follow the definitions and formulations from Cucker [24], Poggio [27], Hastie [39] and Vapnik [18].

2.1 The Mathematical Learning Problem

We will now formulate the general framework for supervised learning tasks and derive a solution to it.

Definition 2.1.1 (The Mathematical Learning Problem)**.** *Let $d, n \in \mathbb{N}$, $\mathcal{D} \subseteq \mathbb{R}^d$ and let $X : \Omega \to \mathcal{D}$ and $Y : \Omega \to \mathbb{R}^n$ be random vectors on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Define the error function w.r.t. the data $Z = (X, Y)$ by*

$$\mathcal{E}_Z : \mathcal{B}(\mathcal{D}, \mathbb{R}^n) \to [0, \infty], \quad f \mapsto \int_\Omega \|f(X) - Y\|^2 d\mathbb{P} = \mathbb{E}[\|f(X) - Y\|^2]. \tag{2.2}$$

The mathematical learning problem defines the search for functions $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$, such that f minimizes the error function $\mathcal{E}_Z$.

More specifically, this form of $\mathcal{E}_Z$ is called the least squares error and its integrand, $\|(f(X) - Y)\|^2$, is referred to as the quadrature loss for f on the data Z. While there are other loss variations, we will strictly consider the former, as it possesses desirable properties and is most commonly used in regression tasks. Note that in general the error minimizing function is not unique.
Moving forward, whenever using the term data $Z = (X, Y)$, we are referring to the setting in the definition above, if not mentioned otherwise.

Definition 2.1.2 (Regression Function)**.** *Consider the mathematical learning problem for some data $Z = (X, Y)$, then we define the regression function w.r.t. Z by*

$$\hat{f}_Z : \mathcal{D} \to \mathbb{R}^n, \quad x \mapsto \mathbb{E}[Y|X = x]. \tag{2.3}$$

Intuitively speaking, $\hat{f}_Z(x)$ is the average over the y coordinate in $\{x\} \times Y$, which seems to be a natural choice for minimizing the least squares error $\mathcal{E}_Z$. We will see in the next Theorem, that it is in fact the best option we have.

Remark 2.1.3. *Note that if $Y \in L^1(\Omega, \mathbb{R}^n; \mathbb{P})$, then the regression function exists and is the $\mathbb{P}_X$-unique function, which fulfills*

$$\int_\Omega \mathbb{1}_{\{X \in D\}} \mathbb{E}[Y|X] d\mathbb{P} = \int_\Omega \mathbb{1}_{\{X \in D\}} Y d\mathbb{P} = \int_D \hat{f}_Z(x) d\mathbb{P}_X(x) \tag{2.4}$$

for every $D \in \mathcal{B}(\mathcal{D})$ (cf. for instance Ash [22, Theorem 5.3.3 and Section 5.4]). Therefore it holds that $\hat{f}_Z(X) = \mathbb{E}[Y|X]$.
Furthermore, in order to gain a deeper understanding of the regression function as a predictor, we aim to examine its relationship with the output Y more closely. We assume now that Y has finite variance and by applying the tower property for conditional expectation to the random variable $\hat{f}_Z(X) - Y$, we can calculate expectation and variance of the difference,

$$\begin{aligned} \mathbb{E}[\hat{f}_Z(X) - Y] &= \mathbb{E}[\mathbb{E}[Y|X] - Y] = 0 \\ Var[\hat{f}_Z(X) - Y] &= \mathbb{E}[\|\hat{f}_Z(X) - Y\|^2] = \mathcal{E}_Z(\hat{f}_Z). \end{aligned} \tag{2.5}$$

Let us now prove that the regression function is the best choice of function $f(X) \approx Y$, we are seeking to find.

Theorem 2.1.4 (Regression Function solves the Learning Problem)**.** *Let $Z = (X, Y)$ be some data and assume Y has finite variance. Let $\hat{f}_Z$ be the regression function w.r.t. Z, then for every function $f \in L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$, the error $\mathcal{E}_Z(f)$ is finite and decomposes into*

$$\mathcal{E}_Z(f) = \int_{\mathcal{D}} \|f(x) - \hat{f}_Z(x)\|^2 d\mathbb{P}_X(x) + \mathcal{E}_Z(\hat{f}_Z) < \infty. \tag{2.6}$$

Further, for every $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n) \backslash L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$ the error is infinite,

$$\mathcal{E}_Z(f) = \infty. \tag{2.7}$$

Proof. Let $f \in L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$ and let Y be of finite variance as the output of some data $Z = (X, Y)$. These assumptions lead to both Y and the concatenation of f and X being in $L^2(\Omega, \mathbb{R}^n; \mathbb{P})$, i.e., $Y \in L^2(\Omega, \mathbb{R}^n; \mathbb{P})$ and $f(X) \in L^2(\Omega, \mathbb{R}^n; \mathbb{P})$. We recall that the form of the error of f in (2.2) is given as $\mathcal{E}_Z(f) = \mathbb{E}[\|f(X) - Y\|^2]$, which already ensures the finiteness in this case.

We will proceed with the proof of the decomposition, where we will show that

$$\mathcal{E}_Z(f) = \mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2] + \mathbb{E}[\|Y - \hat{f}_Z(X)\|^2], \tag{2.8}$$

where $\hat{f}_Z$ is the regression function . We note that, as derived in a comment before, $\hat{f}_Z(X) = \mathbb{E}[Y|X]$ is satisfied and $\hat{f}_Z$ is the $\mathbb{P}_X$-unique function with this property. Hence, we can calculate,

$$\begin{aligned}
\mathcal{E}_Z(f) &= \mathbb{E}[\|f(X) - Y\|^2] \\
&= \mathbb{E}[\|f(X) - \mathbb{E}[Y|X] + \mathbb{E}[Y|X] - Y\|^2] \\
&= \mathbb{E}[\|f(X) - \mathbb{E}[Y|X]\|^2 + \|\mathbb{E}[Y|X] - Y\|^2 + 2\langle f(X) - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - Y\rangle] \\
&= \mathbb{E}[\|f(X) - \mathbb{E}[Y|X]\|^2] + \mathbb{E}[\|\mathbb{E}[Y|X] - Y\|^2] + 2\mathbb{E}[\langle f(X) - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - Y\rangle] \\
&= \mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2] + \mathbb{E}[\|\hat{f}_Z(X) - Y\|^2] + 2\mathbb{E}[\langle f(X) - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - Y\rangle].
\end{aligned} \tag{2.9}$$

To conclude the step, we need to show next, that the last term of the decomposition $2\mathbb{E}[\langle f(X) - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - Y\rangle]$ vanishes. We further note that $f(X)$ and $\mathbb{E}[Y|X]$ are both $\sigma(X)/\mathcal{B}(\mathbb{R}^n)$-measurable, which enables us to use the tower property for conditional expectations,

$$\begin{aligned}
&\mathbb{E}[\langle f(X) - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - Y\rangle] \\
&= \mathbb{E}[\mathbb{E}[\langle f(X) - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - Y\rangle | \sigma(X)]] \\
&= \mathbb{E}[\langle f(X) - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - \mathbb{E}[Y|X]\rangle] = 0.
\end{aligned} \tag{2.10}$$

This finishes the proof of (2.8) and therefore, it holds that

$$\begin{aligned}\mathcal{E}_Z(f) &= \mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2] + \mathbb{E}[\|Y - \hat{f}_Z(X)\|^2] \\ &= \int_{\mathcal{D}} \|f(x) - \hat{f}_Z(x)\|^2 d\mathbb{P}_X(x) + \mathcal{E}_Z(\hat{f}_Z).\end{aligned} \tag{2.11}$$

Moving forward, we assume that $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n) \backslash L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$, which implies that $f(X) \notin L^2(\Omega, \mathbb{R}^n; \mathbb{P})$. We again calculate the error of f

$$\begin{aligned}\mathcal{E}_Z(f) &= \mathbb{E}[\|f(X) - Y\|^2] \\ &= \mathbb{E}[\|f(X) - Y\|^2 + \|Y\|^2] - \mathbb{E}[\|Y\|^2] \\ &\geq \frac{1}{2}\mathbb{E}[\|f(X) - Y\|^2 + \|Y\|^2] + \mathbb{E}[\|f(X) - Y\|\|Y\|] - \mathbb{E}[\|Y\|^2] \\ &= \frac{1}{2}\mathbb{E}[(\|f(X) - Y\| + \|Y\|)^2] - \mathbb{E}[\|Y\|^2] \\ &\geq \frac{1}{2}\mathbb{E}[(\|f(X)\|^2] - \mathbb{E}[\|Y\|^2] = \infty,\end{aligned} \tag{2.12}$$

where we utilized in the first inequality the property that $x^2 - 2xy + y^2 = (x - y)^2 \geq 0$ holds for all $x, y \in \mathbb{R}$, in the last line the triangle inequality and the final result is a consequence of the assumptions made on Y and $f(X)$, thus completing proof.

□

Remark 2.1.5. *Theorem 2.1.4 lets us decompose the error $\mathcal{E}_Z(f)$ into two components. On the right-hand side, the error $\mathcal{E}_Z(\hat{f}_Z)$ is independent of the function f. As a result, $\hat{f}_Z$ minimizes $\mathcal{E}_Z$ and satisfies the mathematical learning problem under the given assumptions. Furthermore, $\mathcal{E}_Z(\hat{f}_Z)$ constitutes a lower bound on the error for functions $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$.*

From now on, when considering some data $Z = (X, Y)$, we will assume Y to have finite variance, or more precisely, $Y \in L^2(\Omega, \mathbb{R}^n; \mathbb{P})$.

2.2 Learning from Samples

Now that we found a theoretical solution to the mathematical learning problem, we need to raise the question of whether it is applicable in practise. In most learning situations, we do not possess the knowledge of the distribution of both random vectors X and Y of our data $Z = (X, Y)$, and explicit calculation of the conditional expectation may not be feasible. Therefore, we cannot always determine the regression function $\hat{f}_Z$. However, we are usually equipped with random samples from the data, which we want to use to approximate the regression function.
To formulate this, we will extend our framework developed earlier with a new error function for samples.

Definition 2.2.1 (The Mathematical Learning Problem, Sample Version)**.** *Let $m \in \mathbb{N}$, let $Z = (X, Y)$ be some data, and let*

$$\mathbf{z} = ((x_i, y_i))_{i=1}^m \in (\mathcal{D} \times \mathbb{R}^n)^m \tag{2.13}$$

be realizations of m samples independently drawn according to the distribution of Z. More precisely, let $((X_i, Y_i))_{i=1}^m$ be m independent and identically distributed (i.i.d.) copies of (X, Y), denoted by

$$\mathcal{Z} = ((X_i, Y_i))_{i=1}^m : \Omega \to (\mathcal{D} \times \mathbb{R}^n)^m, \tag{2.14}$$

then there exists an $\omega \in \Omega$, such that

$$\mathcal{Z}(\omega) = \mathbf{z}. \tag{2.15}$$

We define the sample version of the mathematical learning problem for empirical realizations $\mathbf{z}$ as the search for functions $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$ that result in small errors $\mathcal{E}_Z(f)$, utilizing only the partial knowledge of $\mathbf{z}$.

For the sake of notation we will implicitly include the number of samples m when we write $\mathcal{Z}$.

Remark 2.2.2. *We showed in Theorem 2.1.4, that the regression function minimizes the error function $\mathcal{E}_Z$, thus the sample version of the learning problem can be rewritten. In essence we are seeking for a function $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$ to approximate the regression function, given samples of data Z. Or more specifically, while utilizing only given partial knowledge $\mathbf{z}$, we are searching for f, such that*

$$\mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2] \tag{2.16}$$

is small with high probability.

Remark 2.2.3. *In Definition 2.2.1, note that we are always able to find i.i.d. copies drawn from the distribution of $Z = (X, Y)$, by considering the m-fold product space $(\Omega^m, \mathcal{F}^{\otimes m}, \mathbb{P}^{\otimes m})$, constructed by the probability space $(\Omega, \mathcal{F}, \mathbb{P})$.*
In fact, we can construct such copies by setting

$$((X_i, Y_i))_{i=1}^m = ((X \circ \pi_i, Y \circ \pi_i))_{i=1}^m, \tag{2.17}$$

where the functions $\pi_i : \Omega^m \to \Omega$ project every $\omega = (\omega_j)_{j=1}^m \in \Omega^m$ onto its i-th coordinate $\pi_i(\omega) = \omega_i$. Identifying the given data as $(X, Y) = (X \circ \pi_1, Y \circ \pi_1)$ on the product space $(\Omega^m, \mathcal{F}^{\otimes m}, \mathbb{P}^{\otimes m})$ leads to the requested construction.

To extend the structure of the new learning problem, we introduce a new error function, the empirical error function. This is particularly needed because, in general, we do not understand the distribution of the data Z and only have access to realizations of random samples $\mathcal{Z}$. The empirical error leverages these realizations to approximate the error $\mathcal{E}_Z$, which can no longer be directly measured.

Definition 2.2.4 (Empirical Error)**.** *Let $m \in \mathbb{N}$ and let $\mathcal{Z}$ be m i.i.d. copies of some data $Z = (X, Y)$. For every function $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$, we define the empirical error of f w.r.t. the samples $\mathcal{Z}$ by*

$$\begin{aligned} \mathcal{E}_{\mathcal{Z}} : \Omega \times \mathcal{B}(\mathcal{D}, \mathbb{R}^n) &\to [0, \infty) \\ (\omega, f) &\mapsto \frac{1}{m} \sum_{i=1}^{m} \|f(X_i(\omega)) - Y_i(\omega)\|^2. \end{aligned} \tag{2.18}$$

We also refer to the empirical error with the square loss as the mean squared error.

Remark 2.2.5. *Note that in the definition of the empirical error, if we fix a function $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$, the resulting function*

$$\mathcal{E}_{\mathcal{Z}}(f) : \Omega \to [0, \infty), \quad \omega \mapsto \mathcal{E}_{\mathcal{Z}(\omega)}(f) = \mathcal{E}_{\mathbf{z}}(f) \tag{2.19}$$

is $\mathcal{F}/\mathcal{B}([0, \infty))$-measurable as a composition of measurable functions and therefore a random variable.

In the following, we aim to demonstrate the validity of the empirical error as an approximation of the error. Specifically, we verify that as the number of samples m increases, the empirical error $\mathcal{E}_{\mathcal{Z}}$ is close to the error $\mathcal{E}_Z$, with high probability. Utilizing the law of large numbers, we demonstrate the convergence and provide a result on the rate.

Lemma 2.2.6 (Approximation of the Error)**.** *For $f \in L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$ fixed, as the number of samples m increases, the empirical error $\mathcal{E}_{\mathcal{Z}}(f)$ converges $\mathbb{P}$-almost surely to the error $\mathcal{E}_Z(f)$,*

$$\mathbb{P}(\lim_{m \to \infty} \mathcal{E}_{\mathcal{Z}}(f) = \mathcal{E}_Z(f)) = 1. \tag{2.20}$$

Proof. Let $m \in \mathbb{N}$ and let $\mathcal{Z} = ((X_i, Y_i))_{i=1}^m$ be m i.i.d. copies of some data $Z = (X, Y)$. Define the functions $(\xi_i)_{i=1}^m$ by

$$\xi_i : \Omega \to [0, \infty), \quad \omega \mapsto \|f(X_i(\omega)) - Y_i(\omega)\|^2. \tag{2.21}$$

As compositions of measurable functions, ξ_i are $\mathcal{F}/\mathcal{B}([0, \infty))$-measurable and therefore random variables. Further, from construction of $((X_i, Y_i))_{i=1}^m$ and of $(\xi_i)_{i=1}^m$, they are i.i.d. as well. As $f \in L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$ and by our general assumption that Y has finite variance, it holds that

$$\mathbb{E}[\xi_i] = \mathbb{E}[\|f(X_i) - Y_i\|^2] = \mathbb{E}[\|f(X) - Y\|^2] = \mathcal{E}_Z(f) < \infty. \tag{2.22}$$

Therefore, we can apply the law of large numbers to the random variables ξ_i and show that the claim holds,

$$\mathbb{P}(\lim_{m \to \infty} \mathcal{E}_{\mathcal{Z}}(f) = \mathcal{E}_Z(f)) = \mathbb{P}(\lim_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} \xi_i = \mathbb{E}[\xi_1]) = 1. \tag{2.23}$$

□

Theorem 2.2.7 (Approximation Rates of the Error)**.** *Let $f \in L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$ be fixed and assume the variance of $\|f(X) - Y\|^2$ to be finite,*

$$\sigma^2 = Var(\|f(X) - Y\|^2) < \infty. \tag{2.24}$$

Then for any $\epsilon > 0$, it holds that

$$\mathbb{P}(|\mathcal{E}_{\mathcal{Z}}(f) - \mathcal{E}_Z(f)| \geq \epsilon) < \frac{\sigma^2}{m\epsilon^2}. \tag{2.25}$$

Further, if there exists a $M > 0$ such that $\|f(X) - Y\|^2 \leq M$ holds $\mathbb{P}$-a.s., then for any $\epsilon > 0$, it holds that

$$\mathbb{P}(|\mathcal{E}_{\mathcal{Z}}(f) - \mathcal{E}_Z(f)| \geq \epsilon) < 2e^{-\frac{m\epsilon^2}{2(\sigma^2 + \frac{1}{3}M\epsilon)}}. \tag{2.26}$$

Proof. To show the results, we note that for the expectation and the variance of the empirical error of f, it holds that

$$\begin{aligned} \mathbb{E}[\mathcal{E}_{\mathcal{Z}}(f)] &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}[\|f(X_i) - Y_i\|^2] = \mathbb{E}[\|f(X) - Y\|^2] = \mathcal{E}_Z(f) \\ Var(\mathcal{E}_{\mathcal{Z}}(f)) &= \frac{1}{m^2}\sum_{i=1}^{m} Var(\|f(X_i) - Y_i\|^2) = \frac{\sigma^2}{m} < \infty. \end{aligned} \tag{2.27}$$

To proof the first result, we can now apply Chebychevs's inequality and yield the bound. Extending the assumptions with $\|f(X) - Y\|^2 \leq M$ $\mathbb{P}$-a.s. lets us use the Bernstein inequality, which proofs the second result. □

Remark 2.2.8. *Note that the second bound in Theorem 2.2.7 is superior only for m sufficiently large. For further details, refer to Cucker [24].*

2.3 Hypothesis Spaces and Target Functions

In previous sections, a theoretical framework for describing learning problems was developed and a structure was given to the sampling situations in which most of them occur. This section shifts focus to solution strategies, by restricting the set of allowed functions to a subspace of $\mathcal{B}(\mathcal{D}, \mathbb{R}^n)$. By defining the search space in this way, it becomes more feasible to design a search algorithm. Moreover, if the subspace only contains functions dependent on a set of learnable parameters, the learning problem is greatly simplified compared to searching in the full space of functions, $\mathcal{B}(\mathcal{D}, \mathbb{R}^n)$.
The introduction of the concept of hypothesis spaces provides the framework for working with specific solvers. It should be noted that the regression functions previously defined, as well as the functions that minimize the empirical error given random samples, are typically not included in an arbitrarily chosen hypothesis space. This requires the definition of a new target to be learned.

Recall that $L^\infty(\mathcal{D}, \mathbb{R}^n)$ is the Banach space of bounded functions $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$, equipped with the supremum norm,

$$\|f\|_\infty = \sup_{x \in \mathcal{D}} \|f(x)\|. \tag{2.28}$$

Definition 2.3.1 (Hypothesis Space)**.** *Let $\mathcal{H} \subseteq L^\infty(\mathcal{D}, \mathbb{R}^n)$ be a compact and not-empty subspace, then we call $\mathcal{H}$ a hypothesis space.*

We will now define the targets to learn in a hypothesis space, when given some data $Z = (X, Y)$.

Definition 2.3.2 (Target Functions)**.** *Let $\mathcal{H}$ be a hypothesis space, we define a target function $\hat{f}_\mathcal{H}$, as a function which minimizes the error $\mathcal{E}_Z(f)$ over $\mathcal{H}$, that means*

$$\hat{f}_\mathcal{H} \in \arg\min_{f \in \mathcal{H}} \mathcal{E}_Z(f). \tag{2.29}$$

Definition 2.3.3 (Empirical Target Functions)**.** *Let $\mathcal{H}$ be a hypothesis space and let*

$$\mathbf{z} = \mathcal{Z}(\omega) \tag{2.30}$$

be realizations of m samples independently drawn according to the distribution of Z, we define an empirical target function $\hat{f}_{\mathcal{H},\mathbf{z}}$, as a function which minimizes the empirical error $\mathcal{E}_\mathbf{z}(f)$ over $\mathcal{H}$, that means

$$\hat{f}_{\mathcal{H},\mathbf{z}} \in \arg\min_{f \in \mathcal{H}} \mathcal{E}_\mathbf{z}(f). \tag{2.31}$$

Lemma 2.3.4 (Existence of Target Functions)**.** *Let $\mathcal{H}$ be a hypothesis space, then there exists a target function $\hat{f}_\mathcal{H}$ and an empirical target function $\hat{f}_{\mathcal{H},\mathbf{z}}$, which minimize the errors $\mathcal{E}_Z$ and $\mathcal{E}_\mathbf{z}$ over $\mathcal{H}$, respectively.*

Proof. For every $f \in \mathcal{H} \subseteq L^\infty(\mathcal{D}, \mathbb{R}^n)$, we have that $f \in L^2(\mathcal{D}, \mathbb{R}^n; \mathbb{P}_X)$, as $\mathbb{P}_X$ is a finite measure. As Y is assumed to have finite variance, it also holds that $Y \in L^2(\Omega, \mathbb{R}^n; \mathbb{P})$. Restricting the error function to the hypothesis space $\mathcal{H}$

$$\mathcal{E}_Z\big|_\mathcal{H} : \mathcal{H} \to [0, \infty), \quad f \mapsto \mathcal{E}_Z(f), \tag{2.32}$$

allows us to show that it is a continuous map, by applying Cauchy-Schwarz and the inverse triangle inequality. The same can be shown for the empirical error function, when restricted to $\mathcal{H}$

$$\mathcal{E}_\mathbf{z}\big|_\mathcal{H} : \mathcal{H} \to [0, \infty), \quad f \mapsto \mathcal{E}_\mathbf{z}(f), \tag{2.33}$$

by applying basic principles. Noting that $\mathcal{H}$ is compact, this proofs the claim, as $\mathcal{E}_Z\big|_\mathcal{H}$ and $\mathcal{E}_\mathbf{z}\big|_\mathcal{H}$ are continuous maps between metric spaces $(\mathcal{H}, \|.\|_\infty)$ and $([0, \infty), |.|)$, which always attain a minimum on compact sets. □

Remark 2.3.5. *We note that a target function of a hypothesis space does not generally have to be unique. However,if $\mathcal{H}$ is assumed to be convex, a uniqueness result can be*

shown (cf. Cucker [24, section 7]). Also note that a hypothesis space does generally not include the regression function $\hat{f}_Z$, which would simplify the approach.

In Remark 2.2.5, we noted that with our construction of the empirical error, it can be seen as a random variable for fixed $f \in \mathcal{B}(\mathcal{D}, \mathbb{R}^n)$. A similar concept applies to the empirical target functions. We extend the definition of the empirical target function in the same manner as

$$\hat{f}_{\mathcal{H},\mathcal{Z}} : \Omega \to \mathcal{H}, \quad \omega \mapsto \hat{f}_{\mathcal{H},\mathcal{Z}(\omega)} = \hat{f}_{\mathcal{H},\mathbf{z}} \tag{2.34}$$

to conform with the definition of the empirical $\mathcal{E}_{\mathcal{Z}}$. We will further show, that we can construct $\hat{f}_{\mathcal{H},\mathcal{Z}}$ in a way, which is indeed a random variable again.

Lemma 2.3.6 (Measureability of the Empirical Target Functions)**.** *Let $\mathcal{H}$ be a hypothesis space. Then there exists a $\mathcal{F}/\mathcal{B}(\mathcal{H})$-measurable function, which minimizes $\mathcal{E}_{\mathcal{Z}(\omega)}$ for every $\omega \in \Omega$.*

Proof. First, note that any compact metric space is separable. Therefore, the metric space $(\mathcal{H}, \|.\|_\infty)$ is as well. Further, we established, that the empirical error function for a fixed function $f \in \mathcal{H}$,

$$\mathcal{E}_{\mathcal{Z}}(f) : \Omega \to [0, \infty), \quad \omega \mapsto \mathcal{E}_{\mathcal{Z}(\omega)}(f) \tag{2.35}$$

is a random variable and in particular $\mathcal{F}/\mathcal{B}([0,\infty))$-measurable and in the proof of Lemma 2.3.4, we already commented on the continuity of

$$\mathcal{E}_{\mathcal{Z}(\omega)} : \mathcal{H} \to [0, \infty), \quad f \mapsto \mathcal{E}_{\mathcal{Z}(\omega)}(f) \tag{2.36}$$

for any $\omega \in \Omega$. Being measurable in the first and continuous in the second argument, lets us apply the Measurable Maximum Theorem (cf. for instance Aliprantis [32, an adaption of Theorem 18.9]) to the map

$$\Omega \times \mathcal{H} \ni (\omega, f) \mapsto \mathcal{E}_{\mathcal{Z}(\omega)}(f) \in [0, \infty), \tag{2.37}$$

which states that the set-valued function,

$$\Omega \ni \omega \mapsto \arg\min_{f \in \mathcal{H}} \mathcal{E}_{\mathcal{Z}(\omega)}(f) \tag{2.38}$$

admits a measurable selector. That means, it exists a $\mathcal{F}/\mathcal{B}(\mathcal{H})$-measurable function

$$\hat{f}_{\mathcal{H},\mathcal{Z}} : \Omega \to \mathcal{H}, \tag{2.39}$$

such that for every $\omega \in \Omega$,

$$\hat{f}_{\mathcal{H},\mathcal{Z}(\omega)} \in \arg\min_{f \in \mathcal{H}} \mathcal{E}_{\mathcal{Z}(\omega)}(f), \tag{2.40}$$

which completes the proof. □

In the following, we will adopt the construction of the regression function $\hat{f}_{\mathcal{H},\mathcal{Z}}$ as outlined in Lemma 2.3.7.

2.4 Error Decompositions

To establish a more thorough understanding of the role that sampling plays in the hypothesis space selection process, it is necessary to decompose the error. This will reveal the delicate balance between the number of samples and the diversity of the hypothesis space, which is commonly referred to as the bias-variance trade-off. Further, we will introduce the concept of algorithms and derive another error decomposition, that arises for models generated by them. This will aid in the discussion of the generalization capabilities of algorithms, which refers to the ability to learn from samples and perform well on out-of-sample data. Additionally, the role of the optimization algorithm in the process will also be explored. These considerations are crucial for achieving a well-performing model that can generalize to new, unseen data.

Proposition 2.4.1 (Error Decomposition 1)**.** *Let $\mathcal{H}$ be a hypothesis space. For the empirical target function $\hat{f}_{\mathcal{H},\mathcal{Z}}$, we can decompose the error into*

$$\mathcal{E}_Z(\hat{f}_{\mathcal{H},\mathcal{Z}}) = \epsilon_{samp} + \epsilon_{appr} + \epsilon_{irr} \tag{2.41}$$

with

$$\begin{aligned} \epsilon_{samp} &:= \mathcal{E}_Z(\hat{f}_{\mathcal{H},\mathcal{Z}}) - \mathcal{E}_Z(\hat{f}_{\mathcal{H}}) \geq 0 \\ \epsilon_{appr} &:= \mathcal{E}_Z(\hat{f}_{\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_Z) \geq 0 \\ \epsilon_{irr} &:= \mathcal{E}_Z(\hat{f}_Z) \geq 0. \end{aligned} \tag{2.42}$$

Proof. In Lemma 2.3.6, we established the empirical target function as a random variable. Therefore, the map

$$\mathcal{E}_Z(\hat{f}_{\mathcal{H},\mathcal{Z}}) : \Omega \to [0,\infty), \quad \omega \mapsto \mathcal{E}_Z(\hat{f}_{\mathcal{H},\mathcal{Z}(\omega)}) = \mathbb{E}[\|\hat{f}_{\mathcal{H},\mathcal{Z}(\omega)}(X) - Y\|^2] \tag{2.43}$$

is too. The decomposition is a simple rewriting of this random variable. □

Remark 2.4.2. *In the previous Proposition 2.4.1, we introduced the three terms ϵ_{samp}, ϵ_{appr} and ϵ_{irr}, which are named the sample error or variance, the approximation error or squared bias and the irreducible error, with only the term ϵ_{samp} being a random variable again. We remember, that the last term, the irreducible error, is in fact the lower bound on the error function $\mathcal{E}_Z$. It is also independent of the choice of the hypothesis space and the random samples and can not be avoided, hence irreducible. For the other two terms, they can be optimized. This lets us split up the learning problem of finding a small error $\mathcal{E}_Z(\hat{f}_{\mathcal{H},\mathcal{Z}})$, into the two minimization problems of ϵ_{samp} and ϵ_{appr}. While the first is given on the hypothesis space $\mathcal{H}$, the second is independent of the choice of samples.*

Definition 2.4.3 (Covering Numbers)**.** *Let $\mathcal{H}$ be a hypothesis space, let $\delta > 0$ and define the δ-ball around $f \in \mathcal{H}$ by*

$$B_\delta(f) = \{g \in \mathcal{H} \mid \|f - g\|_\infty < \delta\}. \tag{2.44}$$

We define the δ-covering number $\mathcal{N}(\mathcal{H}, \delta)$ to be the minimal amount of δ-balls needed to cover $\mathcal{H}$,

$$\mathcal{N}(\mathcal{H}, \delta) = \arg\min_{k \in \mathbb{N}} \{\exists f_1, ..., f_k \in \mathcal{H} : \bigcup_{i=1}^{k} B_\delta(f_i) \supseteq \mathcal{H}\}. \tag{2.45}$$

Compactness of $\mathcal{H}$ leads to a finite δ-covering numbers, for every δ. Our assumptions on the metric space $(\mathcal{H}, \|.\|_\infty)$ are therefore implying, $\mathcal{N}(\mathcal{H}, \delta) < \infty$.

Theorem 2.4.4 (Sample Error Bounds)**.** *Let $\mathcal{H}$ be a hypothesis space, and let $M > 0$. Assume that for every $f \in \mathcal{H}$ it holds that $\mathbb{P}$-a.s.,*

$$\|f(X) - Y\| \leq M \tag{2.46}$$

and further, let

$$\sigma^2 = Var(\|f(X) - Y\|^2). \tag{2.47}$$

Then, for every $\epsilon > 0$,

$$\mathbb{P}(\mathcal{E}_Z(\hat{f}_{\mathcal{H},\mathcal{Z}}) - \mathcal{E}_Z(\hat{f}_{\mathcal{H}}) \leq \epsilon) \geq 1 - 2\mathcal{N}(\mathcal{H}, \frac{\epsilon}{16M}) e^{-\frac{m\epsilon^2}{8(4\sigma^2 + \frac{1}{3}M^2\epsilon)}}. \tag{2.48}$$

Proof. See Cucker, Theorems C in [24]. □

Under the same setting as in Theorem 2.4.4, we can transform the statement to the following Lemma.

Lemma 2.4.5. *Let ϵ, $\delta > 0$, if the number of samples satisfies*

$$m \geq \frac{8(4\sigma^2 + \frac{1}{3}M^2\epsilon)}{\epsilon^2} \left[\ln(2\mathcal{N}(\mathcal{H}, \frac{\epsilon}{16M})) + \ln(\frac{1}{\delta})\right], \tag{2.49}$$

then it holds that

$$\mathbb{P}(\mathcal{E}_Z(\hat{f}_{\mathcal{H},\mathcal{Z}}) - \mathcal{E}_Z(\hat{f}_{\mathcal{H}}) \leq \epsilon) \geq 1 - \delta. \tag{2.50}$$

Proof. Directly from Theorem 2.4.4. □

For further improvements of the results presented in Theorem 2.4.4 and Lemma 2.4.5, please refer to Cucker [24, Chapter 6], where the covering number is further estimated. Similar work can be done on the approximation error. To get an overview of existing bounds, please see Cucker [24, Chapter II].

Remark 2.4.6 (Bias-Variance Trade-off)**.** *For a fixed hypothesis space $\mathcal{H}$, the approximation error ϵ_{appr} is constant, and the sample error is small with increasing probability*

when the number of samples increases, as seen in Theorem 2.4.4. On the other side, for a fixed number of samples m, the approximation error can only decrease when enlarging the hypothesis space $\mathcal{H}$, while the sample error typically increases. This is often called overfitting on the samples. As a result, there is a trade-off between the choice of the number of samples and the hypothesis space, which is called the bias-variance trade-off.

In the study of machine learning and related literature, the term *algorithm* is used to describe a procedure for mapping given realizations of random samples $\mathbf{z}$ to a function in a specified hypothesis space $\mathcal{H}$. Our goal is it to find an algorithm, which has small empirical error w.r.t. $\mathbf{z}$. This is the same, as searching for an algorithm, which approximates the empirical target function on the hypothesis space $f_{\mathcal{H},\mathbf{z}}$.
In context of algorithms, the empirical error is commonly referred to as the in-sample error or the training error, and it measures the error of the algorithm when applied to the so-called training set, $\mathbf{z}$. As noted already, we are typically not equipped with the distribution of the data Z, so while the empirical error is computable a posteriori, the error itself is not.. While we use the empirical error as an approximation for the error, this might not always be a good indication of the performance on the overall data, especially for relatively small amounts of samples. Therefore, it is common practise to measure the generalization capabilities of the algorithm by measuring the empirical error on other realizations of samples that were not included in $\mathbf{z}$, often referred to as out-of-sample error or the validation error.
Assuming that we have an algorithm that generates the empirical target function (or functions close to it, in the sense of the distance between their empirical errors), the optimization problem comes down to finding a good balance between the bias and variance, as described in Remark 2.4.6. In this case, we can adjust the number of samples and choice of hypothesis space, by measuring the validation error as an approximation of the error $\mathcal{E}_Z(f_{\mathcal{H},\mathcal{Z}})$.

Definition 2.4.7 (Learning Algorithm)**.** *Let $\mathcal{H}$ be a hypothesis space, define a learning algorithm on $\mathcal{H}$ as a measurable map*

$$\mathcal{A} : \bigcup_{m\in\mathbb{N}} (\mathcal{D}, \mathbb{R}^n)^m \to \mathcal{H}, \quad \mathbf{z} \mapsto \mathcal{A}(\mathbf{z}) \tag{2.51}$$

and call $\mathcal{A}(\mathbf{z})$ a model generated on $\mathbf{z}$.

Remark 2.4.8. *Note that the model produced by the learning algorithm can be viewed as a random variable, denoted by*

$$\Omega \ni \omega \mapsto \mathcal{A}(\mathcal{Z}(\omega)) = \mathcal{A}(\mathbf{z}). \tag{2.52}$$

To adhere with the developed theory, we will therefore use $\mathcal{A}(\mathcal{Z})$. Furthermore, for the sake of readability, we denote by

$$f_{\mathcal{A},\mathcal{Z}} = \mathcal{A}(\mathcal{Z}) \tag{2.53}$$

the model generated by learning algorithm $\mathcal{A}$ on random samples $\mathcal{Z}$.

Let us now derive another decomposition of errors, where we investigate on the error of an actual model, which was generated by an algorithm.

Proposition 2.4.9 (Error Decomposition 2)**.** *Let $\mathcal{H}$ be a hypothesis space, let $\mathcal{A}$ be an algorithm on $\mathcal{H}$. Then it holds that*

$$\mathcal{E}_Z(f_{\mathcal{A},\mathcal{Z}}) - \mathcal{E}_Z(\hat{f}_Z) \leq \epsilon^{opt} + 2\epsilon^{gen} + \epsilon^{appr} \tag{2.54}$$

with

$$\begin{aligned} \epsilon^{opt} &:= \mathcal{E}_{\mathcal{Z}}(f_{\mathcal{A},\mathcal{Z}}) - \mathcal{E}_{\mathcal{Z}}(\hat{f}_{\mathcal{H},\mathcal{Z}}) \geq 0 \\ \epsilon^{gen} &:= \sup_{f\in\mathcal{H}} |\mathcal{E}_Z(f) - \mathcal{E}_{\mathcal{Z}}(f)| \geq 0 \\ \epsilon^{appr} &:= \mathcal{E}_Z(\hat{f}_{\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_Z) \geq 0. \end{aligned} \tag{2.55}$$

Proof. The error bound can be computed by decomposing and bounding each error term, by the defined ϵ^{opt}, ϵ^{gen} and ϵ^{appr},

$$\begin{aligned} &\mathcal{E}_Z(f_{\mathcal{A},\mathcal{Z}}) - \mathcal{E}_Z(\hat{f}_Z) \\ &= \underbrace{\mathcal{E}_Z(f_{\mathcal{A},\mathcal{Z}}) - \mathcal{E}_{\mathcal{Z}}(f_{\mathcal{A},\mathcal{Z}})}_{\leq \epsilon^{gen}} + \underbrace{\mathcal{E}_{\mathcal{Z}}(f_{\mathcal{A},\mathcal{Z}}) - \mathcal{E}_{\mathcal{Z}}(\hat{f}_{\mathcal{H}})}_{\leq \epsilon^{opt}} + \underbrace{\mathcal{E}_{\mathcal{Z}}(\hat{f}_{\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_{\mathcal{H}})}_{\leq \epsilon^{gen}} + \underbrace{\mathcal{E}_Z(\hat{f}_{\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_Z)}_{= \epsilon^{appr}} \\ &\leq \epsilon^{opt} + 2\epsilon^{gen} + \epsilon^{appr}. \end{aligned} \tag{2.56}$$

□

Remark 2.4.10. *In the previous Proposition 2.4.9, we introduced two new terms ϵ_{opt} and ϵ_{gen}, which we call optimization error and generalization error. Again, only one term is a random variable, the optimization error. We discussed already in Remark 2.4.6 how the approximation error can be optimized and that enlarging the hypothesis space $\mathcal{H}$ will decrease it, while the sample error increases. Similar behavior can be observed with the generalization error, which cannot be controlled for a too-big $\mathcal{H}$ and decreases with a growing number of samples, leading to another form of the bias-variance trade-off. As stated earlier, the best choice of an algorithm is the empirical target function, and thus the definition of the optimization error is natural and decreases with an improvement in the choice of the algorithm in that sense. Note that the optimization error primarily depends on the choice of the algorithm. However, with an enlarged hypothesis space, it is typically harder to find good algorithms and computationally more expensive to train them with an increasing number of samples. In Chapter 3 we will see gradient-based methods as an example of such algorithms, giving more structure to the problem of computational effort. For further investigations into the different error terms, see Berner [69, Chapter 1] and Anthony [19].*

3 Neural Networks

This chapter is devoted to give a brief introduction to the wide field of neural networks and discuss some of their most important properties and limitations. Additionally, we will see with the hypothesis space of neural networks a concrete example of how we optimize the choice of aproximators given some samples. The motivation behind neural networks has its origin in the study of human brains, where we consider biological neural networks. A first step towards describing these structures, was done by McCulloch and Pitts [1] in 1943, who proposed a computational model for a biological neuron. The McCulloch and Pitts neuron, is defined by

$$\mathbb{R}^d \ni x \mapsto \mathbb{1}_{\mathbb{R}_+}(\sum_{i=1}^{d} w_i x_i - \theta), \tag{3.1}$$

where $d \in \mathbb{N}$ is the input dimension and $w_i, \theta \in \mathbb{R}$ are the weights and the threshold, which must be surpassed to signal an activation from the neuron. By allowing the output of neurons to form the input of another one, we connect these functions into a network. This mathematical structure laid the ground for the neural networks we are nowadays working with.

Recently, neural networks and deep learning (cf. Goodfellow [41]) have gained much attention due to their success in a variety of fields, such as games (cf. for instance Silver [54] and Berner [55]), image classification (cf. for instance Rombach [72] and Dosovitskiy [64]), in natural language application (cf. for instance Brown [60] and Thoppilan [74]), and in the natural sciences (cf. for instance Jumper [65]) and have a central position in modern machine learning (cf. for instance LeCun [47]). In this chapter, we will focus on the simpler models, fully-connected feedforward neural networks, which are also called multilayer perceptrons (cf. Rosenblatt [2]).

3.1 Definition

We will begin by presenting the definition of neural networks and providing a detailed explanation of their components. To aid understanding, we will use visualizations to make the topic more concrete.

Definition 3.1.1 (Fully-Connected Feedforward Neural Network)**.** *Let $d, L \in \mathbb{N}$. A (fully-connected feedforward) neural network with input dimension d and L layers is*

determined by its parameters θ, a sequence of matrix-vector tuples

$$\theta = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^{L} \in \bigtimes_{\ell=1}^{L} (\mathbb{R}^{N_\ell \times N_{\ell-1}}, \mathbb{R}^{N_\ell}), \tag{3.2}$$

where $N = (N_0, \dots, N_L) \in \mathbb{N}^{L+1}$ and $N_0 = d$. The numbers N_ℓ are called neurons in the ℓ-th layer, while we refer to the 0-th layer, as the input layer and L-th layer as the output layer. For $\ell \in \{1, \dots, L-1\}$, we refer to the ℓ-th layer also as the ℓ-th hidden layer. We define the number of parameters as

$$p(N) := \sum_{\ell=1}^{L} N_\ell N_{\ell-1} + N_\ell, \tag{3.3}$$

which lets us represent the parameters as a vector in $\mathbb{R}^{p(N)}$. Given a function $\varrho : \mathbb{R} \to \mathbb{R}$, we define the associated realization function by

$$\mathcal{R}_{N,\varrho} : \mathbb{R}^d \times \mathbb{R}^{p(N)} \to \mathbb{R}^{N_L}, \quad (x, \theta) \mapsto \Phi^{(L)}(x, \theta), \tag{3.4}$$

where

$$\begin{aligned} &\Phi^{(0)}(x, \theta) := x \\ &\Phi^{(\ell)}(x, \theta) := \varrho(W^{(\ell)} \Phi^{(\ell-1)}(x, \theta) + b^{(\ell)}), \quad \ell \in \{1, \dots, L-1\} \quad \textit{and} \\ &\Phi^{(L)}(x, \theta) := W^{(L)} \Phi^{(L-1)}(x, \theta) + b^{(L)}, \end{aligned} \tag{3.5}$$

and ϱ is understood to act component-wise and is referred to as the activation function. We refer to $a = (N, \varrho)$ as the network architecture and we further call the matrices $W^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and vectors $b^{(\ell)} \in \mathbb{R}^{N_\ell}$, weight matrices and bias vectors. The depth and width of an architecture are given by the number of layers L and the maximal number of neurons $\|N\|_\infty$. We call an architecture deep, if $L > 2$ and call it shallow if $L = 2$. Moreover, we refer to the architecture determining layers L, activation function ϱ and neurons N, as hyper-parameters.

The term neural network will always refer to the Definition 3.1.1 in this thesis. It is noteworthy that a neural network with $L = 1$ layers is equivalent to an affine linear functions.

Remark 3.1.2 (Activation Functions)**.** *We will see later, that in fact all continuous functions, which are neither linear nor a polynomial, might be a valid choice for activation functions. However, in practise there are only a few that are used and researched on frequently. To give a small overview, we present two of the most common ones, which describe an overall class of activation functions, by including simple modifications. The rectified linear unit (ReLU) function, is defined by the map*

$$\mathbb{R} \ni x \mapsto \varrho_R(x) := \max\{0, x\} \tag{3.6}$$

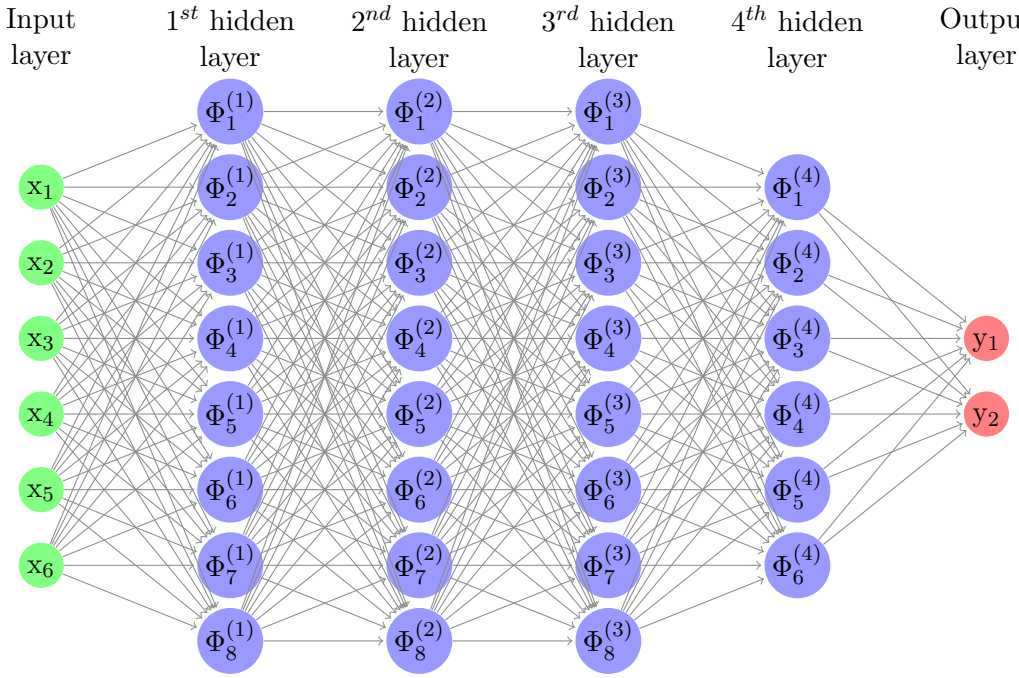


Figure 3.1: Illustration of a neural network with 5 layers and neurons N=(6,8,8,8,6,2).

and sigmoidal functions are continuous functions $f : \mathbb{R} \to \mathbb{R}$*, such that*

$$\lim_{x \to -\infty} f(x) = 0 \quad \textit{and} \quad \lim_{x \to \infty} f(x) = 1. \tag{3.7}$$

Common modified forms of the two mentioned functions are the parametric ReLU function, $\mathbb{R} \ni x \mapsto \varrho_{R,a}(x) := \max\{ax, x\}$ *for some parameter* $a > 0$ *and the hyperbolic tangent function,* $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$*, which has limits* $\lim_{x \to -\infty} tanh(x) = -1$ *and* $\lim_{x \to \infty} tanh(x) = 1$*.*

Every neural network admits an underlying directed acyclic graph $\mathcal{G} = (V, E)$, depending on its architecture $a = (N, \varrho)$. The set of vertices V is given by

$$V = \{\Phi_i^{(\ell)} \,|\, \ell \in \{0, \dots, L\},\, i \in \{1, \dots, N_\ell\}\} \tag{3.8}$$

and the set of edges E by

$$E = \{(\Phi_i^{(\ell)}, \Phi_j^{(\ell+1)}) \,|\, \Phi_i^{(\ell)}, \Phi_j^{(\ell+1)} \in V\}. \tag{3.9}$$

An illustration of a resulting graph can be seen in Figure 3.1. Further, a vertex $\Phi_i^{(\ell)}$ can be interpreted as the i-th neuron in the ℓ-th layer and the edges are describing the computation flow through the network. Note that the recursive definition of the realization function of a neural network, describes a graph of this form, with edge weights

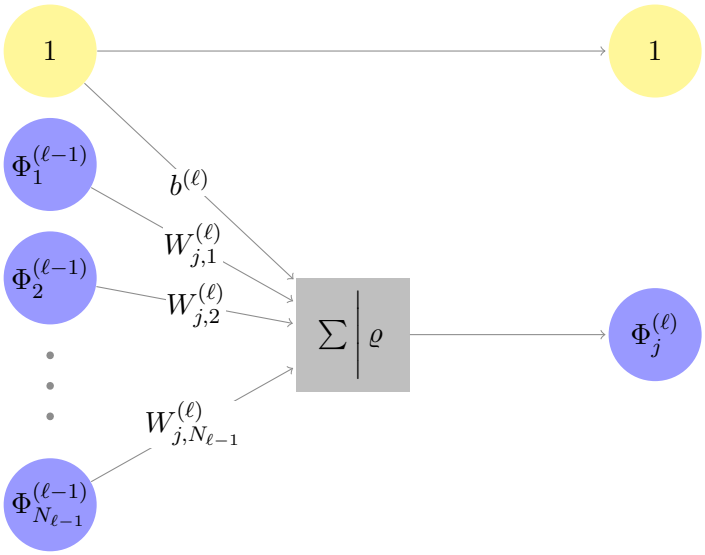


Figure 3.2: Illustration of the bias-free computation between layers, for $\ell \in \{1, \dots, L-1\}$ and $j \in N_\ell$. For $\ell = L$, the connection between the 1's will not appear.

corresponding to the weight matrix, if we neglect the bias vectors. In fact we can always write a neural network without bias vectors, by refining the input and parameters to

$$x \to \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad (W^{(\ell)}, b^{(\ell)}) \to \begin{bmatrix} W^{(\ell)} & b^{(\ell)} \\ 0 & 1 \end{bmatrix} \tag{3.10}$$

for every $\ell \in \{1, \dots, L-1\}$ and the parameters of the last layer to

$$(W^{(L)}, b^{(L)}) \to \begin{bmatrix} W^{(L)} & b^{(L)} \end{bmatrix}. \tag{3.11}$$

In this manner, we can visualize the computation flow between vertices as seen in Figure 3.2.

3.2 Universality

Under certain weak conditions on the activation function, shallow and deep neural networks are able to approximate continuous functions on compact subset of $\mathbb{R}^d$ arbitrarily precise. This forms one of the most important approximation properties for neural networks, called universality, which was first shown by Hornik [7] and Cybenko [6] and points great interest towards the space of neural networks. While there are multiple forms of the statement (cf. for instance Leshno [11] and Scarselli [16]), we will focus on the format from Cybenko [6].

Remark 3.2.1 (Setting of Approximation Space)**.** *In the sense of functional analysis, to speak of approximation results, we need to equip the space of goal functions with a*

topology. We are especially interested in the space of continuous functions $\mathcal{C}(K)$ for a compact $K \subset \mathbb{R}^d$, which we endow with the uniform norm, $\|.\|_\infty$. Note that in this setting, by the representation theorem of Riesz (cf. Rudin [31, Theorem 6.19]), the topological dual space of $\mathcal{C}(K)$ is the space of all signed Borel measures on K, which we will denote by $\mathcal{M}(K)$.

Definition 3.2.2 (Discriminatory Functions)**.** *Let $d \in \mathbb{N}$ and let $K \subset \mathbb{R}^d$ be compact. We call a continuous function $f : \mathbb{R} \to \mathbb{R}$ discriminatory w.r.t. K, if the only $\mu \in \mathcal{M}(K)$ for which*

$$\int_K f(w \cdot x - b) d\mu(x) = 0 \tag{3.12}$$

holds for every $w \in \mathbb{R}^d$ and every $b \in \mathbb{R}$, is $\mu = 0$.

We already introduced two examples of discriminatory functions in Remark 3.1.2. In fact, we will see in the following two Lemmas 3.2.4 and 3.2.5, that the ReLU function and each sigmoidal function are discriminatory.

Proposition 3.2.3. *Let $d \in \mathbb{N}$ and let $K \in \mathbb{R}^d$ be compact. If for a measure $\mu \in \mathcal{M}(K)$ it holds, that for every $w \in \mathbb{R}^d$ and every $b_1, b_2 \in \mathbb{R}$*

$$\int_K \mathbb{1}_{[b_1,b_2]}(w \cdot x) d\mu(x) = 0, \tag{3.13}$$

then $\mu = 0$.

Proof. Let $f : \mathbb{R} \to \mathbb{R}$ be a step function, i.e. we can write the function for every $x \in \mathbb{R}$ as

$$f(x) = \sum_{i=0}^{n} \alpha_i \mathbb{1}_{A_i}(x), \tag{3.14}$$

where $n \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$ and A_i are intervals on $\mathbb{R}$. As each interval A_i in $\mathbb{R}$ is either closed, i.e. $A_i = [b_i, b_{i+1}]$ for some $b_i, b_{i+1} \in \mathbb{R}$, or can be expressed as the union of a sequence of closed intervals, and by the monotone convergence theorem, it holds for every interval $A_i \in \mathbb{R}$ and every $w \in \mathbb{R}^d$ that

$$\int_K \mathbb{1}_{A_i}(w \cdot x) d\mu(x) = 0, \tag{3.15}$$

and so by linearity of the integral we can calculate,

$$\int_K f(w \cdot x) d\mu(x) = \sum_{i=0}^{n} \alpha_i \int_K \mathbb{1}_{A_i}(w \cdot x) d\mu(x) = 0. \tag{3.16}$$

Further, every bounded continuous function $g \in \mathcal{C}(\mathbb{R}) \cap L^\infty(\mathbb{R})$ is the limit of step functions and so by the dominated convergence theorem it holds for every $w \in \mathbb{R}^d$ that

$$\int_K g(w \cdot x) d\mu(x) = 0. \tag{3.17}$$

This holds especially for the bounded continuous functions *sin* and *cos* and therefore,

$$0 = \int_K \cos(w \cdot x) + i \sin(w \cdot x) d\mu(x) = \int_K e^{iw \cdot x} d\mu(x). \tag{3.18}$$

And so the Fourier transform of the measure μ must be 0, which can only happen if we already had $\mu = 0$ (cf. Rudin [10, p. 176]). □

With the Proposition 3.2.3 we can now go on to show the two Lemmas, which will show that the ReLU function and sigmoidal functions are discriminatory.

Lemma 3.2.4 (Rectified Linear Unit Function is Discriminatory)**.** *Let $d \in \mathbb{N}$ and let $K \in \mathbb{R}^d$ be compact. Then the function defined by*

$$\mathbb{R} \ni x \mapsto \varrho_R(x) := \max\{0, x\} \tag{3.19}$$

is discriminatory w.r.t. K.

Proof. For every $w \in \mathbb{R}$ and $b_1, b_2 \in \mathbb{R}$, $b_1 < b_2$, we define the function $I_{w,b_1,b_2} : \mathbb{R} \to \mathbb{R}$ by

$$I_{w,b_1,b_2}(x) := \varrho_R(wx - wb_1 + 1) - \varrho_R(wx - wb_1) - \varrho_R(wx - wb_2) + \varrho_R(wx - wb_2 - 1). \tag{3.20}$$

This function can be rewritten to

$$I_{w,b_1,b_2}(x) = \begin{cases} 0 & \text{if } x \leq b_1 - \frac{1}{w} \\ w(x - b_1 + \frac{1}{w}) \leq 1 & \text{if } b_1 - \frac{1}{w} < x < b_1 \\ 1 & \text{if } b_1 \leq x \leq b_2 \\ w(\frac{1}{w} - x - b_2) \leq 1 & \text{if } b_2 < x < b_2 + \frac{1}{w} \\ 0 & \text{if } x \geq b_2 + \frac{1}{w}, \end{cases} \tag{3.21}$$

which makes it obvious, that for every $x \in \mathbb{R}$, $\lim_{w \to \infty} I_{w,b_1,b_2}(x) = \mathbb{1}_{[b_1,b_2]}(x)$ holds. By construction of I_{w,b_1,b_2}, we can approximate any indicator function $\mathbb{1}_{[b_1,b_2]}$ by sums of ReLU functions point-wise. Let now $\mu \in \mathcal{M}(K)$ be a measure, such that for every $w \in \mathbb{R}^d$ and every $b \in \mathbb{R}$ it holds that

$$\int_K \varrho_R(w \cdot x - b) d\mu(x) = 0, \tag{3.22}$$

then by the monotone convergence theorem it holds for every $w \in \mathbb{R}^d$ and every $b_1, b_2 \in \mathbb{R}$ that

$$\int_K \mathbb{1}_{[b_1,b_2]}(w \cdot x) \mu(x) = 0. \tag{3.23}$$

With Proposition 3.2.3, the claim follows. □

Lemma 3.2.5 (Sigmoidal Functions are Discriminatory)**.** *Let $d \in \mathbb{N}$ and let $K \in \mathbb{R}^d$ be compact. Any continuous function $f : \mathbb{R} \to \mathbb{R}$ with*

$$\lim_{x \to -\infty} f(x) = 0 \quad \textit{and} \quad \lim_{x \to \infty} f(x) = 1, \tag{3.24}$$

is called sigmoidal and is discriminatory w.r.t. K.

Proof. Let f be a sigmoidal function. For $w \in \mathbb{R}^d, b \in \mathbb{R}, \theta \in \mathbb{R}$ and $\lambda > 0$, we consider the function

$$\mathbb{R}^d \ni x \mapsto g_{w,b,\theta,\lambda}(x) := f(\lambda(w \cdot x - b) + \theta), \tag{3.25}$$

which possess the limit behavior,

$$\lim_{\lambda \to \infty} g_{w,b,\theta,\lambda}(x) = \begin{cases} 0 & \text{if } w \cdot x - b < 0 \\ f(\theta) & \text{if } w \cdot x - b = 0 \\ 1 & \text{if } w \cdot x - b > 0, \end{cases} \tag{3.26}$$

by the asymptotically properties of f. As f and therefore also $g_{w,b,\theta,\lambda}$ are bounded, we can apply the dominated convergence theorem, to yield for every $\mu \in \mathcal{M}(K)$, that

$$\int_K g_{w,b,\theta,\lambda} d\mu \to \int_K \mathbb{1}_{H_{w,b,>}} d\mu + \int_K f(\theta) \mathbb{1}_{H_{w,b,=}} d\mu \tag{3.27}$$

for $\lambda \to \infty$, where $H_{w,b,>} := \{x \in K | w \cdot x - b > 0\}$ and $H_{w,b,=} := \{x \in K | w \cdot x - b = 0\}$. We now assume, that

$$\int_K g_{w,b,\theta,\lambda} d\mu = 0 \tag{3.28}$$

holds for every $w \in \mathbb{R}^d$ and every $b \in \mathbb{R}$, which coincides with the assumption in the definition of discriminatory, as we can write

$$g_{w,b,\theta,\lambda}(x) = f((\lambda w) \cdot x - (\lambda b - \theta)). \tag{3.29}$$

Under this assumption, we observe that

$$0 = \int_K \mathbb{1}_{H_{w,b,>}} d\mu + \int_K f(\theta) \mathbb{1}_{H_{w,b,=}} d\mu \to \int_K \mathbb{1}_{H_{w,b,>}} d\mu \tag{3.30}$$

for $\theta \to -\infty$ and conclude that the later integral must also be equal 0 for every $w \in \mathbb{R}^d$ and every $b \in \mathbb{R}$. Note that for any $w \in \mathbb{R}^d, b_1, b_2 \in \mathbb{R}, b_1 < b_2$ fixed we have

$$0 = \int_K \mathbb{1}_{H_{w,b1,>}} d\mu - \int_K \mathbb{1}_{H_{w,b2,>}} d\mu = \int_K \mathbb{1}_{[b_1,b_2]}(w \cdot x) d\mu(x), \tag{3.31}$$

and thus the claim follows with Proposition 3.2.3. □

Now that we have seen two activation functions, which are discriminatory, we want to state the theorem on universality.

Theorem 3.2.6 (Universal Approximation Theorem)**.** *Let $d \in \mathbb{N}$, let $K \in \mathbb{R}^d$ be compact and let $\varrho : \mathbb{R} \to \mathbb{R}$ be any continuous discriminatory function. Then the set*

$$\mathcal{N}_\varrho := \bigcup_{n \in \mathbb{N}} \left\{ \mathcal{R}_{(d,n,1),\varrho}(\cdot, \theta) \,|\, \theta \in \mathbb{R}^{p((d,n,1))} \right\} \tag{3.32}$$

of realization functions of two layer neural networks with input d and activation function ϱ, is dense in $\mathcal{C}(K)$. In other words, given a function $f \in \mathcal{C}(K)$ and $\epsilon > 0$, there exists $n \in \mathbb{N}$ and $\theta \in \mathbb{R}^{p((d,n,1))}$, such that

$$\sup_{x \in K} |\mathcal{R}_{(d,n,1),\varrho}(x, \theta) - f(x)| < \epsilon. \tag{3.33}$$

Proof. Note that any realization function of a neural network with continuous activation function can be written as a composition of continuous functions. Indeed, we can rewrite the definition towards a composition of affine linear functions and the activation function. More precisely, given parameters $((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^L = \theta \in \mathbb{R}^{p(N)}$ for fixed $L \in \mathbb{N}$ and $N \in \mathbb{N}^{L+1}$, let us define a sequence of affine linear functions $(\phi_\theta^{(\ell)})_{\ell=1}^L$ by

$$\phi_\theta^{(\ell)} : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, \quad x \mapsto \phi_\theta^{(\ell)}(x) := W^{(\ell)} x + b^\ell. \tag{3.34}$$

By abuse of notation, we assume that the activation function acts component-wise and so we can rewrite the realization function as

$$\mathcal{R}_{N,\varrho}(\cdot, \theta) = \phi_\theta^{(L)} \circ \varrho \circ \phi_\theta^{(L-1)} \circ \cdots \circ \varrho \circ \phi_\theta^{(1)}. \tag{3.35}$$

Therefore, the realization function $\mathcal{R}_{N,\varrho}(\cdot, \theta)$ must be continuous itself. This holds especially true for any realization function of a two-layer neural network and moreover if restricted to K and we can finally conclude that $\mathcal{N}_\varrho \subseteq \mathcal{C}(K)$ holds. Assume now, that $\mathcal{N}_\varrho$ is not dense in $\mathcal{C}(K)$, then we can find some $g \in \mathcal{C}(K) \setminus \overline{\mathcal{N}_\varrho}$. Moreover, by the theorem of Hahn-Banach (cf. Rudin [31, Theorem 5.19]), there exists a functional

$$0 \neq G \in \mathcal{C}(K)' = \mathcal{M}(K), \tag{3.36}$$

such that $G = 0$ on the set $\mathcal{N}_\varrho$. Further, note that

$$\mathbb{R}^d \ni x \mapsto \varrho_{w,b}(x) := \varrho(w \cdot x - b) \in \mathcal{N}_\varrho, \tag{3.37}$$

for some $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$, which implies $G(\varrho_{w,b}) = 0$ for every $w \in \mathbb{R}^d$ and every $b \in \mathbb{R}$. Therefore, there exists a non-zero measure $\mu \in \mathcal{M}(K)$, such that

$$\int_K \varrho_{w,b} d\mu = 0 \tag{3.38}$$

for every $w \in \mathbb{R}^d$ and every $b \in \mathbb{R}$. But this is a contradiction to the assumption, that ϱ is discriminatory and so we finish the proof. □

Remark 3.2.7. *Under additional assumptions on the activation function, the above statement can be extended to hold not only for shallow neural networks but also for deep neural networks. Note that there are many variations of the statement in Theorem 3.2.6, slightly differing in assumptions and statements (cf. for instance Leshno [11], Scarselli [16] and Hornik [7]). Especially Leshno [11] states, that universality already holds for any activation function which is neither linear nor polynomial and even extends to set of measurable functions $\mathcal{B}(K)$.*

Remark 3.2.8. *Theorem 3.2.6 shows, that we can approximate any continuous function on a compact set with a neural network up to any degree of accuracy. However, we are still lacking an understanding of how many neurons are required in a neural network, to achieve a certain approximation accuracy, as the theorem only assumes the number of neurons in the hidden layer to be $N_1 \in \mathbb{N}$. Further, we also did not gain any insight into an applicable algorithm, which could deliver the appropriate parameters θ, to achieve any approximation result. Regarding the algorithm, we will see in the next section the parameter optimization algorithm, gradient descent, which is widely used in practise. On the other hand, we will not go into details on methods to bound the number of required neurons N w.r.t. the number of layers L and varying additional assumptions, but reference to the works as Anthony [19], Mhaskar [12], Blum [9] and Maiorov [20].*

3.3 Hypothesis Space of Neural Networks

The previous section established that neural networks possess optimal approximation properties, which suggests a convenient hypothesis space for learning problems. We therefore propose the set of realization functions of neural networks with a fixed architecture as the hypothesis space. To fully conform with Definition 2.3.1, we need this space to be a compact subspace of the space of bounded measurable functions. Clearly, without any further assumptions, we are not fulfilling the conditions of Definition 2.3.1. Allowing only neural network with parameters bounded by a constant and restricting the input to a compact subset, we assure the conditions to be satisfied. Further, we introduce the most commonly used solution methods for the newly stated learning problem on the hypothesis space of neural networks, optimization algorithms based on gradients.

Definition 3.3.1 (Hypothesis Spaces of Neural Networks)**.** *Let $d, n, L \in \mathbb{N}$, let $\mathcal{D} \subseteq \mathbb{R}^d$ be compact, let $N = (N_0, \dots, N_L) \in \mathbb{N}^{L+1}$, such that $N_0 = d$ and $N_L = n$, let $R > 0$, let $q \in [1, \infty]$ and let $\varrho : \mathbb{R} \to \mathbb{R}$ be continuous and non-polynomial. We define the hypothesis space of neural networks on $\mathcal{D}$ with the given architecture $a = (N, \varrho)$ and the hyper-parameters R and q by*

$$\mathcal{N}_{N,\varrho,R,q}(\mathcal{D}, \mathbb{R}^n) := \left\{ \mathcal{R}_{N,\varrho}(\cdot, \theta)|_{\mathcal{D}} \;\middle|\; \theta \in \mathbb{R}^{p(N)} \text{ and } \|\theta\|_{\ell^q} \leq R \right\}, \tag{3.39}$$

where $\|\theta\|_{\ell^q} = \max\limits_{1 \leq i \leq L} \max \left\{ \|W^{(\ell)}\|_{\ell^q}, \|b^{(\ell)}\|_{\ell^q} \right\}$.

Note that we allow parameters $\theta \in \mathbb{R}^{p(N)}$ with $\|\theta\|_{\ell^q} \leq R$ and especially also $\theta_i = 0$ in certain $i \in p(N)$. This enables us to set weight matrix and bias vector entries of certain layers ℓ to 0. By doing so, we can construct not only neural networks with number of neurons N, but with any number of neurons $\tilde{N} = (N_0, \tilde{N}_1, \ldots, \tilde{N}_{L-1}, N_L) \in \mathbb{N}^{L+1}$ with $\tilde{N}_\ell \leq N_\ell$ for every $\ell \in \{1, \ldots, L-1\}$. Therefore, we can see the hypothesis space of neural networks as the set of all realization functions of neural networks with number of neurons up to N.
The hyper-parameter q determines the regularization of the weights. Throughout the rest of the chapter we will assume the setting from Definition 3.3.1 to hold with $q = \infty$ and denote

$$\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n) := \left\{ \mathcal{R}_{N,\varrho}(\cdot, \theta)|_{\mathcal{D}} \;\middle|\; \theta \in [-R, R]^{p(N)} \right\}, \tag{3.40}$$

as the hypothesis space under these assumptions, if not said otherwise. To assure, that Definition 3.3.1 conforms with the conditions on hypothesis spaces from Definition 2.3.1, we will show that $\mathcal{N}_{(N,\varrho,R)}(\mathcal{D}, \mathbb{R}^n)$ is a compact subspace.

Lemma 3.3.2 (Parameter-Bounded Neural Networks are Compact)**.** *Assuming the setting of Definition 3.3.1,*

$$\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n) \subseteq \mathcal{B}(\mathcal{D}, \mathbb{R}^n) \tag{3.41}$$

is a non-empty compact subset.

Proof. Note that we can always find a realization function of a neural network in the space $\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n)$, e.g. by taking $\theta = R^{p(N)}$ and especially $\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n) \neq \emptyset$. Further, note that a realization function of a neural network with a continuous activation function is always continuous itself, as a composition of continuous functions (cf. proof of Theorem 3.2.6). This is as well true for the special case of bounded parameters and in summary we can state that

$$\emptyset \neq \mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n) \subseteq \mathcal{C}(\mathcal{D}, \mathbb{R}^n) \tag{3.42}$$

holds. We will move on to show the compactness, by first defining the map

$$\psi : [-R, R]^{p(N)} \to \mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n), \quad \theta \mapsto \mathcal{R}_{N,\varrho}(\cdot, \theta)|_{\mathcal{D}}\,, \tag{3.43}$$

which is continuous w.r.t. the topologies induced by the norm

$$\begin{aligned} \|\cdot\|_N : [-R, R]^{p(N)} &\to [0, \infty) \\ \theta = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^L &\mapsto \max_{\ell \in \{0,\ldots,L\}} \|W^{(\ell)}\|_{max} + \max_{\ell \in \{0,\ldots,L\}} \|b^{(\ell)}\|_\infty \end{aligned} \tag{3.44}$$

on $[-R, R]^{p(N)}$ and the uniform norm $\|\cdot\|_\infty$ on $\mathcal{C}(\mathcal{D}, \mathbb{R}^n)$ (cf. Petersen [67, Proposition 5.1]). Since $[-R, R]^{p(N)}$ is compact in the topology induced by the Euclidean norm and the fact that all norms on finite-dimensional vector spaces induce the same topology, $[-R, R]^{p(N)}$ must be compact in the topology induced by the norm $\|\cdot\|_N$ as well. By definition of $\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n)$, the map ψ is surjective. Further by continuity, the image of

compact sets is compact, and therefore

$$\psi\left([-R,R]^{p(N)}\right) = \mathcal{N}_{N,\varrho,R}(\mathcal{D},\mathbb{R}^n) \subseteq \mathcal{C}(\mathcal{D},\mathbb{R}^n) \subseteq \mathcal{B}(\mathcal{D},\mathbb{R}^n) \tag{3.45}$$

is a compact subset. □

Now that we established the hypothesis space of neural networks, we can turn to the optimization problem on samples. We will assume the setting of Definition 2.2.1 for realizations of samples, which we again denote by

$$\mathbf{z} = ((x_i, y_i))_{i=1}^m \in (\mathcal{D},\mathbb{R}^n)^m. \tag{3.46}$$

As a realization function of a neural network is again a function in $\mathcal{B}(\mathcal{D},\mathbb{R}^n)$ for fixed θ, the optimization problem can be stated as the quest to find an optimal choice of parameters $\hat{\theta}$ that minimizes the empirical error. Note that under our assumptions, the space $\mathcal{N}_{N,\varrho,R}(\mathcal{D},\mathbb{R}^n)$ is compact, and we showed before that, therefore, an optimal $\hat{\theta}$ exists, while not necessarily unique. This allows us to talk of optimization strategies or algorithms. Note that we focused in Chapter 2 on the quadrature loss in the definition of the empirical error function. However, we will state the arising minimization problem for general loss functions as a measure of performance.

Definition 3.3.3 (Optimization Problem, Neural Network Version)**.** *Let $s \in \mathbb{N}_0$ and let $\varrho \in \mathcal{C}^s(\mathbb{R},\mathbb{R})$ be non-polynomial. For a given loss function $\mathcal{L} \in \mathcal{C}^s(\mathbb{R}^n \times \mathbb{R}^n, [0,\infty))$, we define the maps*

$$\upsilon_i : [-R,R]^{p(N)} \to [0,\infty), \quad \theta \mapsto \mathcal{L}(\mathcal{R}_{N,\varrho}(x_i,\theta), y_i) \tag{3.47}$$

and the empirical error w.r.t. $\mathbf{z}$ and the loss function $\mathcal{L}$ by

$$\Upsilon : [-R,R]^{p(N)} \to [0,\infty), \quad \theta \mapsto \frac{1}{m}\sum_{i=1}^m \nu_i(\theta). \tag{3.48}$$

The neural network version of the optimization problem, w.r.t. $\mathbf{z}$ and the loss function $\mathcal{L}$, is defined as the search for parameters $\theta \in [-R,R]^{p(N)}$, such that the empirical error $\Upsilon(\theta)$ is minimal. More precisely, we want to find the parameters in the set

$$\operatorname*{arg\,min}_{\theta\in[-R,R]^{p(N)}} \Upsilon(\theta) = \operatorname*{arg\,min}_{\theta\in[-R,R]^{p(N)}} \frac{1}{m}\sum_{i=1}^m \mathcal{L}(\mathcal{R}_{N,\varrho}(x_i,\theta), y_i) \tag{3.49}$$

Note that the concept of learning algorithms from Definition 2.4.7 transfers to neural network, as algorithms which seek to find parameters θ for a given neural network architecture. By far, the most used optimization methods are gradient-based, and in the field of neural networks, we often specifically use *gradient descent* algorithms. We will first introduce the deterministic version and later the stochastic version of *gradient descent*.

Definition 3.3.4 (Gradient Descent Algorithm)**.** *Let $\Gamma = (\gamma_k)_{k\in\mathbb{N}} \in (0,\infty)^{\mathbb{N}}$, let $p \in \mathbb{N}$, let $\theta^{(0)} \in \mathbb{R}^p$ and let $\Upsilon \in \mathcal{C}^1(\mathbb{R}^p, \mathbb{R})$. Then the sequence $\left(\theta^{(k)}\right)_{k\in\mathbb{N}_0}$ defined through the iteration step*

$$\theta^{(k)} = \theta^{(k-1)} - \gamma_k \nabla\Upsilon\left(\theta^{(k-1)}\right), \tag{3.50}$$

is called gradient descent sequence for Υ with initial point $\theta^{(0)}$ and step sizes Γ. We call the step size sequence $(\gamma_k)_{k\in\mathbb{N}}$ fixed, if $\gamma_k = \gamma_1$ holds for all $k \in \mathbb{N}_{\geq 2}$.

In the machine learning context, step sizes are also called learning rates.

Remark 3.3.5. *We observe that a differentiable multivariable function Υ has the property that it decreases the fastest from a point $\theta^{(k-1)}$ in the direction of the negative gradient at $\theta^{(k-1)}$, i.e., $-\nabla\Upsilon(\theta^{(k-1)})$. Assuming Υ has local minima, moving against the gradient points towards one of them. Hence, for sufficiently small step sizes Γ, the function values of the gradient descent sequence are monotonically decreasing,*

$$\Upsilon(\theta^{(k-1)}) \geq \Upsilon(\theta^{(k)}). \tag{3.51}$$

Note that the sequence of function values of the gradient descent sequence $(\Upsilon(\theta^{(k)}))_{k\in\mathbb{N}_0}$ still heavily relies on the choice of step sizes and the initial point $\theta^{(0)}$. Moreover, it may prefer a close local minimum over the global minimum for non-convex Υ (see Figure 3.3). While this presents a challenging problem for fixed step sizes, one can choose another step size sequence design to force exploration of the landscape by starting with larger step sizes. This could lead to convergence in a global minimum (cf. for instance Smith [49] and Smith [50]). Note that this design is not trivial, as the starting location can vary and highly influence the result.

Remark 3.3.6. *The gradient descent algorithm asks for $\Upsilon \in \mathcal{C}^1(\mathbb{R}^p, \mathbb{R})$. According to the assumptions made in the optimization problem for neural networks in Definition 3.3.3, the objective function satisfies $\Upsilon \in \mathcal{C}^s(\mathbb{R}^{p(N)}, \mathbb{R})$ and therefore, we will assume in the following discussions that $s \geq 1$. Moreover, in order to apply the gradient descent algorithm on the parameters of the hypothesis space of neural networks, we need to ensure that $\theta^{(K)} \in [-R, R]^{p(N)}$ holds. While the starting parameters can be chosen in this way $\theta^{(0)} \in [-R, R]^{p(N)}$, the algorithm might not provide us with the required property on the gradient descent sequence. To ensure that each $\theta^{(k)}$ in the gradient descent sequence is a valid parameter for a neural network in a hypothesis space with fixed $R > 0$, we need to project every $\theta^{(k)}$ onto the hypercube $[-R, R]^{p(N)}$. This is called the projected gradient descent (cf. Iusem [26]). A satisfactory version of the recursive definition (3.50) is therefore*

$$\theta^{(k)} = \pi_{[-R,R]^{p(N)}}\left(\theta^{(k-1)} - \gamma_k \nabla\Upsilon\left(\theta^{(k-1)}\right)\right), \tag{3.52}$$

where

$$\pi_{[-R,R]^{p(N)}} : \mathbb{R}^{p(N)} \to [-R, R]^{p(N)} \tag{3.53}$$

projects every point $x \in \mathbb{R}^{p(N)}$ to its closest elements in the hypercube $[-R, R]^{p(N)}$, which is a unique point, as the hypercube is convex in $\mathbb{R}^{p(N)}$.

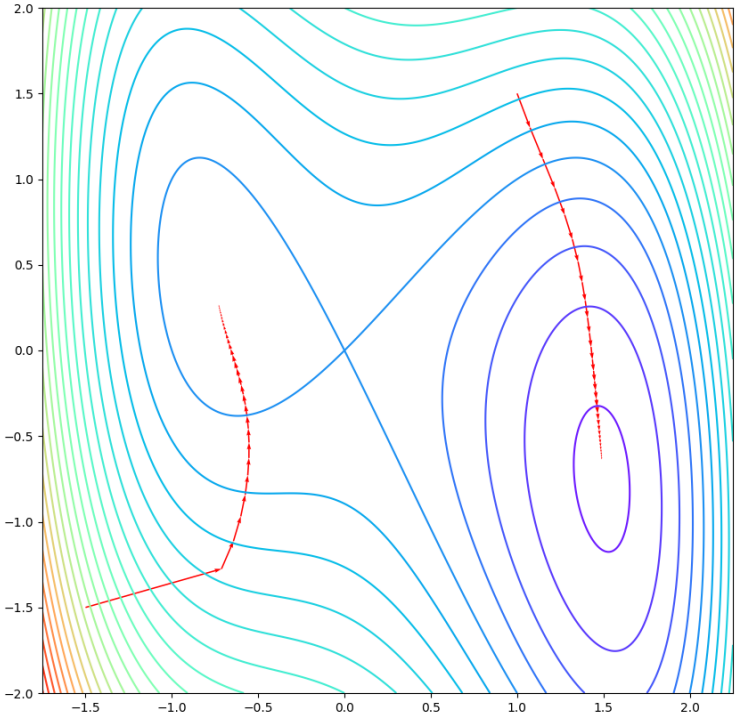


Figure 3.3: Contour lines on the subspace $\left[-\frac{7}{9}, \frac{9}{4}\right] \times [-2, 2] \subset \mathbb{R}^2$ of the function Υ, defined by $\mathbb{R}^2 \ni (\theta_1, \theta_2) \mapsto \Upsilon(\theta_1, \theta_2) := \theta_1^4 + \theta_2^2 + \theta_1\theta_2 - 2\theta_1^2 - \theta_1^3 \in \mathbb{R}$ and the first 30 steps of two gradient descent sequences (red) starting in $\left(-\frac{3}{2}, -\frac{3}{2}\right)$ and $\left(1, \frac{3}{2}\right)$ with step sizes $\gamma_k = 0.05$ for all $k \in \{1, \ldots, 20\}$. The function Υ has no maxima and two local minimum, where one is at $\left(-\frac{3}{4}, \frac{3}{8}\right)$ and the other one at $\left(\frac{3}{2}, -\frac{3}{4}\right)$, which is also the global minimum. While the gradient descent sequence starting in $\left(1, \frac{3}{2}\right)$ converges to the global minimum, the gradient descent sequence starting in $\left(-\frac{3}{2}, -\frac{3}{2}\right)$ prefers the nearest local minimum over the optimal choice. This illustrates the dependency on the initial point $\theta^{(0)}$ in the gradient descent algorithm.

The next theorem demonstrates that, under additional assumptions, the gradient descent sequence converges.

Theorem 3.3.7 (Convergence of Gradient Descent)**.** *Let $p \in \mathbb{N}$, let $K \in \mathbb{R}$ and let $\Upsilon \in \mathcal{C}^1(\mathbb{R}^p, \mathbb{R})$, such that Υ is bounded from below by K and that the gradients are*

Lipschitz-continuous with some constant $L > 0$. That means

$$\|\nabla\Upsilon(x) - \nabla\Upsilon(y)| \leq L\|x - y\| \tag{3.54}$$

holds for every $x, y \in \mathbb{R}^p$. Then, for every initial point $\theta^{(0)} \in \mathbb{R}^p$ and for every fixed step size sequence

$$\Gamma = (\gamma_k)_{k\in\mathbb{N}} \in \left(0, \frac{1}{L}\right)^{\mathbb{N}}, \tag{3.55}$$

the associated gradient descent sequence $\left(\theta^{(k)}\right)_{k\in\mathbb{N}_0}$ fulfills

$$\lim_{k\to\infty} \nabla\Upsilon(\theta^{(k)}) = 0. \tag{3.56}$$

Proof. Ruszczyński [36, Theorem 5.1]. □

Now that we established an algorithm for the optimization problem in Definition 3.3.3, we would like to apply it to a realization function of a neural network. We first note that the gradient of

$$\Upsilon(\theta) = \sum_{i=1}^{m} \nu_i(\theta) \tag{3.57}$$

requires the computation of the gradients of the maps ν_i,

$$\nabla\Upsilon(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla\nu_i(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla\mathcal{L}(\mathcal{R}_{N,\varrho}(x_i, \theta), y_i). \tag{3.58}$$

One computationally efficient and common way of computing the gradients w.r.t. the parameters $\theta \in \mathbb{R}^{p(N)}$ in the case of neural network, is the backpropagation algorithm (cf. Rumelhart [5]). In essence, backpropagation makes use of the fact that, in computing the gradients of the maps ν_i, one has to compute particular parts multiple times while recursively using the chain rule of classical calculus. To illustrate, let us consider the gradient

$$\begin{aligned} \frac{\partial\nu_i(\theta)}{\partial\theta_j} &= \frac{\partial\left(\mathcal{L}(\cdot, y_i) \circ \mathcal{R}_{N,\varrho}(x_i, \cdot)\right)(\theta)}{\partial\theta_j} \\ &= \mathcal{L}(\cdot, y_i)'\left(\mathcal{R}_{N,\varrho}(x_i, \theta)\right) \frac{\partial\mathcal{R}_{N,\varrho}(x_i, \cdot)(\theta)}{\partial\theta_j} \end{aligned} \tag{3.59}$$

and further

$$\frac{\partial\mathcal{R}_{N,\varrho}(x_i, \cdot)(\theta)}{\partial\theta_j} = \frac{\partial\left(\phi_\theta^{(L)} \circ \varrho \circ \phi_\theta^{(L-1)} \circ \cdots \circ \varrho \circ \phi_\theta^{(1)}(x_i)\right)}{\partial\theta_j}. \tag{3.60}$$

Reminding ourselves that θ_j is nothing else than some weight matrix entry $W^{(\ell)}_{k_1,k_2}$ or

some bias vector entry $b_k^{(\ell)}$ in a layer ℓ, and by applying the chain rule repeatedly, we can finally notice that all θ_j in the layers $\hat{\ell} \leq \ell$ have repeated calculations. Therefore, the most efficient way to calculate the gradients w.r.t. all θ_j is backwards through the layers while reusing the previous calculations, giving the algorithm its name. For more details, refer to source as Bishop [30, Section 5] or Rumelhart [5].
Further, the asymptotic computational cost for computing the gradient $\nabla \nu_{z_i}$ with the backpropagation algorithm is going towards $\mathcal{O}(p(N))$ floating point operations (cf. Bishop [30]). As we need to compute the gradient of ν_i for all $i \in \{1, \dots, m\}$ in order to compute the gradient $\nabla \Upsilon$, using backpropagation in the gradient-descent algorithm results in a computational complexity tending towards $\mathcal{O}(mp(N))$ floating point operations for each single gradient step. We remind ourselves that we aim towards a small error and for a fixed architecture of neural networks, increasing the amount of samples would decrease the former, as the approximation error is not effected (cf. Proposition 2.4.1). However, as we have now established, under the assumption of using the backpropagation algorithm, the computational complexity of the gradient descent algorithm scales asymptotically linearly with m, which hints to the idea of reducing the number of samples to some $\beta \in \mathbb{N}_{\leq m}$ in each gradient descent step. We refer to these smaller amounts of samples, as mini-batches, which are drawn uniformly at random from all m samples. The resulting algorithm is referred to as (mini-batch) stochastic gradient descent, which we will now define for neural networks.

Definition 3.3.8 ((Mini-Batch) Stochastic Gradient Descent Algorithm)**.** *Let $\beta \in \mathbb{N}_{\leq m}$, let $\theta^{(0)} \in \mathbb{R}^{p(N)}$, let $\Gamma = (\gamma_k)_{k \in \mathbb{N}} \in (0, \infty)^{\mathbb{N}}$, let $\left(\kappa^{(k)}\right)_{k \in \mathbb{N}}$ be independent uniformly distributed random vectors, such that for every $k \in \mathbb{N}$*

$$\kappa^{(k)} = \left(\kappa_1^{(k)}, \kappa_2^{(k)}, \dots \kappa_\beta^{(k)}\right) : \Omega \to \mathbb{N}_{\leq m}^{\beta}. \tag{3.61}$$

Then, the sequence $\left(\theta^{(k)}\right)_{k \in \mathbb{N}_0}$ defined through the iteration step

$$\theta^{(k)} = \theta^{(k-1)} - \frac{\gamma_k}{\beta} \sum_{i=1}^{\beta} \nabla_\theta \mathcal{L}\left(\mathcal{R}_{N,\varrho}\left(x_{\kappa_i^{(k)}}, \theta^{(k-1)}\right), y_{\kappa_i^{(k)}}\right), \tag{3.62}$$

is called stochastic gradient descent sequence with initial point $\theta^{(0)}$, step sizes Γ and batch size β.

Note that for $\beta = m$ and with only allowing unique draws

$$\kappa^{(k)} : \Omega \to \left\{ (i_1, i_2, \dots, i_\beta) \in \mathbb{N}_{\leq m}^{\beta} \,\middle|\, i_1 < i_2 < \dots < i_\beta \right\} \subset \mathbb{N}_{\leq m}^{\beta}, \tag{3.63}$$

the stochastic gradient descent algorithm coincides with the gradient descent algorithm. For this reason, gradient descent is sometimes also referred to as *batch gradient descent*. For any $\beta < m$, the asymptotic computational complexity for the stochastic gradient descent algorithm decreases to $\mathcal{O}(\beta p(N)) < \mathcal{O}(mp(N))$ floating point operations. Under

certain conditions, convergence of the stochastic gradient algorithm can be ensured, even though it is not deterministic and well-behaved like the original gradient descent algorithm (cf. for instance Jentzen [52], Bertsekas [23] and Shamir [40]).

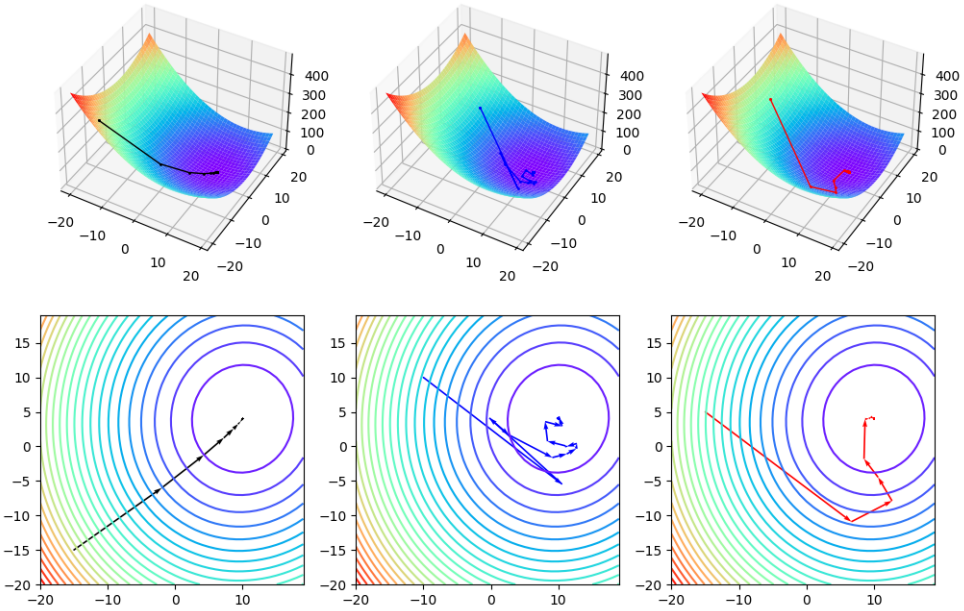


Figure 3.4: Plots of the stochastic gradient descent algorithm with different mini-batch sizes, from left to right $\beta = m = 1000$, $\beta = 1$ and $\beta = 4$, for an objective function with one local and global minimum. With increasing size of the mini-batch, the stochastic gradient descent sequence decreases in variance and fluctuations.

Moreover, the gradient step in (3.62) uses an unbiased estimator of the gradient $\nabla\Upsilon$, as

$$\begin{aligned} &\mathbb{E}\left[\frac{1}{\beta}\sum_{i=1}^{\beta}\nabla_\theta\mathcal{L}\left(\mathcal{R}_{N,\varrho}\left(x_{\kappa_i^{(k)}},\theta^{(k-1)}\right),y_{\kappa_i^{(k)}}\right)\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\nabla\nu_i\left(\theta^{(k-1)}\right) \\ &= \nabla\Upsilon(\theta^{(k-1)}) \end{aligned} \tag{3.64}$$

This inspires the idea of reformulating the stochastic gradient step in (3.62) to a more general approach. Towards a broader definition, let $\left(G^{(k)}\right)_{k\in\mathbb{N}}$ be a sequence of random

vectors, such that

$$\mathbb{E}[G^{(k)}|\theta^{(k-1)}] = \nabla\Upsilon(\theta^{(k-1)}). \tag{3.65}$$

Then we state the recursive definition of the stochastic gradient descent sequence in (3.62) in its general form,

$$\theta^{(k)} = \theta^{(k-1)} - \gamma_k G^{(k)}. \tag{3.66}$$

While the average over a mini-batch of gradients $\nabla\nu_i$ is an unbiased estimator of the gradient $\nabla\Upsilon$, the variance in the estimates goes up with decreasing computational cost, which leads to fluctuations in the stochastic gradient descent sequence (cf. Figure 3.4).

Remark 3.3.9 (Gradient Descent Improvements)**.** *In addition to the classical gradient descent algorithm and the stochastic version, there exist numerous improved methods that aim for more optimal ways of convergence. Some of the noteworthy ideas include letting the sequence generate momentum (cf. Quian [21]) and the Adam optimizer (cf. Kingma [42]). We refrain from going into more detail on any of these methods and and refer interested readers to Goodfellow [41, Chapter 8] and Ruder [48] for an overview.*

3.4 Asymptotic Sample Behavior

In chapter 2, we already established some basic understandings of the relationship between the number of samples and the errors on a given hypothesis space $\mathcal{H}$. In this chapter, we focus specifically on neural networks for solving the minimization problem and as hypothesis spaces, as introduced in the last section. Hence, we aim to understand the errors in more detail under the selection of $\mathcal{H} = \mathcal{N}_{N,\varrho,R,q}(\mathcal{D}, \mathbb{R}^n)$. So far, we only developed bounds and results for errors in form of the expected value of the quadrature loss. While this might seem sufficient, we can encounter many real-world situations, which require stronger forms of accuracy measures. For example, in the security field (cf. for instance Eykholt [51]), no only good average performance is desired, but in each single data input, hence in the uniform norm. To gain insights in the relationship between the amount of samples needed and a sufficiently good universal accuracy, we are discussing in this section the results of the work by Berner [68]. The paper specifically focuses on the hypothesis spaces

$$\mathcal{H} = \mathcal{N}_{N,\varrho_R,R,q}\left([0,1]^d, \mathbb{R}\right) \tag{3.67}$$

as defined in Definition 3.3.1, where ϱ_R denotes the ReLU activation function

$$\mathbb{R} \ni x \mapsto \varrho_R(x) := \max\{0, x\} \in \mathbb{R}. \tag{3.68}$$

Conversely, unlike the optimization problems observed before, we will not seek a function $f \in \mathcal{H}$ or, in the sense of the hypothesis space of neural networks as in Definition 3.3.3, for parameters $\theta \in \mathbb{R}^{p(N)}$ with $\|\theta\|_{\ell^q} \leq R$ which minimizes a given empirical error w.r.t. realizations of random samples $\mathbf{z}$, but we are able to deterministically chose $\mathbf{z}$. More precisely, we no longer aim to find an empirical target function $\hat{f}_{\mathcal{H},\mathbf{z}}$ that minimizes the

empirical error function, but we are given a target function u, to be reconstructed from some chosen $\mathbf{z} = (x_i, u(x_i))_{i=1}^m$. The considered target classes are $U \subset \mathcal{C}\left([0,1]^d\right)$, which contain a copy of the hypothesis space

$$u_0 + c_0\mathcal{H} \subset U \tag{3.69}$$

for some $u_0 \in U$ and constant $c_0 > 0$.
In an attempt to guarantee that the bounds conform to uniform accuracy over all optimization ideas, no specific optimization algorithms will be assumed, but the bounds will be shown for the theoretically best algorithm with the capability to sample deterministically in an adaptive fashion. This means that we will not be equipped with samples, but the algorithm is able to generate them according to its own needs. We will begin by introducing these kinds of algorithms with the new error concept and then move on to state and prove the bounds on the amount of samples needed for uniform accuracy.

Definition 3.4.1 (Adaptive Deterministic Methods)**.** *Let $d, m \in \mathbb{N}$ and let Y be a Banach space, then for any given $U \subset \mathcal{C}([0,1]^d) \cap Y$, we call a map $\tilde{\mathcal{A}} : U \to Y$ an adaptive deterministic method using m point samples, if there exist a sequence of mappings $(\alpha_i)_{i=1}^m$ with*

$$\alpha_1 \in [0,1]^d \quad \text{and} \quad \alpha_i : \left([0,1]^d\right)^{i-1} \times \mathbb{R}^{i-1} \to [0,1]^d \quad \text{for } i = 2, \dots, m \tag{3.70}$$

and another mapping

$$Q : \left([0,1]^d\right)^m \times \mathbb{R}^m \to Y, \tag{3.71}$$

such that for every $u \in U$, with the point sequence $\mathbf{x} = (x_1, \dots . x_m) \subset \left([0,1]^d\right)^m$ generated by $(\alpha_i)_{i=1}^m$ as

$$x_1 = \alpha_1 \quad \text{and} \quad x_i = \alpha_i(x_1, \dots, x_{i-1}, u(x_1), \dots, u(x_{i-1})) \quad \text{for } i = 2, \dots, m, \tag{3.72}$$

the map $\tilde{\mathcal{A}}$ follows the scheme

$$\tilde{\mathcal{A}}(u) = Q(x_1, \dots, x_m, u(x_1), \dots, u(x_m)) \in Y. \tag{3.73}$$

We denote by $Alg_m(U, Y)$ the set of all deterministic methods using m point samples.

Next we will see a randomized version of the algorithms in the definition above.

Definition 3.4.2 (Adaptive Random Methods)**.** *Let $d, m \in \mathbb{N}$ and let Y be a Banach space, then for any given $U \subset \mathcal{C}([0,1]^d) \cap Y$ and some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, we call a tuple $(\mathbf{A}, \mathbf{m})$ adaptive random method using m point samples on average, if $\mathbf{A} = (\tilde{\mathcal{A}}_\omega)_{\omega \in \Omega}$ is a sequence over Ω and $\mathbf{m} : \Omega \to \mathbb{N}$ is a mapping, such that the following conditions apply to them:*

1. *$\mathbf{m}$ is measurable, and $\mathbb{E}[\mathbf{m}] \leq m$,*

2. $\forall u \in U : \omega \mapsto \tilde{\mathcal{A}}_\omega(u)$ *is a measurable map with respect to the Borel* σ*-algebra on* Y*, and*

3. $\forall \omega \in \Omega : \tilde{\mathcal{A}}_\omega \in Alg_{\mathbf{m}(\omega)}(U, Y)$.

We denote by $Alg_m^{MC}(U, Y)$ *the set of all random methods using* m *point samples on average.*

The described methods in Definition 3.4.2 are sometimes also referred to as *Monte-Carlo algorithms*. Furthermore, note that $Alg_m(U, Y) \subset Alg_m^{MC}(U, Y)$ holds, as any deterministic sampling method can be reinterpreted as a randomized method over a trivial probability space.

Definition 3.4.3 (Optimal (Randomized) Error)**.** *Let* $d, m \in \mathbb{N}$ *and let* Y *be a Banach space, then for any given* $U \subset \mathcal{C}([0,1]^d) \cap Y$*, we define the optimal (randomized) error as*

$$err_m^{MC}(U, Y) := \inf_{(\mathbf{A},\mathbf{m}) \in Alg_m^{MC}(U,Y)} \sup_{u \in U} \mathbb{E}[\|u - \tilde{\mathcal{A}}_\omega(u)\|_Y]. \tag{3.74}$$

The optimal randomized error will be our measure of accuracy when attempting to reconstruct target functions from m point samples.

Remark 3.4.4. *In contrast to the previous definition of algorithms in Definition 2.4.8 and its resulting error* $\mathcal{E}_Z(\mathcal{A}(\mathbf{z}))$*, does not depend on a choice of method or some* $\mathbf{z}$*, but instead considers the optimal algorithms with the optimal choices of samples.*

Definition 3.4.5 (Copy of Hypothesis Space)**.** *Let* $d \in \mathbb{N}$ *and let* $U, \mathcal{H} \subset \mathcal{C}\left([0,1]^d\right)$*. If there exists* $u_0 \in U$ *and* $c_0 > 0$*, such that*

$$u_0 + c_0 \mathcal{H} \subset U \tag{3.75}$$

holds, we say that U *contains a copy of* $\mathcal{H}$ *(attached to* u_0 *with constant* c_0*).*

Next, we will state the lower bound found on the established error in the special case of $Y = L^\infty\left([0,1]^d\right)$ and the necessary conditions for the behavior of the number of samples m required to achieve a certain level of uniform accuracy.

Theorem 3.4.6 (Lower Bound on Uniform Accuracy)**.** *Let* $L \in \mathbb{N}_{\geq 3}$*, let* $d \in \mathbb{N}$*, let* $q \in [1, \infty]$ *and let* $R > 0$*. Let* $U \subset \mathcal{C}\left([0,1]^d\right)$ *be a target class, which contains a copy of* $\mathcal{N}_{N,\varrho_R,R,q}\left([0,1]^d, \mathbb{R}\right)$ *with some constant* $c_0 > 0$*, where* $N_\ell = 3d$ *fixed for every* $\ell \in \{1, \ldots, L-1\}$*. Then, with*

$$\Omega_{low} = \begin{cases} \frac{1}{4 \cdot 3^{2/q}} \cdot R^L \cdot d^{1-\frac{2}{q}} & \text{if } q \leq 2 \\ \frac{1}{24} \cdot R^L \cdot \left((3d)^{1-\frac{2}{q}}\right)^{L-1} & \text{if } q \geq 2, \end{cases} \tag{3.76}$$

it holds that

$$err_m^{MC}(U, L^\infty([0,1]^d)) \geq c_0 \cdot \frac{\Omega_{low}}{64d} \cdot m^{-\frac{1}{d}}. \tag{3.77}$$

A more general case will be stated in Theorem 3.4.8 and proven afterwards.

Remark 3.4.7 (Lower Sample Bound for Uniform Accuracy)**.** *The lower bound of the optimal error in (3.77) can be rewritten to a lower bound on the samples needed to reach accuracy $\epsilon \geq 0$. Assume that there exists an $\epsilon \geq 0$ such that $err_m^{MC}(U, L^\infty([0,1]^d)) \leq \epsilon$. Then, with Ω_{low} as in (3.76), it holds that*

$$\begin{aligned} m &\geq c_0^d \cdot \left(\frac{\Omega_{low}}{64d}\right)^d \cdot err_m^{MC}(U, L^\infty([0,1]^d))^{-d} \\ &\geq c_0^d \cdot \left(\frac{\Omega_{low}}{64d}\right)^d \cdot \epsilon^{-d}. \end{aligned} \tag{3.78}$$

If we set $\epsilon = \frac{1}{1024}$, we can even give a more tangible estimate,

$$m \geq 2^d \cdot R^{dL} \cdot (3d)^{d(L-2)}. \tag{3.79}$$

Here, the required number of samples grows exponentially w.r.t. the input dimension d.

Theorem 3.4.8 (Lower Bound on Optimal Error)**.** *Let $L \in \mathbb{N}_{\geq 3}$, let $d, J \in \mathbb{N}$, let $p, q \in [1, \infty]$ and let $R > 0$. Let $U \subset \mathcal{C}\left([0,1]^d\right)$ be a target class, which contains a copy of $\mathcal{N}_{N,\varrho_R,R,q}\left([0,1]^d, \mathbb{R}\right)$ with some constant $c_0 > 0$, where $N_\ell = J$ fixed for every $\ell \in \{1, \ldots, L-1\}$. Then, for any $s \in \mathbb{N}$ with $s \leq \min\left\{\frac{J}{3}, d\right\}$ and with*

$$\Omega_{low} = \begin{cases} \frac{1}{4 \cdot 3^{2/q}} \cdot R^L \cdot s^{1-\frac{2}{q}} & \text{if } q \leq 2 \\ \frac{1}{24} \cdot R^L \cdot \left((J^{1-\frac{2}{q}}\right)^{L-1} & \text{if } q \geq 2, \end{cases} \tag{3.80}$$

it holds that

$$err_m^{MC}(U, L^p([0,1]^d)) \geq c_0 \cdot \frac{\Omega_{low}}{(64s)^{1+\frac{s}{p}}} \cdot m^{-\frac{1}{p}-\frac{1}{s}}. \tag{3.81}$$

By taking $p = \infty$ and $J = 3d$, Theorem 3.4.8 directly implies Theorem 3.4.6. To show that Theorem 3.4.8 holds, we use a lemma and another theorem. Lemma 3.4.10 is implying, that each hypothesis space $\mathcal{N}_{(d,J\ldots,J,1),\varrho_R,R,q}([0,1]^d, \mathbb{R})$ contains a large class of hat functions (cf. Definition 3.4.9). The second statement, Theorem 3.4.11, shows that under the condition of the class U containing a large class of hat functions, the lower bound stated in Theorem 3.4.8 holds.

Definition 3.4.9 (Hat Functions)**.** *Let $d \in \mathbb{N}$, let $B > 0$, let $\sigma \in \mathbb{R}$ and let $y \in \mathbb{R}^d$. Then we define a hat function with sharpness B and peak σ by*

$$\Lambda_{B,\sigma} : \mathbb{R} \to (-\infty, 1], \quad t \to \begin{cases} 0 & \text{if } t \leq \sigma - \frac{1}{B} \\ 1 - B \cdot |t - \sigma| & \text{if } t \geq \sigma - \frac{1}{B}. \end{cases} \tag{3.82}$$

Further, let $s \in \{1, \dots, d\}$, then we define

$$\Lambda_{B,y}^{(s)} : \mathbb{R}^d \to (-\infty, 1], \quad x \mapsto \left(\sum_{i=1}^{s} \Lambda_{B,y_i}(x_i) \right) - (s-1) \tag{3.83}$$

and

$$\zeta_{B,y}^{(s)} : \mathbb{R}^d \to [0,1], \quad x \mapsto \varrho_R(\Lambda_{B,y}^{(s)}(x)). \tag{3.84}$$

Lemma 3.4.10. *Let $L, J \in \mathbb{N}_{\geq 3}$, let $d \in \mathbb{N}$, let $R > 0$ and let $s \in \mathbb{N}$ with $s \leq \min\{d, \frac{J}{3}\}$. Then, there exist for every $q \in [1, \infty]$ a constant $\lambda_q \in \mathbb{R}$ such that for every $B \in \mathbb{N}$, every $\nu \in \{\pm 1\}$ and every $y \in [0,1]^d$, we have*

$$\nu \cdot \frac{\lambda_q}{Bs} \cdot \zeta_{B,y}^{(s)} \in \mathcal{N}_{N,\varrho_R,R,q}([0,1]^d, \mathbb{R}), \tag{3.85}$$

where $N_\ell = J$ for $\ell \in \{2, \dots, L-1\}$.

Proof. Berner [68, Lemma 2.4 and Lemma 2.5]. □

Theorem 3.4.11. *Let $d, m \in \mathbb{N}$, let $s \in \{1, \dots, d\}$ and let $U \subset \mathcal{C}([0,1]^d)$. If for every $\nu \in \{\pm 1\}$ and every $y \in [0,1]^d$, there exists a certain $\lambda > 0$ and $u_0 \in \mathcal{C}([0,1]^d)$ such that*

$$u_0 + \nu \cdot \frac{\lambda}{Bs} \zeta_{B,y}^{(s)} \in U \tag{3.86}$$

for $B = 8 \lceil m^{1/s} \rceil$, then it holds for every $p \in [1, \infty]$ that

$$err_m^{MC}(U, L^p([0,1]^d)) \geq \frac{\lambda/2}{(64s)^{1+s/p}} \cdot m^{-\frac{1}{p}-\frac{1}{s}}. \tag{3.87}$$

Proof. We begin by setting $k := \lceil m^{1/s} \rceil$ and define the sequence $(y^\ell)_{\ell \in \{1,\dots,4k\}^d}$ with $y^\ell := \frac{(1,\dots,1)}{8k} - \frac{\ell - (1,\dots,1)}{4k} \in [0,1]^d$. Let $H_m := (\{1, \dots, 4k\}^s \times \{0\}) \times \{\pm 1\} \subset \mathbb{Z}^d \times \{\pm 1\}$, then it holds by assumption that

$$\alpha_{\ell,\nu} := u_0 + \nu \cdot \frac{\lambda}{Bs} \cdot \zeta_{B,y^\ell}^{(s)} \in U \tag{3.88}$$

for all $(\ell, \nu) \in H_m$. By Berner [68, Lemma 2.3] it holds that

$$\operatorname{supp}(\alpha_{\ell,\nu} - u_0) = \operatorname{supp} \zeta_{B,y^\ell}^{(s)} \subset y^\ell + (B^{-1}[-1,1]^s \times \mathbb{R}^{d-s}) \tag{3.89}$$

and since for $\ell \neq \tilde{\ell}$ we have $\|y^\ell - y^{\tilde{\ell}}\| \geq \frac{1}{4k} > \frac{1}{8k} = B^{-1}$, we can state

$$\forall (\ell, \nu), (\tilde{\ell}, \tilde{\nu}) \in H_m : \ell \neq \tilde{\ell} \implies \operatorname{supp}(\alpha_{\ell,\nu} - u_0)^\circ \cap \operatorname{supp}(\alpha_{\tilde{\ell},\tilde{\nu}} - u_0)^\circ = \emptyset. \tag{3.90}$$

Let now $\tilde{\mathcal{A}} \in Alg_{2m}(U, L^p[0,1]^d)$ be arbitrary and its according point sequence as con-

structed in (3.72), $\mathbf{x} = (x_1, \dots, x_{2m}) \in ([0,1]^d)^{2m}$. We define

$$I_{\mathbf{x}} := \left\{ \ell \in \{1, \dots, 4k\}^s \times \{0\} : \forall i \in \{1, \dots, 2m\} : \zeta^{(s)}_{B,y^\ell}(x_i) = 0 \right\} \subset \mathbb{Z}^d \tag{3.91}$$

and want to show that $|I_{\mathbf{x}}| \geq (4k)^s - 2m$ holds. By definition $I_{\mathbf{x}} \subset (\{1, \dots, 4k\}^s \times \{0\})$, and so instead of estimating the cardinality directly, we will consider the complement set $I^c_{\mathbf{x}} := (\{1, \dots, 4k\}^s \times \{0\}) \backslash I_{\mathbf{x}}$. Let $\ell \in I^c_{\mathbf{x}}$, then there must exist some $i_\ell \in \{1, \dots, 2m\}$, such that $\zeta^{(s)}_{B,y^\ell}(x_{i_\ell}) \neq 0$, which implies $x_{i_\ell} \in (\operatorname{supp} \zeta^{(s)}_{B,y^\ell})^\circ$. Hence, with the former result in (3.90), the map $I^c_{\mathbf{x}} \ni \ell \mapsto i_\ell \in \{1, \dots, 2m\}$ is injective and therefore $|I^c_{\mathbf{x}}| \leq 2m$, which shows that $|I_{\mathbf{x}}| \geq (4k)^s - 2m$. Combining the earlier definition of the sequence $\alpha_{\ell,\nu}$ from (3.88) with the definition of $I_{\mathbf{x}}$ and the conditions on $\tilde{\mathcal{A}}$ lets us further state

$$\forall (\ell, \nu) \in H_m : \ell \in I_{\mathbf{x}} \implies \tilde{\mathcal{A}}(\alpha_{\ell,\nu}) = \tilde{\mathcal{A}}(u_0). \tag{3.92}$$

Next, by noting that

$$\frac{|I_{\mathbf{x}}|}{(4k)^s} \geq 1 - \frac{2m}{(4k)^s} = 1 - \frac{2m}{(4\lceil m^{1/s} \rceil)^s} \geq \frac{1}{2}, \tag{3.93}$$

and with using previous results and definitions, it holds that

$$\begin{aligned}
&\frac{1}{|H_m|} \sum_{(\ell,\nu) \in H_m} \|\alpha_{\ell,\nu} - \tilde{\mathcal{A}}(\alpha_{\ell,\nu})\|_{L^p([0,1]^d)} \\
&= \frac{1}{(4k)^s} \sum_{\ell \in \{1,\dots,4k\}^s \times \{0\}} \left(\frac{1}{2} \|\alpha_{\ell,-1} - \tilde{\mathcal{A}}(\alpha_{\ell,-1})\|_{L^p([0,1]^d)} + \frac{1}{2} \|\alpha_{\ell,1} - \tilde{\mathcal{A}}(\alpha_{\ell,1})\|_{L^p([0,1]^d)} \right) \\
&\geq \frac{1}{(4k)^s} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|\alpha_{\ell,-1} - \tilde{\mathcal{A}}(\alpha_{\ell,-1})\|_{L^p([0,1]^d)} + \frac{1}{2} \|\alpha_{\ell,1} - \tilde{\mathcal{A}}(\alpha_{\ell,1})\|_{L^p([0,1]^d)} \right) \\
&\geq \frac{1}{2} \cdot \frac{1}{|I_{\mathbf{x}}|} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|\alpha_{\ell,-1} - \tilde{\mathcal{A}}(\alpha_{\ell,-1})\|_{L^p([0,1]^d)} + \frac{1}{2} \|\alpha_{\ell,1} - \tilde{\mathcal{A}}(\alpha_{\ell,1})\|_{L^p([0,1]^d)} \right) \\
&\geq \frac{1}{2} \cdot \frac{1}{|I_{\mathbf{x}}|} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|\alpha_{\ell,-1} - \tilde{\mathcal{A}}(u_0)\|_{L^p([0,1]^d)} + \frac{1}{2} \|\alpha_{\ell,1} - \tilde{\mathcal{A}}(u_0)\|_{L^p([0,1]^d)} \right) \\
&\geq \frac{1}{2} \cdot \frac{1}{|I_{\mathbf{x}}|} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|\alpha_{\ell,-1} - \alpha_{\ell,1}\|_{L^p([0,1]^d)} \right) \\
&= \frac{1}{2} \cdot \frac{1}{|I_{\mathbf{x}}|} \sum_{\ell \in I_{\mathbf{x}}} \| \frac{\lambda}{Bs} \cdot \zeta^{(s)}_{B,y^\ell} \|_{L^p([0,1]^d)} \\
&\geq \lambda \cdot (4s)^{-1-\frac{s}{p}} \cdot B^{-1-\frac{s}{p}} \\
&\geq \lambda \cdot (4s)^{-1-\frac{s}{p}} \cdot 16^{-1-\frac{s}{p}} \cdot m^{-\frac{1}{s}-\frac{1}{p}} = \frac{\lambda}{(64s)^{1+\frac{s}{p}}} \cdot m^{-\frac{1}{s}-\frac{1}{p}},
\end{aligned} \tag{3.94}$$

where we again used results from Berner [68, Lemma 2.3] in the before last line and in the last line the fact that $B = 8k \leq 8m^{1/s} + 8 \leq 16m^{1/s}$.
We will now move on to show the last step before we can conclude the proof. For that, let $(\Omega, \mathcal{F}, \mathbb{P})$ be some probability space, and let $(\mathbf{A}, \mathbf{m}) \in Alg_m^{MC}(U, L^p([0,1]^d)$ be arbitrary, with $\mathbf{A} = (\tilde{\mathcal{A}}_\omega)_{\omega\in\Omega}$ being a sequence over Ω. Define $\Omega_{\mathbf{m}} := \{\omega \in \Omega | \mathbf{m}(\omega) \leq 2m\}$, then by the Markov inequality, we have

$$m \geq \mathbb{E}[\mathbf{m}] \geq 2m \cdot \mathbb{P}(\Omega_{\mathbf{m}}^c) \tag{3.95}$$

and therefore,

$$\mathbb{P}(\Omega_{\mathbf{m}}) = 1 - (\Omega_{\mathbf{m}}^c) \geq \frac{1}{2}. \tag{3.96}$$

To finish the proof, we note that we chose $(\mathbf{A}, \mathbf{m}) \in Alg_m^{MC}(U, L^p([0,1]^d)$ arbitrary and hence to show the estimate in (3.87) it is enough to compute

$$\begin{aligned}
&\sup_{u\in U} \mathbb{E}[\|u - \tilde{A}_\omega(u)\|_{L^p([0,1]^d)}] \\
&\geq \frac{1}{|H_m|} \sum_{(\ell,\nu)\in H_m} \mathbb{E}[\|\alpha_{\ell,\nu} - \tilde{A}_\omega(\alpha_{\ell,\nu})\|_{L^p([0,1]^d)}] \\
&= \mathbb{E}\left[\frac{1}{|H_m|} \sum_{(\ell,\nu)\in H_m} \|\alpha_{\ell,\nu} - \tilde{A}_\omega(\alpha_{\ell,\nu})\|_{L^p([0,1]^d)}\right] \\
&\geq \mathbb{E}\left[\mathbb{1}_{\Omega_{\mathbf{m}}}(\omega) \cdot \frac{1}{|H_m|} \sum_{(\ell,\nu)\in H_m} \|\alpha_{\ell,\nu} - \tilde{A}_\omega(\alpha_{\ell,\nu})\|_{L^p([0,1]^d)}\right] \\
&\geq \mathbb{P}(\Omega_{\mathbf{m}}) \cdot \frac{\lambda}{(64s)^{1+\frac{s}{p}}} \cdot m^{-\frac{1}{s}-\frac{1}{p}} \\
&\geq \frac{\lambda/2}{(64s)^{1+\frac{s}{p}}} \cdot m^{-\frac{1}{p}-\frac{1}{s}},
\end{aligned} \tag{3.97}$$

where the first estimate follows from $\alpha_{\ell,\nu} \subset U$ and in the before last line (3.94) was applied.

□

Proof of Theorem 3.4.8. Follows directly from Theorem 3.4.11 with Lemma 3.4.10. □

Finally, we will state the upper bound.

Theorem 3.4.12 (Upper Bound on Uniform Accuracy)**.** *Let $d, L \in \mathbb{N}$, let $q \in [1, \infty]$, let $R > 0$ and let $N_1, \ldots, N_{L-1} \in \mathbb{N}$. Then, with*

$$\Omega_{up} = \begin{cases} 2\sqrt{d} \cdot R^L & \text{if } q \leq 2 \\ 2\sqrt{d} \cdot R^L \cdot (\sqrt{d} \cdot N_1 \cdots N_{L-1})^{1-\frac{2}{q}} & \text{if } q \geq 2, \end{cases} \tag{3.98}$$

it holds that

$$\begin{aligned} &err_m^{MC}\left(\mathcal{N}_{(d,N_1\dots,N_{L-1},1),\varrho_R,R,q}([0,1]^d,\mathbb{R}),L^\infty([0,1]^d)\right)\\ &\leq \Omega_{up}\cdot m^{-\frac{1}{d}}. \end{aligned} \tag{3.99}$$

We will proof Theorem 3.4.12 with the help of two lemmas. Lemma 3.4.15 will imply that each neural network is Lipschitz continuous and the Lipschitz constant (Definition 3.4.14) can be appropriately bounded with the hyper-parameters and numbers of neurons N. The second lemma states that functions satisfying this property can be reconstructed by piece-wise constant interpolation from samples.

Remark 3.4.13 (Upper Sample Bound for Uniform Accuracy)**.** *As for the lower bound, we can rewrite the upper bound towards an upper bound on the needed samples. To do so, we set* $U = \mathcal{N}_{N,\varrho_R,R,q}([0,1]^d,\mathbb{R})$*. Then, with* Ω_{up} *as in (3.98), it holds that*

$$m \leq \Omega_{up}^d \cdot err_m^{MC}(U, L^\infty([0,1]^d))^{-d}. \tag{3.100}$$

Definition 3.4.14 (Lipschitz Constant)**.** *Let* $d,k\in\mathbb{N}$ *and let* $q\in[1,\infty]$*, then we define the Lipschitz constant for any* $f:\mathbb{R}^d\to\mathbb{R}^k$ *w.r.t. the* ℓ^q*-norm by*

$$\mathrm{Lip}_{\ell^q}(f) := \sup_{x,y\in\mathbb{R}^d, x\neq y} \frac{\|f(x)-f(y)\|_{\ell^q}}{\|x-y\|_{\ell^q}}. \tag{3.101}$$

Lemma 3.4.15. *Let* $d,L\in\mathbb{N}$*, let* $N=(N_0,\dots,N_L)\in\mathbb{N}^{L+1}$*, where* $N_0=d$ *and* $N_L=1$ *fixed, let* $q\in[1,\infty]$ *and let* $R>0$*. Then, the Lipschitz constant of any realization function* $\mathcal{R}_{N,\varrho_R}(\cdot,\theta)\in\mathcal{N}_{N,\varrho_R,R,q}([0,1]^d,\mathbb{R}),L^\infty([0,1]^d)$ *w.r.t. the* ℓ^2*-norm, can be estimated by*

$$\mathrm{Lip}_{\ell^2}(\mathcal{R}_{N,\varrho_R}(\cdot,\theta)) \leq \begin{cases} R^L & \text{if } q\leq 2\\ R^L\cdot(\sqrt{N_0N_L}\cdot N_1\cdots N_{L-1})^{1-2/q} & \text{if } q\geq 2. \end{cases} \tag{3.102}$$

Proof. Let $\mathcal{R}_{N,\varrho_R}(\cdot,\theta)\in\mathcal{N}_{N,\varrho_R,R,q}([0,1]^d,\mathbb{R}),L^\infty([0,1]^d)$ be arbitrary. As already noted in (3.34), with

$$\phi_\theta^{(\ell)}:\mathbb{R}^{N_{\ell-1}}\to\mathbb{R}^{N_\ell},\quad x\mapsto W^{(\ell)}x+b^\ell, \tag{3.103}$$

where $((W^{(\ell)},b^{(\ell)}))_{\ell=1}^L=\theta\in\mathbb{R}^{p(N)}$, and with assuming ϱ_R to act component-wise, we can write $\mathcal{R}_{N,\varrho_R}(\cdot,\theta)$ as

$$\mathcal{R}_{N,\varrho_R}(\cdot,\theta)=\phi_\theta^{(L)}\circ\varrho\circ\phi_\theta^{(L-1)}\circ\cdots\circ\varrho\circ\phi_\theta^{(1)}. \tag{3.104}$$

By Berner [68, Lemma B.1] and the fact that $\|W^{(\ell)}\|_{\ell^q}\leq R$, it holds that

$$\mathrm{Lip}_{\ell^2}(\phi_\theta^{(\ell)}) \leq \begin{cases} R & \text{if } q\leq 2\\ R\cdot(\sqrt{N_{i-1}N_i})^{1-2/q} & \text{if } q\geq 2. \end{cases} \tag{3.105}$$

Further, note that for all $x, y \in \mathbb{R}$ the ReLU function satisfies $|\varrho_R(x) - \varrho_R(y)| \leq |x - y|$ and therefore also

$$\|\varrho_R(x) - \varrho_R(y)\|_{\ell^2} \leq \|x - y\|_{\ell^2} \tag{3.106}$$

for every $x, y \in \mathbb{R}^B$, for every $B \in \mathbb{N}$. Finally, by noting that for any appropriate sequence of functions $(f_i)_{i=1}^k$

$$\mathrm{Lip}_{\ell^2}(f_k \circ \cdots \circ f_1) \leq \prod_{i=1}^{k} \mathrm{Lip}_{\ell^2}(f_i) \tag{3.107}$$

holds, we can conclude and finish the proof by combining the observations.

□

Lemma 3.4.16. *Let $d \in \mathbb{N}$. Then, for any $m \in \mathbb{N}$, there exist points $x_1, \ldots, x_m \in [0,1]^d$ and a map $T_m : \mathbb{R}^m \to L^\infty([0,1]^d)$, such that for every map $u : [0,1]^d \to \mathbb{R}$ with $\mathrm{Lip}_{\ell^2}(u) < \infty$, it holds that*

$$\|T_m\left(u(x_1), \ldots, u(x_m)\right) - u\|_{L^\infty([0,1]^d)} \leq \mathrm{Lip}_{\ell^2}(u) \cdot 2\sqrt{d} \cdot m^{-\frac{1}{d}}. \tag{3.108}$$

Proof. Berner [68, Lemma 2.9]. □

Proof of Theorem 3.4.12. Follows directly from combining Lemma 3.4.15 and Lemma 3.4.16. □

Remark 3.4.17. *Even that the ReLU activation function is one of the most used in practise and therefore worthwhile studying, one might want to extend the results from Theorem 3.4.6 and 3.4.12 to other options. However, due to the well-behaved properties of ReLU, other activation functions might pose more complex settings and need for additional research.*
Furthermore, previously we considered the error with regards to the quadrature loss. While the work discussed in this section is rather new, there are a multitude of previous efforts to find bounds on the errors w.r.t. the quadrature loss for regression and also classification tasks. Since the variety of different approaches is vast, we will only mention further readings from Anthony [19] and Blum [9].

The presented results make it clear that it is worthwhile to measure not only the average performance but also to consider other errors, such as the maximum error.

3.5 Stability and Regularization Methods

In the final part of this chapter, we will discuss the concept of stability in neural networks training and present a strategy to enhance it. Here, stability refers to the ability of a trained learning algorithm to generalize to new and unseen data. As neural networks usually have a large number of parameters, they are prone to picking up underlying noise in the data, leading to over-fitting on the training samples and poor generalization

performance. This poses the problem of finding a balance between variance and bias in the hypothesis space complexity. In contrast to the typical U-shaped error curve that shows the trade-off between bias and variance, neural networks provide tools to regulate variance without significantly altering the bias. These include modifications to the architecture or algorithm, such as drop-out (Srivastava [45]) and data augmentation (Shorten [58]), as well as regularization functions $r : \mathbb{R}^{p(N)} \to \mathbb{R}$ that modify the optimization problem to the form

$$\arg\min_{\theta \in \mathbb{R}^{p(N)}} \Upsilon(\theta) + r(\theta). \tag{3.109}$$

We will continue to assume the setting of Definition 3.3.1 throughout this section.

Definition 3.5.1 (Stability)**.** *Let $\xi : \mathbb{N} \to \mathbb{R}$ be a monotonously decreasing sequence, let $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^n \times \mathbb{R}^n, [0, \infty))$ be a loss function and let $\mathcal{A}$ be a learning algorithm on $\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n)$. We call $\mathcal{A}$ on-average-replace-one-stable with rate ξ w.r.t. $\mathcal{L}$ if, for any data Z and every number of samples $m \in \mathbb{N}$, it holds that*

$$\mathcal{S}_{\mathcal{Z}}(\mathcal{A}) := \mathbb{E}\left[\mathbb{E}_{i\sim\mathcal{U}(m)}\left[\mathcal{L}(\mathcal{A}(\mathcal{Z}^i)(X_i), Y_i)) - \mathcal{L}(\mathcal{A}(\mathcal{Z})(X_i), Y_i))\right]\right] \leq \xi(m), \tag{3.110}$$

where $\mathcal{Z} = ((X_j, Y_j))_{j=1}^m$ and $\mathcal{Z}^i = ((X_j^i, Y_j^i))_{j=1}^m$ are different m i.i.d. copies of Z such that $\mathcal{Z}_j \sim \mathcal{Z}_j^i$ for every $j \neq i$ and $\mathcal{Z}_i^i \sim Z$ independent of $\mathcal{Z}$.

To put the concept of Definition 3.5.1 into words, we might explain stability to not change in the output too much given a small change in the input. We will see now how this definition of stability is related to the generalization error.

Theorem 3.5.2. *Let $\mathcal{L} \in \mathcal{C}^1(\mathbb{R}^n \times \mathbb{R}^n, [0, \infty)$ be a loss function, let Z be some data and let $\mathcal{Z}$ and $\mathcal{Z}^i$ be different m i.i.d. copies of Z such that $\mathcal{Z}_j \sim \mathcal{Z}_j^i$ for every $j \neq i$ and $\mathcal{Z}_i^i \sim Z$ independent of $\mathcal{Z}$. Then, it holds for every learning algorithm $\mathcal{A}$ on $\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n)$, that*

$$\mathbb{E}\left[\mathcal{E}_Z^{\mathcal{L}}(\mathcal{A}(\mathcal{Z})) - \mathcal{E}_{\mathcal{Z}}^{\mathcal{L}}(\mathcal{A}(\mathcal{Z}))\right] = \mathcal{S}_{\mathcal{Z}}(\mathcal{A}), \tag{3.111}$$

where $\mathcal{E}_Z^{\mathcal{L}}$ and $\mathcal{E}_{\mathcal{Z}}^{\mathcal{L}}$ are the error and empirical error function for the loss function $\mathcal{L}$.

Proof. First, we note

$$\mathbb{E}\left[\mathcal{E}_{\mathcal{Z}}^{\mathcal{L}}(\mathcal{A}(\mathcal{Z}))\right] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^m \mathcal{L}(\mathcal{A}(\mathcal{Z})(X_i), Y_i))\right] = \mathbb{E}\left[\mathbb{E}_{i\sim\mathcal{U}(m)}\left[\mathcal{L}(\mathcal{A}(\mathcal{Z})(X_i), Y_i))\right]\right]. \tag{3.112}$$

Further, since $\mathcal{Z}_i^i = (X_i^i, Y_i^i)$ is independent of $\mathcal{Z}$, it holds that

$$\mathbb{E}\left[\mathcal{E}_Z^{\mathcal{L}}(\mathcal{A}(\mathcal{Z}))\right] = \mathbb{E}\left[\mathcal{L}(\mathcal{A}(\mathcal{Z})(X_i^i), Y_i^i))\right] = \mathbb{E}\left[\mathcal{L}(\mathcal{A}(\mathcal{Z}^i)(X_i), Y_i))\right]. \tag{3.113}$$

By the linearity of the expected value and combining the two equations we can complete the proof. □

Theorem 3.5.2 tells us that the definition of stability is equivalent with not over-fitting on a random sample, used for training with the algorithm. Now that we fixed a concept

for stability, we will introduce regularization of the loss function as a way of increasing it. We will assume the setting given in the optimization problem stated in Definition 3.3.3. Then, we will modify the objective by introducing regularization of the loss function as a way of increasing stability.

Definition 3.5.3 (Regularized Optimization Problem, Neural Network Version)**.** *Let $s \in \mathbb{N}_0$ and let $r : \mathbb{R}^{p(N)} \to \mathbb{R}$. For a given loss function $\mathcal{L} \in \mathcal{C}^s(\mathbb{R}^n \times \mathbb{R}^n, [0, \infty))$, let Υ be defined as in Definition 3.3.3. The regularized version of the optimization problem for neural networks, w.r.t. the loss function $\mathcal{L}$, $\mathbf{z}$ and regularizer r, is defined as the search for parameters $\theta \in [-R, R]^{p(N)}$, such that the average regularized loss $\Upsilon(\theta) + r(\theta)$ is minimal. More precisely, we want to find the parameters in the set*

$$\underset{\theta \in [-R,R]^{p(N)}}{\arg\min}\ \Upsilon(\theta) + r(\theta) = \underset{\theta \in [-R,R]^{p(N)}}{\arg\min} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\mathcal{R}_{N,\varrho}(x_i, \theta), y_i) + r(\theta). \tag{3.114}$$

Intuitively, a regularizer can measure the complexity of the hypothesis space, or in the case of neural networks the complexity of the parameters θ. Therefore, the regularized version of the optimization problem seeks for minimizing the empirical error Υ, while keeping the hypothesis spaces complexity low, acting as a balance towards stability (cf. Remark 2.4.10 on bias-variance trade-off). It should come to no surprise that a common choice is

$$r_\lambda(\theta) := \lambda \|\theta\|_{\ell_1} \tag{3.115}$$

for some $\lambda > 0$, which is called the *L1-regularizer*. Since we consider bounded parameters $\theta \in [-R, R]^{p(N)}$, we already have through the choice of R a regularization option, which can be made more tight by optimizing the newly posed objective function with sufficient regularizer. If we assume that $s \geq 1$ and that $r \in \mathcal{C}^1(\mathbb{R}^{p(N)}, \mathbb{R})$ holds, we can still apply the same learning algorithms as before, since only the objective function has changed. Another commonly used regularizer is the so called *L2-regularizer* or *Tikhonov regularizer*, which, for some $\lambda > 0$ is defined by

$$\tilde{r}_\lambda(\theta) := \lambda \|\theta\|_{\ell_2}. \tag{3.116}$$

We shall investigate its properties and stability characteristics next.

Definition 3.5.4 (Strongly Convex)**.** *Let $\lambda \geq 0$ and let $(S, \|\cdot\|)$ be a normed space. We call a function $f : X \to \mathbb{R}$ strongly λ-convex if for every $x_1, x_2 \in S$ and $\alpha > 0$, we have*

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2) - \frac{\lambda}{2} \alpha (1-\alpha) \|x_1 - x_2\|^2. \tag{3.117}$$

For functions which are 0-strongly convex, we also use the term convex.

Lemma 3.5.5. *Let $\lambda \geq 0$ and let $(S, \|\cdot\|)$ be a normed space. Then, we have the following properties:*

1. *$\tilde{r}_\lambda$ is 2λ-strongly convex.*

2. *If $f : X \to \mathbb{R}$ is λ-strongly convex and $g : X \to \mathbb{R}$ is convex, then $f+g$ is λ-strongly convex.*

3. *If $f : X \to \mathbb{R}$ is λ-strongly convex and $\tilde{x} = \arg\min_{x \in S} f(x)$, then for any $y \in X$,*

$$f(y) - f(\tilde{x}) \geq \frac{\lambda}{2}\|\tilde{x} - y\|^2. \tag{3.118}$$

Proof. Shalev-Shwartz [44, Lemma 13.5]. □

Proposition 3.5.6. *Let $\lambda > 0$, let $\rho > 0$, let $s \in \mathbb{N}_0$ and let $\mathcal{L} \in \mathcal{C}^s(\mathbb{R}^n \times \mathbb{R}^n, [0, \infty))$ be a loss function that is ρ-Lipschitz in the first variable and such that the mapping*

$$\theta \mapsto \mathcal{L}(\mathcal{R}_{N,\varrho}(x, \theta), y) \tag{3.119}$$

is convex for every $(x, y) \in \mathbb{R}^n \times \mathbb{R}^n$. Furthermore, let $\mathcal{A}$ be a learning algorithm that solves the regularized optimization problem for neural networks w.r.t. the loss function $\mathcal{L}$, $\mathbf{z}$ and the regularizer $\tilde{r}_\lambda$. Then, $\mathcal{A}$ is on-average-replace-one-stable with rate $\frac{2\rho^2}{\lambda m}$, and it holds that

$$\mathbb{E}\left[\mathcal{E}_Z^{\mathcal{L}}(\mathcal{A}(\mathcal{Z})) - \mathcal{E}_{\tilde{\mathcal{Z}}}^{\mathcal{L}}(\mathcal{A}(\mathcal{Z}))\right] \leq \frac{2\rho^2}{\lambda m}. \tag{3.120}$$

Proof. Shalev-Shwartz [44, Corollary 13.6]. □

Under the assumptions from Proposition 3.5.6, the stability term decreases if the regularization parameter λ increases. One might note that the quadrature loss is only Lipschitz continuous when restricted to a compact space, which is satisfied for our assumptions on the hypothesis space $\mathcal{N}_{N,\varrho,R}(\mathcal{D}, \mathbb{R}^n)$.
So far we have seen how regularizers can be utilized to control the size of the parameters towards better stability in the sense of generalization. In general does regularization act as a soft constraint on the learning problem, imposing additional requirements on the parameters. In the next chapter we will see how this strategy can also balance multiple error terms, offering a possibility to learn multiple tasks at once towards a common learning goal. In fact, we will regularize the parameters θ not directly but through constraints on the realization function $\mathcal{R}_{N,\varrho}(\cdot, \theta)$ with the help of additional data for the regularizer.

4 PDEs as a Learning Problem

The purpose of this chapter is to familiarize ourselves with partial differential equations (PDEs), a widely used tool for describing different natural and man-made phenomena in science and engineering. The usage of PDEs opens up possibilities to model easy to complex behaviors in physics and other fields, which are often derived from first principles. However, one caveat with the study of PDEs is that a significant number of problems do not admit solvability in the analytical way towards an explicit formula. The use of numerical methods allows researchers to accurately and efficiently approximate solutions of these complex equations and understand the behavior of these systems. However, numerical approximation has its downsides, such as computational effort and introduction of additional error, and is typically only point-wise available.
We reinterpret the problem stated by a PDE as a learning problem and will present our algorithm as a potential way to choose approximators in the hypothesis space of neural networks. Additionally, we present an abstract error bound on the resulting algorithm, supporting our choice.

4.1 Framework

To begin with, we will establish the necessary framework for studying PDEs through the introduction of relevant notation for differential operators. To illustrate the diversity of PDEs and the challenges that can arise in their solution, we will also provide several examples. Note that, while we will limit ourselves to one-dimensional PDEs, the definitions can be extended by considering systems of equations.

Definition 4.1.1 (Differential Operators)**.** *Let $d, k \in \mathbb{N}$, let $U \subset \mathbb{R}^d$ be open and let $F : \mathbb{R}^{d^k} \times \mathbb{R}^{d^{k-1}} \times \cdots \times \mathbb{R}^d \times \mathbb{R} \times U \to \mathbb{R}$. Then the mapping L which for any function $u : U \to \mathbb{R}$ and $x \in U$ is given by*

$$L(u)(x) := F\left(D^k u(x), D^{k-1}u(x), \ldots, Du(x), u(x), x\right), \tag{4.1}$$

defines a k^{th}-order differential operator on the region U. A differential operator can take following forms:

1. *L is called linear, if it is of the form*

$$L(u)(x) = \sum_{|\alpha| \leq k} a_\alpha(x) D^\alpha u(x) \tag{4.2}$$

for some functions $a_\alpha : U \to \mathbb{R}$.

2. *L is called semilinear, if it is of the form*

$$L(u)(x) = \sum_{|\alpha|=k} a_\alpha(x) D^\alpha u(x) + \tilde{L}(u)(x) \tag{4.3}$$

for some functions $a_\alpha : U \to \mathbb{R}$ and $\tilde{L}$, a differential operator of order $\leq k-1$ on region U.

3. *L is called quasilinear, if it is of the form*

$$L(u)(x) = \sum_{|\alpha|=k} \tilde{L}_\alpha(u)(x) D^\alpha u(x) + \tilde{L}_0(u)(x) \tag{4.4}$$

for some $\tilde{L}_\alpha$ and $\tilde{L}_0$, differential operators of order $\leq k-1$ on region U.

4. *L is called non-linear, if it possesses non-linearities on one or more of the highest-order derivatives.*

Definition 4.1.2 (Partial Differential Equation)**.** *Let $d, k \in \mathbb{N}$, let $U \subset \mathbb{R}^d$ be open and let L be a differential operator of order k on the region U. Then we call the equation*

$$L(u) = 0 \tag{4.5}$$

a k^{th}-order partial differential equation describing unknown functions $u : U \to \mathbb{R}$.

A PDE is considered solved if we can find all functions verifying the equation (4.5), though in general this does not yield a unique solution. Moreover, most of the times it is not possible to explicitly write down a solution, hence in these situations we are only interested in certain properties that characterise the solutions.
Often in physics, we want to describe space and time dependent phenomena through PDEs, namely functions $u : U \times (0,T) \to \mathbb{R}$, with $T > 0$. We denote by the variable $x \in U$ space and $t \in (0,T)$ the time component, and we write

$$U \times (0,T) \ni (x,t) \mapsto u(x,t). \tag{4.6}$$

In the following, we will restrict our attention to PDEs of this specific form. To allow for the existence and uniqueness of solutions, we will also consider auxiliary boundary conditions defined on the boundary of the domain U, as well as initial value conditions on the time coordinate t.

Definition 4.1.3 (Auxiliary Conditions)**.** *Let $T > 0$, let $d, k \in \mathbb{N}$, let $U \subset \mathbb{R}^d$ be open and bounded and let L be a differential operator of order k on the region $U \times (0,T)$. Then the following is defined as an initial-value problem*

$$\begin{cases} L(u) = 0 & \text{on } U \times (0,T) \\ u(\cdot, 0) = \varphi & \text{on } U \end{cases} \tag{4.7}$$

for some $\varphi : U \to \mathbb{R}$. Further, we call

$$\begin{cases} L(u) = 0 & \text{on } U \times (0,T) \\ \mathcal{G}(u) = \psi & \text{on } \partial U \times [0,T] \end{cases} \tag{4.8}$$

a boundary-value problem for some $\mathcal{G}$, a differential operator of order $\leq k$ on the region $\bar{U} \times [0,T]$ and some function $\psi : \bar{U} \times [0,T] \to \mathbb{R}$. If a problem states both an initial and a boundary condition, we call it initial/boundary-value problem.

Sometimes the differential operator $\mathcal{G}$ in Definition 3.1.3 is called the boundary operator. Since the definition of PDEs is rather broad, it encompasses a vast range of problems, making it infeasible to develop a single general theory of solvability for all PDE forms. To gain an understanding of the complexity of such a theory and to familiarize ourselves with the concept of PDEs, we will consider a few examples in the following. It should be noted that we will present only a limited number of problems and that it is beyond the scope of this thesis to delve deeply into the various branches of PDEs. Instead, we refer the reader to the works of Evans [34], Brezis [35], Folland [14] and Renardy [28] for a comprehensive overview of numerous examples and a starting point for further exploration.

Example 4.1.4 (PDE Examples)**.** *We will state five differential operators modelling different simple physical phenomena of high interest in research and touch on their meaning in the context of mathematical applications. Our setting follows Definition 3.1.1 for differential operators, with consideration of space and time variables if needed.*

1. *Laplace's equation. The differential operator that describes Laplace's equation has the linear form*

$$L(u) = \Delta u = \sum_{i=1}^{d} u_{x_i x_i}. \tag{4.9}$$

   Typically, Laplace's equation is interpreted to describe the density of some quantity in equilibrium, including e.g. chemical concentrations.

2. *Transport equation. A transport equation is a hyperbolic PDE and has a linear differential operator of the form*

$$L(u) = u_t + b \cdot \nabla u, \tag{4.10}$$

   where $b \in \mathbb{R}^d$. The resulting problem models transport phenomena, like fluid movements through space and time.

3. *Reaction-diffusion equation. Reaction-diffusion equations are of semilinear form and can be modelled by the differential operator*

$$L(u) = u_t - \nu \Delta u - f(u), \tag{4.11}$$

   for some $\nu > 0$ and function f. Phenomena described by problems rising from this operator contain temperature distributions.

4. *Burgers' equation. Burgers' equation is another fundamental PDE and its semi-linear differential operator is given by*

$$L(u) = u_t + uu_x - \beta u_{xx}, \tag{4.12}$$

where $\beta \in \mathbb{R}$. The PDE counts fluid dynamics and traffic flow modelling to its applications.

As noted earlier and illustrated through the examples, it is not entirely clear what the different ways of solving PDEs contain, as this highly depends on the form of the differential operator and on possible auxiliary conditions. We shall differentiate between the problems in need of numerical methods and the ones which can be solved by analytical methods.

Definition 4.1.5 (Solutions of PDEs)**.** *Let $T > 0$, let $d, k \in \mathbb{N}$, let $U \subset \mathbb{R}^d$ be open and bounded and let L be a differential operator of order k on the region $U \times (0, T)$. Further, assuming implicit auxiliary conditions, we call a problem well-posed, if it holds that:*

1. *There exists an $\mathbf{u} : U \times (0, T) \to \mathbb{R}$ which solves the PDE $L(u) = 0$.*
2. *The solution is unique.*
3. *The solution depends continuously on the conditions specifying the PDE.*

If additionally $\mathbf{u} \in \mathcal{C}^k(U \times (0, T))$, we call $\mathbf{u}$ a classical solution. When formulating the differential operator L with weak derivatives, we call $\mathbf{u}$ a weak solution.

Classical solutions to PDEs can be obtained through the use of analytical techniques, such as separation of variables or the method of characteristics (cf. for instance Evans [34]). These techniques involve expressing the solution to the PDE as a mathematical formula. However, it may not always be possible to obtain classical solutions to PDEs, particularly when the equations are nonlinear or have complex boundary conditions. In such cases, numerical methods can be employed to approximate the solution to the PDE. Furthermore, we often encounter the case that even without classical solution, the weak formulated problem is well-posed cite (cf. for instance Brezis [35]). Note that a solution to the PDE is also a solution to the weak formulated PDE.
While numerical methods provide a useful alternative for solving PDEs that may not have a known analytical solutions, they also have their limitations, such as the possibility of errors and the requirement of additional computational resources (cf. for instance Morten [29] and Thomas [17]).
Furthermore, note that we introduced auxiliary conditions in a general setting, only assuming bounded spatial domain U. In specific problems, it may not be enough for a classical solution to be k times differentiable on the interior of the region $U \times (0, T)$, but additional conditions are needed for the solution function and its derivatives to ensure the boundary condition to be well-defined. In some cases, this may require the solution function to be differentiable up to some order on the closure of the region, or to satisfy

suitable growth or decay conditions near the boundary. In other cases, we need only to consider certain types of spatial domains that contain more restrictive assumptions on the boundary of U.

4.2 PDEs as a Learning Problem

The search for solutions to a PDE with auxiliary conditions can be seen as a learning problem, where the solution itself is viewed as a statistical model that is being learned from samples. While we are interested in cases where solutions are not available through analytical methods, we can view the learning problem in the sense of chapter 2 with the data $Z = (X, \tilde{\mathbf{u}}(X))$, where $\tilde{\mathbf{u}}$ is a point-wise approximation through numerical methods. However, this approach might not always be computationally feasible, and includes an approximation error of the method. Additionally, it does not make use of any condition describing the PDE. On the other hand, only the initial condition can be stated in the form of data for solutions to be learned in a supervised way. One way to overcome this issue is to restrict the hypothesis space to functions that satisfy both other conditions. As this poses a rather strong restriction and involves the problem of finding such hypothesis spaces, we also obtain from this idea.
However, we have seen in the study of neural networks that we can include constraints by introducing regularizers in an optimization problem. This opens up the possibility to state the learning problem of PDEs in a regularized form, where the differential and boundary operator are utilized as soft constraints. In this section, we shall demonstrate how this idea can be formulated within the framework of the mathematical learning problem as described in chapter 2, under the assumptions specified in Definition 4.1.3.

Definition 4.2.1 (Regularized Learning Problem for PDEs)**.** *Let $T > 0$, let $d, k \in \mathbb{N}$, let $U \subset \mathbb{R}^d$ be open and bounded and let L be a differential operator of order k on the region $U \times (0,T)$ with auxiliary conditions describing an initial/boundary-value problem*

$$\begin{cases} L(u) = 0 & \text{on } U \times (0,T) \\ u(\cdot, 0) = \varphi & \text{on } U \\ \mathcal{G}(u) = \psi & \text{on } \partial U \times [0,T]. \end{cases} \tag{4.13}$$

Furthermore, let $\mathcal{D}_t \subset U$, $\mathcal{D}_d \subset U \times (0,T)$ and $\mathcal{D}_s \subset \partial U \times [0,T]$ be compact subsets and let $X^t : \Omega \to \mathcal{D}_t$, $X^d : \Omega \to \mathcal{D}_d$ and $X^s : \Omega \to \mathcal{D}_s$ be independent uniformly distributed[1] random vectors on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Then we define by

$$Z = (X^t, Y^t) = (X^t, \varphi(X^t)) \tag{4.14}$$

the data and define the regularized error function $\tilde{\mathcal{E}}_{\tilde{Z}} : \mathcal{C}^k(U \times (0,T)) \to [0, \infty]$ w.r.t. the

[1] uniformly distributed on a compact $\mathcal{D} \subset \mathbb{R}^d$ refers here to the probability density function assigning the same value to every element in $\mathcal{D}$ and 0 outside. The probability of such a random vector X taking on a subset $D \subset \mathcal{D}$ is given by the Lebesgue measure of D divided by the Lebesgue measure of $\mathcal{D}$, i.e., $\mathbb{P}(X \in D) = \lambda(D)/\lambda(\mathcal{D})$.

data and regularization data $\tilde{Z} = (Z, X^d, X^s)$ *by*

$$\tilde{\mathcal{E}}_{\tilde{Z}}(u) := \mathcal{E}_Z(u(\cdot, 0)) + \int_\Omega \|L(u)(X^d)\|^2 d\mathbb{P} + \int_\Omega \|\mathcal{G}(u)(X^s) - \psi(X^s)\|^2 d\mathbb{P}. \tag{4.15}$$

The regularized learning problem defines the search for functions $u \in \mathcal{C}^k(U \times (0,T))$ *that minimize the error function* $\tilde{\mathcal{E}}_{\tilde{Z}}$*. We refer to the last two error terms in (4.15) as the regularizing error terms.*

Note that in Definition 4.2.1, the map $\Omega \ni \omega \mapsto Y^t(\omega)$ is a random variable if φ is $\mathcal{B}(U)/\mathcal{B}(\mathbb{R})$-measurable, which we assume to hold for the purpose of conforming to the learning theory developed in Chapter 2. Furthermore, in order for the error to be well-defined, we also assume $L(u)$, $\mathcal{G}(u)$ and ψ to be measurable.
It is also important to note that even though the differential operator L and the boundary operator $\mathcal{G}$ are not directly included in the definition of data, we are still only interested in functions u that approximately satisfy all conditions of the PDE. In other words, we seek functions u such that

$$\begin{aligned} u(X^t, 0) &\approx Y^t \\ L(u) &\approx 0 \\ \mathcal{G}(u) &\approx \psi \end{aligned} \tag{4.16}$$

holds. Heuristically speaking, we are forcing the learning method to respect the PDE conditions as a soft constraint by adding the error terms of the operators to the error function. Further, by limiting the data to the initial condition, we can learn the underlying phenomena in a straightforward manner and avoid the need for any additional efforts to gather point-wise information about the PDE.

Remark 4.2.2. *Under the assumption that there exists a numerical method which approximates the solutions of problem (4.13) point-wise for every* $(x,t) \in U \times (0,T)$*, we are able to state the learning problem with additional data and further to extend Definition 4.2.1 by another error term through application of the method to uniformly at random sampled points inside* $U \times (0,T)$*. It should be noted that the computation of the point-wise solutions may be computationally intensive and that an error may be introduced through this method.*

It is also worth noting that the previously obtained result for the regression function is no longer valid, as the minimizer must not only take data Z into account, but additionally needs to minimize the regularizing error terms. Under the assumption that the considered PDE is well-posed and in possession of a classical solution, it is easy to see that the minimizing function is exactly this solution. This tells us, that the learning problem we stated in Definition 4.2.1 describes a valid reinterpretation for the problem stated by a PDE.
In order to properly address the altered error function in our current situation, we must redefine the empirical error function in accordance with the modified learning problem.

Previously, we considered m samples of the data Z. However, in the current analysis, we require additionally for each of the random vectors X^d and X^s an individual amount of samples m_d, $m_s \in \mathbb{N}$. Recall that we can obtain $m \in \mathbb{N}$ i.i.d. copies of a random vector, which allows for the revised empirical error function. We will now assume the setting specified in Definition 4.2.1.

Definition 4.2.3 (Regularized Empirical Error for PDEs)**.** *Let $m_t, m_d, m_s \in \mathbb{N}$, let $\mathcal{Z} = ((X_i^t, Y_i^t))_{i=1}^{m_t}$ be m_t i.i.d. copies of Z and let $\mathcal{X}^d = (X_i^d)_{i=1}^{m_d}$ and $\mathcal{X}^s = (X_i^s)_{i=1}^{m_s}$ be m_d and m_s i.i.d. copies of X^d and X^s. For every function $u \in \mathcal{C}^k(U \times (0,T))$, we define the empirical error function w.r.t. $\tilde{\mathcal{Z}} = (\mathcal{Z}, \mathcal{X}^d, \mathcal{X}^s)$ as $\tilde{\mathcal{E}}_{\tilde{\mathcal{Z}}} : \Omega \times \mathcal{C}^k(U \times (0,T)) \to [0, \infty]$ given by*

$$\begin{aligned}\tilde{\mathcal{E}}_{\tilde{\mathcal{Z}}}(\omega, u) := &\mathcal{E}_{\mathcal{Z}}(\omega, u(\cdot, 0)) \\ &+ \frac{1}{m_d} \sum_{i=1}^{m_d} \|L(u)(X_i^d(\omega))\|^2 \\ &+ \frac{1}{m_s} \sum_{i=1}^{m_s} \|\mathcal{G}(u)(X_i^s(\omega)) - \psi(X_i^s(\omega))\|^2.\end{aligned} \tag{4.17}$$

These rather drastic changes in the error functions raise the question of whether our previously developed results still hold. So far, we obtained from specifying any further details on function spaces, where L and $\mathcal{G}$ operate and did only assume measurability. This already entails for a fixed $u \in \mathcal{C}^k(U \times (0,T))$ that the function

$$\tilde{\mathcal{E}}_{\tilde{\mathcal{Z}}}(u) : \Omega \to [0, \infty), \quad \omega \mapsto \tilde{\mathcal{E}}_{\tilde{\mathcal{Z}}(\omega)}(u) = \tilde{\mathcal{E}}_{(\mathcal{Z}(\omega), \mathcal{X}^d(\omega), \mathcal{X}^s(\omega))}(u) \tag{4.18}$$

is again $\mathcal{F}/\mathcal{B}([0,\infty])$-measurable.
Furthermore, the results of Section 2.2 ask for square integrability of all error terms. Assuming that the differential operators map u into the function space $L^2(U \times (0,T), \mathbb{R}; \mathbb{P}_{X^d})$ for L and $L^2(\bar{U} \times [0,T], \mathbb{R}; \mathbb{P}_{X^s})$ for $\mathcal{G}$ and by only allowing for $\varphi \in L^2(U, \mathbb{R}; \mathbb{P}_{X^t})$, $\psi \in L^2(\bar{U} \times [0,T], \mathbb{R}; \mathbb{P}_{X^s})$, and $u(\cdot, 0) \in L^2(U, \mathbb{R}; \mathbb{P}_{X^t})$, measurability and square integrability can be assured. To allow for these results to hold, we will only consider PDEs that possess the mentioned properties unless otherwise mentioned.

4.3 PINNs as a Special Regularization Approach

By construction, neural networks with smooth activation functions are fully differentiable w.r.t. all input coordinates and parameters θ, which makes them a fitting option for applying differential operators. Further, by the universal approximation theorem we can approximate continuous and even measurable functions arbitrarily good by realization functions of neural networks, which makes the hypothesis space of neural networks a natural choice for our learning problem. In this section we propose our algorithm, by utilizing the concept of regularized optimization problems to encode the soft constraints derived in the last section into the loss function. The resulting learned neural network

using this strategy is called a physics-informed neural network and was first introduced by Dissanayake [13] and Lagaris [15] and further developed by Raissi [57]. We will assume the settings of Definition 4.2.3 for PDEs and the settings of Definition 3.3.1 and Definition 3.5.3.

Algorithm 4.3.1 (Learning Strategy for PDEs with Neural Networks)**.**

Input *$T > 0$, $U \subset \mathbb{R}^d$ open and bounded, L differential operator of order k on region $U \times (0,T)$ with auxiliary conditions describing an initial/boundary-value problem*

$$\begin{cases} L(u) = 0 & \text{on } U \times (0,T) \\ u(\cdot,0) = \varphi & \text{on } U \\ \mathcal{G}(u) = \psi & \text{on } \partial U \times [0,T]. \end{cases} \tag{4.19}$$

(i) *Choose an architecture $a = (N, \varrho)$ with $\varrho \in \mathcal{C}^k(\mathbb{R},\mathbb{R})$ and hyper-parameter $R > 0$ for the hypothesis space of neural networks $\mathcal{N}_{N,\varrho,R}(U \times (0,T),\mathbb{R})$ and consider the regularized optimization problem w.r.t the quadrature loss, $\mathbf{z} = ((x_i^t, y_i^t))_{i=1}^{m_t}$ and regularizer*

$$r(\theta) := \frac{\lambda_d}{m_d} \sum_{i=1}^{m_d} |L(\mathcal{R}_{N,\varrho}(\cdot,\theta))(x_i^d)|^2 + \frac{\lambda_s}{m_s} \sum_{i=1}^{m_s} |\mathcal{G}(\mathcal{R}_{N,\varrho}(\cdot,\theta))(x_i^s) - \psi(x_i^s)|^2 \tag{4.20}$$

for some $\lambda_d > 0$ and $\lambda_s > 0$, where $\mathbf{x}^d = (x_i^d)_{i=1}^{m_d}$ and $\mathbf{x}^s = (x_i^s)_{i=1}^{m_s}$ are realizations of m_d and m_s samples independently drawn according to the distribution of X^d and X^s.

(ii) *Choose a gradient-based optimization algorithm, the step sizes $(\gamma_k)_{k\in\mathbb{N}}$ and numbers of samples m_t, m_d and m_s.*

(iii) *Apply the optimization algorithm to the regularized optimization problem and run it until a local minimum or a satisfactory level of accuracy is reached.*

We call the neural network which uses parameters learned through the Algorithm 4.3.1 a physics-informed neural network (PINN). Note that the PINN and the trained parameters depend on $\mathbf{x} := (\mathbf{z}, \mathbf{x}^d, \mathbf{x}^s)$ used in the steps of Algorithm 4.3.1. To emphasize this, we write for the trained parameters a map

$$\mathbf{x} \mapsto \theta^*(\mathbf{x}) \in [-R, R]^{p(N)} \tag{4.21}$$

and consequently for the PINN

$$\mathbf{x} \mapsto u^*(\mathbf{x}) := \mathcal{R}_{N,\varrho}(\cdot, \theta^*(\mathbf{x})). \tag{4.22}$$

We can express the map θ^* as a concatenation of the regularized loss function, the optimization step using the gradient of the resulting loss and the map $\theta^{(i)} \mapsto \mathcal{R}_{N,\varrho}(\cdot, \theta^{(i)})$

for updating the parameters, starting at some $\theta^{(0)}$. Hence θ^* is measurable, if we assume satisfying function spaces for the differential operators and regular enough functions φ and ψ of the PDE, since we consider gradient-based optimization algorithms and $\theta \mapsto \mathcal{R}_{N,\varrho}(\cdot, \theta)$ is measurable. This also implies that the map u^* is measurable under these assumptions. For the sake of brevity, we omit the explicit details on the required assumptions for the PDE and the specific expression of the map θ^*.
Further, note that the objective function for the considered regularized optimization problem coincides with the empirical error function defined in Definition 4.2.3, when choosing $\lambda_d = 1$ and $\lambda_s = 1$. Again, intuitively this strategy is forcing the neural network to learn every condition representing the PDE simultaneously. However, it is not immediately clear that minimizing the regularizer term in (4.20) would result in any control over the approximation capabilities of the PINN. In the following, we will consider under which circumstances a PINN can be guaranteed to be a good approximator of the PDE, which will also act as a more concrete reasoning for why we are considering PINNs as solution strategies.
The main idea behind PINNs is in utilizing the differential operator L in the regularizer instead of learning solely from data. Because of this and for simplicity we assume that auxiliary conditions are implicitly contained in the differential operator L, which also includes an implicit assumption for sufficient regularity on the boundary of U, in order to yield a regular enough solution. Further, for definiteness we will focus on function spaces $X^* \subset W^{k,q}(U \times (0,T), \mathbb{R})$ and $Y^* \subset L^p(U \times (0,T))$ for $L : X^* \to Y^*$ to operate on, which assumes some additional regularity on the solutions of the PDE. For the sake of simplicity in notation we set $p = 1$ in the theorem. Further, note that we only need to consider the random variable X^d from Definition 4.2.1 in this setting and that the hypothesis space of neural networks is a closed subspace of $W^{k,q}(U \times (0,T), \mathbb{R})$, if we choose an activation function $\varrho \in \mathcal{C}^k(\mathbb{R}, \mathbb{R})$.

Theorem 4.3.2 (Error Estimate of PINNs)**.** *Let $T > 0$, let $d \in \mathbb{N}$, let $U \subset \mathbb{R}^d$ be open and bounded, let $\mathcal{D} \subset U \times (0,T)$ be closed, let $1 \leq q < \infty$, let $k \in \mathbb{N}_0$ and let $X^* \subset W^{k,q}(U \times (0,T), \mathbb{R})$ and $Y^* \subset L^1(U \times (0,T), \mathbb{R})$ be closed subspaces with norms $\|\cdot\|_{X^*}$ and $\|\cdot\|_{Y^*}$ and let $L : X^* \to Y^*$ be a differential operator of order k on the region $U \times (0,T)$, describing the abstract PDE*

$$L(u) = 0, \tag{4.23}$$

including implicit auxiliary conditions. Further, let u^* be the map described in (4.22) trained until a local minimum for a given hypothesis space $\mathcal{N}_{N,\varrho,R}(U \times (0,T), \mathbb{R})$ with some activation function $\varrho \in \mathcal{C}^k(\mathbb{R}, \mathbb{R})$ and $X^d : \Omega \to \mathcal{D}$. Let $m_T, m_V \in \mathbb{N}$ and let $\mathcal{X}^T = (X_i^T)_{i=1}^{m_T}$ and $\mathcal{X}^V = (X_i^V)_{i=1}^{m_V}$ be two independent i.i.d. samples of X^d. We assume for L, that the following holds:

(A1) : $\|L(u)\|_{Y^*} < \infty$ *for all* $u \in X^*$, *with* $\|u\|_{X^*} < \infty$.

(A2) : There exists a unique solution $\mathbf{u} \in X^*$ *of (4.23).*

(A3) : *For any $\omega_1, \omega_2 \in \Omega$, it holds that*

$$\|u^*(\mathcal{X}^T(\omega_1)) - u^*(\mathcal{X}^T(\omega_2))\|_{W^{k,q}} \leq C\|L(u^*(\mathcal{X}^T(\omega_1))) - L(u^*(\mathcal{X}^T(\omega_2)))\|_1, \quad (4.24)$$

where the $C > 0$ is a constant and explicitly depends on $\|u^*(\mathcal{X}^T(\omega_1))\|_\infty$ and $\|u^*(\mathcal{X}^T(\omega_2))\|_\infty$.

Then, it holds that

$$\mathbb{E}[\|\mathbf{u} - u^*(\mathcal{X}^T)\|_{W^{k,q}}] \leq C \cdot \left(\mathcal{E}_T + \mathcal{E}_{TV} + \frac{std^*}{\sqrt{m_V}}\right), \quad (4.25)$$

where we define

$$\mathcal{E}_T := \mathbb{E}\left[\frac{1}{m_T}\sum_{i=1}^{m_T}\left|L(u^*(\mathcal{X}^T))(X_i^T)\right|\right] \quad (4.26)$$

$$\mathcal{E}_V := \frac{1}{m_V}\int_{\mathcal{D}^{m_T}}\sum_{i=1}^{m_V}\left|L(u^*(\mathbf{x}^T))(X_i^V)\right| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T) \quad (4.27)$$

$$\mathcal{E}_{TV} := \mathbb{E}\left[|\mathcal{E}_T - \mathcal{E}_V|\right] \quad (4.28)$$

and

$$std^* := \sqrt{\mathbb{E}\left[\left|\int_{\mathcal{D}^{m_T}}\left|L(u^*(\mathbf{x}^T))(X_1^V)\right| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T) - \int_{\mathcal{D}}\int_{\mathcal{D}^{m_T}}|L(u^*(\mathbf{x}^T))(x)| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T)dx\right|^2\right]}. \quad (4.29)$$

Proof. First, we note that

$$\omega \mapsto \frac{1}{m_V}\int_{\mathcal{D}^{m_T}}\sum_{i=1}^{m_V}\left|L(u^*(\mathbf{x}^T))(X_i^V(\omega))\right| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T) \quad (4.30)$$

is $\mathcal{F}/\mathcal{B}(\mathbb{R})$-measurable which implies that $\mathcal{E}_V$ is a random variable, while $\mathcal{E}_T \geq 0$ is not. Further, we can compute

$$\begin{aligned}
&\mathbb{E}[\|\mathbf{u} - u^*(\mathcal{X}^T)\|_{W^{k,q}}] \\
&\leq C \cdot \mathbb{E}[\|L(\mathbf{u}) - L(u^*(\mathcal{X}^T))\|_1] \\
&= C \cdot \mathbb{E}[\|L(u^*(\mathcal{X}^T))\|_1] \\
&\leq C \cdot \mathbb{E}\left[\left|\mathbb{E}[\|L(u^*(\mathcal{X}^T))\|_1] - \mathcal{E}_V + \mathcal{E}_V - \mathcal{E}_T + \mathcal{E}_T\right|\right] \\
&\leq C \cdot \left(\mathbb{E}\left[\left|\mathbb{E}[\|L(u^*(\mathcal{X}^T))\|_1] - \mathcal{E}_V\right|\right] + \mathbb{E}\left[|\mathcal{E}_V - \mathcal{E}_T|\right] + \mathbb{E}\left[|\mathcal{E}_T|\right]\right) \\
&= C \cdot \left(\mathcal{E}_T + \mathcal{E}_{TV} + \mathbb{E}\left[\left|\mathbb{E}[\|L(u^*(\mathcal{X}^T))\|_1] - \mathcal{E}_V\right|\right]\right) \\
&\leq C \cdot \left(\mathcal{E}_T + \mathcal{E}_{TV} + \sqrt{\mathbb{E}\left[\left|\mathbb{E}[\|L(u^*(\mathcal{X}^T))\|_1] - \mathcal{E}_V\right|^2\right]}\right),
\end{aligned} \quad (4.31)$$

where we used assumption (A3) in the first inequality, in the forth line the triangle inequality, and in the sixth line Hölder's inequality with $\frac{1}{2}$. To finish the prove, we rewrite the term under the square root

$$\begin{aligned}
&\mathbb{E}\left[\left|\mathbb{E}[\|L(u^*(\mathcal{X}^T))\|_1] - \mathcal{E}_V\right|^2\right] \\
&= \mathbb{E}\left[\left|\int_{\mathcal{D}^{m_T}}\int_{\mathcal{D}} |L(u^*(\mathbf{x}^T))(x)| dx d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T) - \frac{1}{m_V}\int_{\mathcal{D}^{m_T}} \sum_{i=1}^{m_V} \left|L(u^*(\mathbf{x}^T))(X_i^V)\right| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T)\right|^2\right] \\
&= \frac{1}{m_V^2}\sum_{i=1}^{m_V} \mathbb{E}\left[\left|\int_{\mathcal{D}^{m_T}}\int_{\mathcal{D}} |L(u^*(\mathbf{x}^T))(x)| dx d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T) - \int_{\mathcal{D}^{m_T}} \left|L(u^*(\mathbf{x}^T))(X_i^V)\right| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T)\right|^2\right] \\
&= \frac{1}{m_V}\mathbb{E}\left[\left|\int_{\mathcal{D}}\int_{\mathcal{D}^{m_T}} |L(u^*(\mathbf{x}^T))(x)| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T) dx - \int_{\mathcal{D}^{m_T}} \left|L(u^*(\mathbf{x}^T))(X_1^V)\right| d\mathbb{P}_{\mathcal{X}^T}(\mathbf{x}^T)\right|^2\right],
\end{aligned} \tag{4.32}$$

where we utilized in the second line that $(X_i^V)_{i=1}^{m_V}$ are independent random variables, and in the third line that they are distributed identically, along with Fubini-Tonelli. □

Using additional independent samples as $\mathcal{X}^V$ for the evaluation of a trained neural network to assist in the process of hyper-parameters selection (tuning) is a common practise called validation, and it also aids in monitoring overfitting behavior in supervised learning. One can even validate the model while training to adjust hyper-parameters like step sizes inside of the optimization. For the estimate (4.25), the samples $\mathcal{X}^V$ are used to measure the gap $\mathcal{E}_{TV}$ to the independent samples used for training $\mathcal{X}^T$, which can be a good indicator of the generalisation capabilities of the PINN (cf. Definition 3.5.1), when using a well chosen amount of samples m_V. Furthermore, the constant C reflects the stability of the underlying PDE and depends on the boundedness of the solution $\mathbf{u}$ as well as of the PINNs. The estimate (4.25) serves to relate the expected error between the solution and PINNs to $\mathcal{E}_T$, the so-called expected training error of the neural network, establishing a connection by leveraging the assumed stability of the PDE. The expected training error is a good measure on how well we can fit the given samples used for training, this bound strengthens our idea that a well-trained PINN can be used to approximate a PDE. Recall that this is the same as asking for small optimization error (cf. Proposition 2.4.9).
However, we note that this is rather meant to illustrate the mechanics of the PINN algorithm and that an error estimate involves in practise more work including the auxiliary conditions directly. To get some concrete examples, we reference to Mishra [71], where this error estimate is discussed on a few important initial/boundary-value problem classes.

Remark 4.3.3 (Other Bounds on PINNs)**.** *Next to the presented estimate in Theorem 4.3.2, Mishra [71] also discusses an estimate for non-random points used in training,*

which can be applied for low dimensional PDEs. Instead of considering the expected error, the estimate is computed for the error

$$\|\mathbf{u} - u^*(\mathbf{x}^T)\|_{W^{k,q}}, \tag{4.33}$$

where $\mathbf{x}^T$ *are the* m_T *points used in training, relating it to the training error*

$$\left(\sum_{i=1}^{m_T} \left|L(u^*(\mathbf{x}^T))(\mathbf{x}_i^T)\right|^p\right)^{\frac{1}{p}}. \tag{4.34}$$

Details on this bound can be found in Mishra [71, Theorem 2.6]. Another work that focuses on understanding the approximation capabilities of PINNs was done by Ryck [73], which specifically examines Kolmogorov PDEs.

Remark 4.3.4 (Other Learning Strategies for PDEs with Neural Networks)**.** *In addition to PINNs, alternative learning strategies for deep neural network have been widely used in the numerical approximation of PDEs. One method involves the use of explicit or semi-implicit representation formulas, such as the Feynman-Kac formula for parabolic and elliptic PDEs. The compositional structure of these formulas can then be utilized for approximation by deep neural networks. This approach has been presented and analyzed for a range of parametric elliptic, parabolic, and linear transport PDEs (cf. for instance Beck [63]).*
Another approach in the use of deep learning for the numerical approximation of PDEs involves augmenting existing numerical methods with deep learning-based modules. For example, free parameters of numerical schemes can be learned from data using this approach (cf. for instance Mishra [56]).
A third strategy involves learning observables or quantities of interest of the solutions to the underlying PDEs from data. This approach has been discussed in the context of uncertainty quantification, PDE constrained optimization, and model order reduction (cf. for instance Lye [61]).

5 Numerical Experiments

In this chapter, we investigate, through numerical experiments, whether PINNs, generated using our proposed Algorithm 4.3.1, are an appropriate choice for solving our learning problem of PDEs. Additionally, we will evaluate how the results obtained from different problem classes relate to the bound found in Theorem 4.3.2.
In contrast to the classical supervised learning task, we do not equip our neural network with a large amount of samples during training. Therefore, we choose to run LBFGS (cf. Liu [8]) as our optimization algorithm and search over learning rates from 1e-4 to 2. Note that LBFGS needs a twice continuously differentiable activation function. Following previous work (cf. Raissi [57]), we use the hyperbolic tangent function as the activation function in our examples. So far, we have not mentioned the influence of the choice of the initial parameters $\theta^{(0)}$ of a neural network on the optimization. We choose to initialize the weights $W^{(\ell)}$ and biases $b^{(\ell)}$ in layer $\ell \in \{1, ..., L\}$ uniformly at random on the interval $[-\sqrt{1/N_{\ell-1}}, \sqrt{1/N_{\ell-1}}]$ to improve training speed and effectiveness by decreasing the risk of convergence to sub-optimal minima. Additionally, we follow the common practice of normalizing the input to ensure a smooth loss landscape (cf. LeCun [25]). Furthermore, we disregard the choice of the hyper-parameter $R > 0$ of the hypothesis space of neural networks since it is not necessary to manually specify a bound for the size of the neural network parameters in practice. This is because they are automatically constrained by the specific limitations of the computing environment. We apply the quadrature loss function for training the PINNs and experiment with adding $L2$-regularizers to the loss function, for different $\lambda > 0$. To fully understand the behavior of PINNs, one needs to run experiments not only with a fixed neural network architecture but also allow the parameters determining the neural network to be searched for. However, this is beyond the scope of this thesis, which is why we pick reasonable architectures for each model problem.
We will exclusively present 1-dimensional model problems and choose bounded intervals $U = (x_L, x_R) \subset \mathbb{R}$ as the spatial domains. This leaves us with the boundary for values of x as $\partial U = \{x_L, x_R\}$. Consequently, we will sample our samples used for training in the following way:

1. Interior training samples $(X_i^d)_{i=1}^{m_d}$, i.i.d. uniformly distributed random variables on $(x_L, x_R) \times (0, T)$.

2. Temporal boundary training samples $(X_i^t)_{i=1}^{m_t}$ i.i.d. uniformly distributed random variables on (x_L, x_R).

3. Spatial boundary training samples $(X_i^s)_{i=1}^{m_s}$, i.i.d. uniformly distributed random variables on $\{x_L, x_R\} \times [0, T]$ or for periodic boundary conditions on $[0, T]$.

For the weights λ_d and λ_s in the regularizer (4.20), we select both to be 1, resulting in the abstract loss function

$$\Upsilon_{reg}(\theta) = \frac{1}{m_t}\sum_{i=1}^{m_t}|\delta_i^t|^2 + \frac{1}{m_d}\sum_{i=1}^{m_d}|\delta_i^d|^2 + \frac{1}{m_s}\sum_{i=1}^{m_s}|\delta_i^s|^2, \tag{5.1}$$

where δ_i^t, δ_i^d and δ_i^s are the error terms of the initial condition, the differential operator, and the boundary condition, respectively. All precise hyper-parameters used in training of each example can be found in Tables 2, 3 and 4 in the Appendix.
To assess the performance of our numerical solutions in comparison to the actual solutions (or suitable approximations, if they do not exist), of the PDEs in question, we evaluate

1. the mean squared error (MSE),
2. the mean squared relative error (MSRE) modified with $\epsilon = 1$,
3. and the maximum error (L_∞)

between the functions, using Monte Carlo integration (cf. for instance Graham [38] and Keller [33] on Monte Carlo integration) with 10^5 uniformly at random chosen samples on the domain of the PDE. We also compute the empirical error of the trained neural network with the points used for training ($\mathcal{E}_T^2$) to compare the relation with the errors as in Theorem 4.3.2. Moreover, for each setting in the experiments, we run 10 independent trials and present the results in the form of the mean and standard deviation.
The implementation for the experiments was written in the Python programming language using the open-source deep learning package Pytorch, as well as the experiment packages Ray and Weights&Biases. The experiments were implemented within a larger framework that includes a version of the source code using the Hydra package (cf. Yadan [59]) to enhance the efficiency of configuration planning for running multiple experiments. Both versions of the source code for the experiments can be found at `https://github.com/.../theory2practise_ext`. The experiments were executed on a MacBook Air (2020) equipped with an Apple M1 chip.
For the sake of readability, we adapt the notation slightly in the following examples and write u_θ for the realization function $\mathcal{R}_{N,\varrho}(\cdot,\theta)$.

5.1 Convection Equation

In our first experiment, we apply our algorithm to convection equations, a class of simple linear problems,

$$u_t + \beta u_x = 0, \tag{5.2}$$

where $\beta > 0$ is called the convection coefficient. We choose a function $\varphi \in \mathcal{C}^1([x_L, x_R])$ as the initial condition, and study periodic boundary conditions

$$u(x_L, t) = u(x_R, t), \quad \forall t \in [0, T]. \tag{5.3}$$

Problems of the form (5.2) with boundary condition (5.3) have an analytical solution that can be derived using Fourier transforms

$$\mathbf{u}(x,t) = \mathcal{F}^{-1}(\mathcal{F}(\varphi(x))e^{-i\beta tk}), \tag{5.4}$$

where k denotes the frequency in the Fourier domain. We are interested to approximate the solution $\mathbf{u}$ by training PINNs on the initial/boundary-value problem. For this example, the regularized objective function that we want to minimize with neural networks has error terms

$$\begin{aligned} \delta_i^t &= u_\theta(x_i^t, 0) - \varphi(x_i^t) \\ \delta_i^d &= \frac{\partial u_\theta}{\partial t}(x_i^d) + \beta \frac{\partial u_\theta}{\partial x}(x_i^d) \\ \delta_i^s &= u_\theta(x_L, x_i^s) - u_\theta(x_R, x_i^s)). \end{aligned} \tag{5.5}$$

We consider this problem with a maximal time of $T = 1$, in spatial domain $U = (0, 2\pi)$, with initial condition $\varphi(x) = sin(x)$, and for different values of the convection coefficient $\beta \in \{0.001, 0.01, 0.1, 1, 5, 10, 20, 30, 40, 50, 100, 200\}$, and with varying numbers of samples $m_d \in \{128, 256, 512, 1024, 2048\}$ and $m_t = m_s \in \{32, 64, 128, 256\}$. Since our choice of φ is periodic with period 2π and we have $x_R - x_L = 2\pi$, the solution

$$\mathbf{u}(x,t) = sin(x - \beta t) \tag{5.6}$$

satisfies all conditions of the PDE and is therefore used in the computation of the errors. Training is performed with a 5-layer neural network and with neurons

$$N = (2, 50, 50, 50, 50, 1) \tag{5.7}$$

for one optimizer step.

Tables 5.1-6 present the resulting mean squared relative error and maximum error between the model trained with the propsed algorithm and the exact solution of the PDE with different values of $\beta \in \{1, 30, 50\}$ and varying numbers of samples. We can observe that in Table 5.1 and 5.2, the needed amount of samples to achieve a certain level of performance is less than in Table 5.3 and 5.4 and certainly less than in Table 5.5 and 5.6. Hence, we can reason that for more complex PDEs (cf. Figure 5.1), we might need higher amounts of samples in training. Furthermore, it is apparent that even when sampling amounts are scaled up, the error from the less complex PDE cannot be reached anymore but saturates at an inferior level. This indicates that sheer scaling efforts must not be rewarded with better accuracies. Intuitively, we would assume that with higher amounts of samples, the errors would go down or at least stay at a similar level. However, due to numerical variation in neural network training, we cannot always be certain to face this situation and meet natural fluctuations. When we take a look at the maximum error, it is also clear that at $\beta = 50$, the PINN is no longer a valid option for approximating the PDE, as the maximum error surpasses the maximal absolute value of the solution itself. In Table 5.7, we have selected the best observed sampling amounts and shown the

resulting errors for scaling in the convection coefficient β. With an increase of complexity and samples, the optimization algorithm needs more iterations, which we can see in the time differences. Moreover, we can still observe a relationship between training error and other errors in Table 5.7, as suggested by the results of the theorem. Furthermore, in Figure 5.2, we recognize that after $\beta = 10$, the errors start to grow quickly, which might be explained by the complexity, but can also be understood as a result of rougher loss landscapes that require more sophisticated initialization methods (cf. Krishnapriyan [66]). Hence, we can argue that even that the bound from Theorem 4.3.2 promises good results for well-trained PINNs, we still need to make sure that our algorithm is well-prepared to reach the required levels, which requires special attention, e.g. with naive tools like splitting the time domain in smaller pieces and learning for each time interval individually (cf. Krishnapriyan [66]).

m_d \ m_t/m_s	16	32	64	128
32	0.0208+/-0.0179	0.0136+/-0.0046	0.0109+/-0.0051	0.0107+/-0.0039
64	0.0093+/-0.0049	0.0057+/-0.0012	0.0039+/-0.0013	0.0045+/-0.0015
128	0.0059+/-0.0027	0.0043+/-0.0010	0.0047+/-0.0019	0.0035+/-0.0003
256	0.0101+/-0.0041	0.0041+/-0.0022	0.0039+/-0.0006	0.0033+/-0.0007

Table 5.1: Convection MSRE with $\beta = 1$.

m_d \ m_t/m_s	16	32	64	128
32	0.1062+/-0.0199	0.0664+/-0.0239	0.0857+/-0.0455	0.0806+/-0.0376
64	0.0473+/-0.0209	0.0339+/-0.0153	0.0454+/-0.0268	0.0327+/-0.0049
128	0.0313+/-0.0081	0.0210+/-0.0044	0.0210+/-0.0075	0.0170+/-0.0035
256	0.0459+/-0.0165	0.0174+/-0.0039	0.0173+/-0.0058	0.0117+/-0.0028

Table 5.2: Convection L_∞ with $\beta = 1$.

m_d \ m_t/m_s	64	128	256	512
512	0.3548+/-0.1243	0.2356+/-0.1771	0.1440+/-0.1641	0.1714+/-0.1585
1024	0.3777+/-0.0262	0.1728+/-0.1613	0.0236+/-0.0101	0.0536+/-0.0836
2048	0.3397+/-0.1323	0.0579+/-0.0885	0.0298+/-0.0636	0.0239+/-0.0187
4196	0.3033+/-0.1232	0.0481+/-0.0167	0.0239+/-0.0187	0.0188+/-0.0077

Table 5.3: Convection MSRE with $\beta = 30$.

m_d \ m_t/m_s	64	128	256	512
512	1.0150+/-0.3700	0.7305+/-0.4927	0.4757+/-0.4720	0.5422+/-0.5058
1024	1.1290+/-0.0112	0.6259+/-0.4545	0.1941+/-0.2556	0.2236+/-0.3416
2048	1.0130+/-0.2919	0.2689+/-0.3191	0.1391+/-0.0686	0.1346+/-0.0289
4196	0.9867+/-0.3068	0.1792+/-0.0683	0.1991+/-0.3018	0.0967+/-0.0481

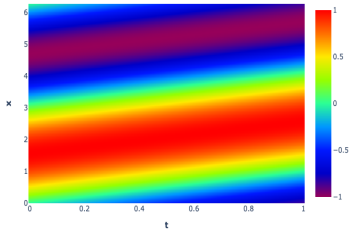
Table 5.4: Convection L_∞ with $\beta = 30$.

m_d \ m_t/m_s	64	128	256	512
512	0.4782+/-0.0543	0.4343+/-0.0833	0.4248+/-0.0685	0.3428+/-0.0423
1024	0.5886+/-0.0725	0.4055+/-0.0717	0.3995+/-0.0665	0.3137+/-0.0028
2048	0.4723+/-0.1269	0.3697+/-0.0479	0.3327+/-0.0578	0.3168+/-0.0208
4196	0.5370+/-0.1418	0.3588+/-0.0565	0.3288+/-0.0174	0.3198+/-0.0214

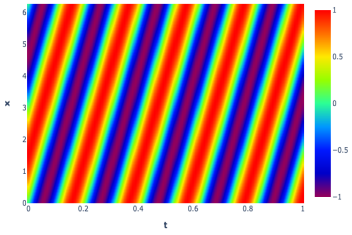
Table 5.5: Convection MSRE with $\beta = 50$.

m_d \ m_t/m_s	64	128	256	512
512	1.5550+/-0.0841	1.4210+/-0.3422	1.3360+/-0.3266	1.1780+/-0.1269
1024	1.7811+/-0.1670	1.5100+/-0.2718	1.4062+/-0.2584	1.0500+/-0.0282
2048	1.5040+/-0.4078	1.2300+/-0.1835	1.1620+/-0.1833	1.0530+/-0.0548
4196	1.6040+/-0.3331	1.2270+/-0.2159	1.0630+/-0.0447	1.0520+/-0.0418

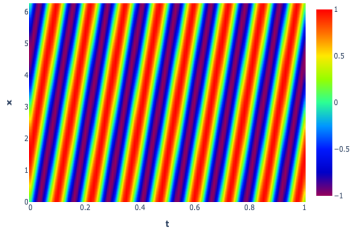
Table 5.6: Convection L_∞ with $\beta = 50$.



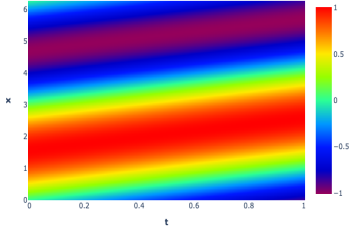
(a) Exact for $\beta = 1$



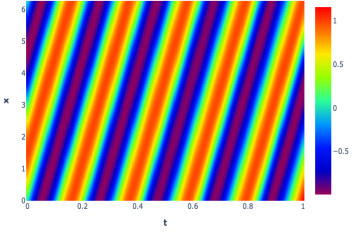
(b) Exact for $\beta = 30$



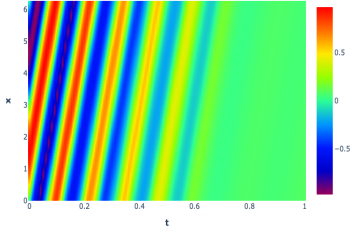
(c) Exact for $\beta = 50$



(d) PINN for $\beta = 1$



(e) PINN for $\beta = 30$



(f) PINN for $\beta = 50$

β	Time [s]	MSE	MSRE	L_∞	$\mathcal{E}_T^2$
0.001	1.62	0.0052+/-0.0011	0.0036+/-0.0007	0.0150+/-0.0034	6.4e-05+/-2.9e-05
0.01	1.29	0.0044+/-0.0012	0.0029+/-0.0007	0.0115+/-0.0030	6.2e-05+/-2.9e-05
0.1	1.73	0.0076+/-0.0029	0.0052+/-0.0019	0.0210+/-0.0082	9.4e-05+/-5.2e-05
1	1.69	0.0062+/-0.0018	0.0033+/-0.0007	0.0117+/-0.0028	1.1e-04 +/-3.4e-05
5	5.01	0.0079+/-0.0027	0.0052+/-0.0018	0.0281+/-0.0095	1.1e-04 +/-4.7e-05
10	19.73	0.0054+/-0.0027	0.0034+/-0.0018	0.0185+/-0.0079	6.1e-05 +/-2.7e-05
20	69.41	0.0225+/-0.0109	0.0142+/-0.0069	0.0714+/-0.0312	2.7e-04 +/-1.6e-04
30	116.08	0.0301+/-0.0131	0.0188+/-0.0077	0.0967+/-0.0481	2.4e-04 +/-1.3e-04
40	112.09	0.3709+/-0.2229	0.2056+/-0.1204	0.8043+/-0.4312	0.0108 +/-0.0078
50	128.17	0.5695+/-0.0430	0.3198+/-0.0214	1.0520+/-0.0418	0.0128 +/-0.0029
100	100.30	0.6805+/-0.0118	0.3762+/-0.0076	1.1320+/-0.0734	0.0103 +/-0.0018
200	153.21	0.7044+/-0.0122	0.3900+/-0.0093	1.1390+/-0.0597	0.0096 +/-0.0017

Table 5.7: Selected errors and average wall time for the convection PDE with sample amounts $m_d = 256$ and $m_s = m_t = 128$ for $\beta \in \{0.001, 0.01, 0.1, 1, 5\}$, and $m_d = 4196$ and $m_s = m_t = 512$ for upper β.

5.2 Heat Equation

Let us now turn towards a problem with another differential operator. We will be examining a linear class of problems known as heat equations with varying diffusivity constant $\alpha > 0$,

$$u_t = \alpha \Delta u. \tag{5.8}$$

As the initial condition, we will select a function $\varphi \in \mathcal{C}^k((x_L, x_R))$ with $k \geq 2$, and we will study zero Dirichlet boundary conditions

$$u(x_L, t) = u(x_R, t) = 0, \quad \forall t \in [0, T]. \tag{5.9}$$

It is important to note that in general, if we allow for $\varphi \in W^{k+1,2}((x_L, x_R))$, then the existence and uniqueness of a solution in $W^{k,2}((x_L, x_R) \times (0, T))$ can be proven (cf. for instance Friedman [3]). With our assumption on φ and for fixed α, the PDE even possesses a solution in the classical sense (cf. for instance Cannon [4]), namely

$$\mathbf{u}(x, t) = \int_{-\infty}^{\infty} \frac{e^{\frac{-(x-s)^2}{4\alpha t}}}{\sqrt{4\pi\alpha t}} \varphi(s) ds. \tag{5.10}$$

We can use the properties of the integral identity of equation (5.10) to approximate the true solution function $\mathbf{u}$ of the initial/boundary-value problem by numerical integration. This approximation can be used to measure how well the PINN approximates the solution $\mathbf{u}$. The regularized objective function that we seek to minimize using neural networks

includes error terms

$$\begin{aligned} \delta_i^t &= u_\theta(x_i^t, 0) - \varphi(x_i^t) \\ \delta_i^d &= \frac{\partial u_\theta}{\partial t}(x_i^d) - \alpha \frac{\partial^2 u_\theta}{\partial x^2}(x_i^d) \\ \delta_i^s &= u_\theta(x_i^s). \end{aligned} \tag{5.11}$$

We consider the problem in the temporal domain until $T = 1$ and spatial domain of $U = (-5, 5)$, with the bump function restricted to the interval $[-3, 3]$ as our initial condition,

$$\varphi(x) = \mathbb{1}_{[-3,3]} e^{-\frac{1}{1-(x/3)^2}} \tag{5.12}$$

and choose the diffusivity coefficients $\alpha \in \{1, 2, 3, 4, 5\}$.

To train the neural network, we use varying numbers $m_d \in \{32, 64, 128, 256, 512\}$ for spatial and consequently $m_t = m_s \in \{16, 32, 64, 128, 256\}$ for temporal and boundary samples. The neural network used for training is a 5-layer architecture with neurons

$$N = (2, 20, 20, 20, 20, 1) \tag{5.13}$$

and was trained for one optimizer step. Additionally, we apply $L2$-regularization with $\lambda = 10^{-6}$.

Using our algorithm with these specific configurations mentioned, we compare the mean squared relative error and maximum error for different sampling amounts for $m_s = m_t$ and for $\alpha \in \{1, 3, 5\}$ in Tables 5.8-13. Unlike in the earlier example, the complexity in these three values for α does not vary as much (cf. Figure 5.3). However, we still observe an increase in the mean squared relative error with increasing α, which might be caused by the slightly rising complexity of the solution. On the other hand, the maximum error remains constant over all three PDEs due to the similar nature of the solutions. The early saturation in errors in Table 5.8-13 further confirms our observations from the earlier example and raises the question of whether we can achieve similar performances by fixing either m_d or $m_s = m_t$, thereby forcing an imbalance in the training data. Figure 5.4 displays the average errors for a fixed m_d over multiple trials with $\alpha = 5$, allowing the samples $m_s = m_t$ to be in the range mentioned above. Not only does the average error saturate, but the deviation also decreases. This indicates that the ratio between m_d and $m_s = m_t$ plays a significant role, as we need m_d to surpass a certain threshold for the best results. Furthermore, this behavior is expected since we assume an increase in robustness and better generalization capabilities of the PINN with growing variation in input data. Table 5.14 compares the selected errors for the best performing sample variation over $\alpha \in \{1, 2, 3, 4, 5\}$. We can see that the ratio between the errors L_∞ and $\mathcal{E}_T^2$ fluctuates little, while the mean squared error and mean squared relative error do scale up in α, which suggests the relationship developed from Theorem 4.3.2.

m_d \ m_t/m_s	16	32	64	128
32	0.0134+/-0.0055	0.0069+/-0.0017	0.0076+/-0.0031	0.0132+/-0.0061
64	0.0105+/-0.0037	0.0045+/-0.0011	0.0059+/-0.0022	0.0055+/-0.0012
128	0.0092+/-0.0056	0.0069+/-0.0013	0.0049+/-0.0007	0.0036+/-0.0012
256	0.0098+/-0.0063	0.0068+/-0.0031	0.0051+/-0.0013	0.0045+/-0.0005

Table 5.8: Heat MSRE with $\alpha = 1$.

m_d \ m_t/m_s	16	32	64	128
32	0.1000+/-0.0561	0.0916+/-0.0641	0.0553+/-0.0509	0.1744+/-0.1420
64	0.0565+/-0.0196	0.0531+/-0.0481	0.0461+/-0.0141	0.0749+/-0.0829
128	0.0682+/-0.0415	0.1325+/-0.0574	0.1115+/-0.0896	0.0765+/-0.0770
256	0.1534+/-0.1600	0.0620+/-0.0325	0.0698+/-0.0344	0.0491+/-0.0113

Table 5.9: Heat L_∞ with $\alpha = 1$.

m_d \ m_t/m_s	16	32	64	128
32	0.0176+/-0.0059	0.0172+/-0.0155	0.0153+/-0.0042	0.0195+/-0.0119
64	0.0121+/-0.0046	0.0118+/-0.0048	0.0085+/-0.0020	0.0089+/-0.0031
128	0.0093+/-0.0025	0.0081+/-0.0011	0.0073+/-0.0008	0.0069+/-0.0009
256	0.0149+/-0.0119	0.0079+/-0.0011	0.0075+/-0.0076	0.0071+/-0.0003

Table 5.10: Heat MSRE with $\alpha = 3$.

m_d \ m_t/m_s	16	32	64	128
32	0.0708+/-0.0307	0.1054+/-0.1031	0.0522+/-0.0087	0.0685+/-0.0297
64	0.0865+/-0.0577	0.0449+/-0.0069	0.0537+/-0.0338	0.1139+/-0.1260
128	0.0674+/-0.0275	0.0777+/-0.0485	0.0347+/-0.0032	0.0368+/-0.0039
256	0.1029+/-0.0686	0.0610+/-0.0427	0.0451+/-0.0111	0.0448+/-0.0109

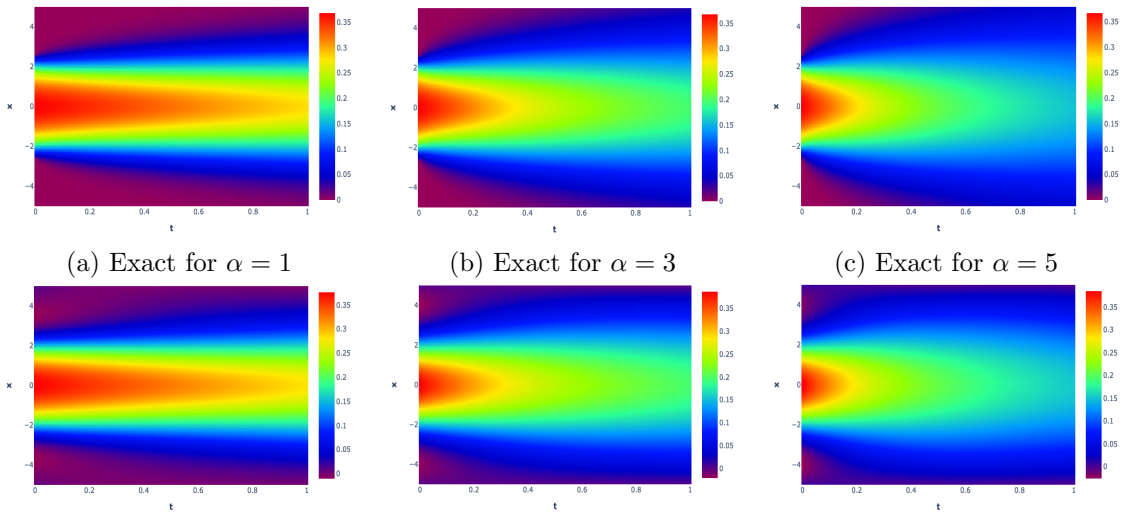
Table 5.11: Heat L_∞ with $\alpha = 3$.

m_d \ m_t/m_s	16	32	64	128
32	0.0253+/-0.0168	0.0288+/-0.0101	0.0277+/-0.0098	0.0235+/-0.0074
64	0.0181+/-0.0056	0.0127+/-0.0009	0.0145+/-0.0043	0.0133+/-0.0016
128	0.0157+/-0.0029	0.0136+/-0.0015	0.0124+/-0.0004	0.0124+/-0.0004
256	0.0135+/-0.0006	0.0127+/-0.0005	0.0124+/-0.0003	0.0124+/-0.0002

Table 5.12: Heat MSRE with $\alpha = 5$.

m_d \ m_t/m_s	16	32	64	128
32	0.0952+/-0.0528	0.0779+/-0.0279	0.0810+/-0.0209	0.0816+/-0.0376
64	0.0914+/-0.0298	0.0879+/-0.0625	0.0544+/-0.0036	0.0654+/-0.0318
128	0.0638+/-0.0119	0.0523+/-0.0024	0.0579+/-0.0127	0.0908+/-0.0871
256	0.0854+/-0.0607	0.0451+/-0.0035	0.0539+/-0.0018	0.0527+/-0.0023

Table 5.13: Heat L_∞ with $\alpha = 5$.



(a) Exact for $\alpha = 1$ (b) Exact for $\alpha = 3$ (c) Exact for $\alpha = 5$

(d) PINN for $\alpha = 1$ (e) PINN for $\alpha = 3$ (f) PINN for $\alpha = 5$

Figure 5.3: Exact and PINN solutions of the heat PDE as heatmaps. PINNs were trained with $m_d = 256$ and $m_s = m_t = 128$ samples.

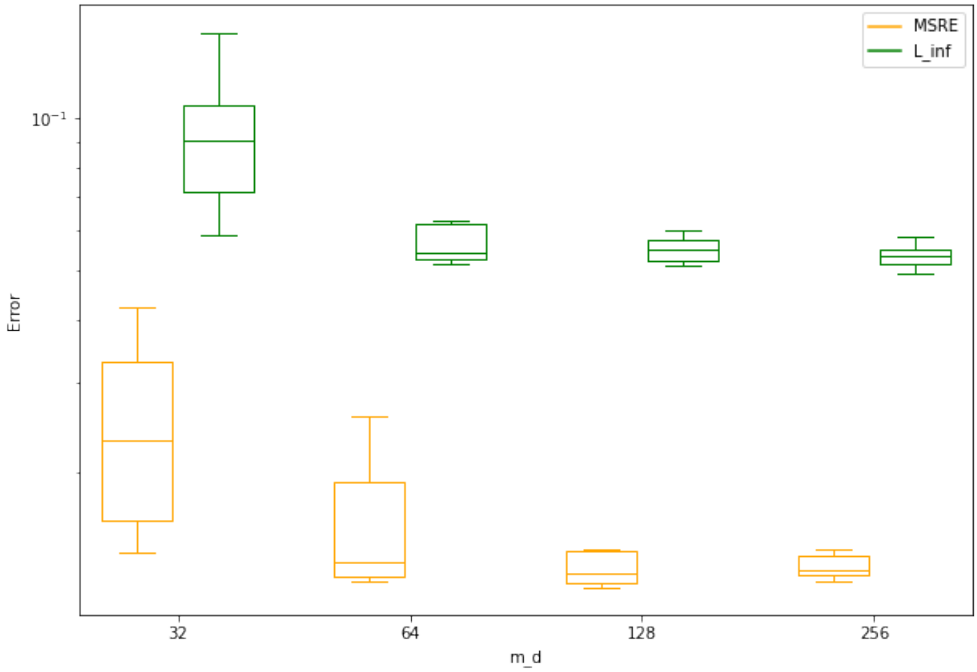


Figure 5.4: Heat MSRE (yellow) and L_∞ (green) with $\alpha = 5$ for varying sample amounts, averaged over different $m_s = m_t$.

α	Time [s]	MSE	MSRE	L_∞	$\mathcal{E}_T^2$
1	1.11	0.0049+/-0.0006	0.0045+/-0.0005	0.0491+/-0.0113	2.8e-04+/-6.5e-05
2	1.10	0.0058+/-0.0004	0.0054+/-0.0003	0.0543+/-0.2820	4.7e-04+/-3.7e-05
3	1.94	0.0076+/-0.0004	0.0071+/-0.0003	0.0448+/-0.0109	3.3e-04+/-7.2e-05
4	2.62	0.0103+/-0.0002	0.0096+/-0.0002	0.0559+/-0.0272	4.2e-04+/-5.6e-05
5	2.69	0.0133+/-0.0003	0.0124+/-0.0002	0.0527+/-0.0023	3.7e-04+/-6.2e-05

Table 5.14: Selected errors and average wall time for the heat PDE with sample amounts $m_d = 256$ and $m_s = m_t = 128$.

5.3 Burgers' Equation

Finally, we examine the well-known viscous and inviscid Burgers' equation, which is a slightly more complicated class of quasi-linear problems,

$$u_t + uu_x = \nu u_{xx} \tag{5.14}$$

with the viscosity coefficient $0 < \nu \ll 1$ for the viscous version and $\nu = 0$ in the inviscid case. Once again, we choose a function $\varphi \in \mathcal{C}^k([x_L, x_R])$ with $k \geq 1$ as the initial condition and study zero Dirichlet boundary conditions

$$u(x_L, t) = u(x_R, t) = 0, \quad \forall t \in [0, T]. \tag{5.15}$$

By applying the Cole-Hopf transformation to the viscous case, the problem (5.14) can be converted to a linear equation and solved in the classical sense (cf. for instance Evans [34]). Given the solution and inverting the transformation, we obtain

$$\mathbf{u}(x,t) = -2\nu \frac{\partial}{\partial x} ln \left(\frac{1}{\sqrt{4\pi\nu t}} \int_{-\infty}^{\infty} e^{\frac{-(x-s)^2}{4\nu t} - \frac{1}{2\nu}\int_0^s \varphi(r)dr} ds \right). \tag{5.16}$$

The inviscid case can only be solved if the initial condition does not produce shock waves, which we are interested in.

In both cases, we discretize the space and time variables and apply finite difference methods (cf. for instance Thomas [17]) to obtain an approximation to the solutions. We then use this approximation to measure the performance of the PINN. Hence, in this example, the samples used in the computation of the errors are chosen from a discrete set.

The specific error terms are

$$\begin{aligned} \delta_i^t &= u_\theta(x_i^t, 0) - \varphi(x_i^t) \\ \delta_i^d &= \frac{\partial u_\theta}{\partial t}(x_i^d) + u_\theta(x_i^d)\frac{\partial u_\theta}{\partial x}(x_i^d) - \nu \frac{\partial^2 u_\theta}{\partial x^2}(x_i^d) \\ \delta_i^s &= u_\theta(x_i^s). \end{aligned} \tag{5.17}$$

For this example, we choose the maximal time of $T = 1$ and the spatial domain of $U = (-1, 1)$. The initial condition is given by

$$\varphi(x) = -\sin(\pi x) \tag{5.18}$$

and we consider the viscosity coefficient $\nu = \frac{0.01}{\pi}$ for the viscous case.

To train the PINN with our algorithm, we utilize varying numbers of spatial samples $m_d \in \{1024, 2048, 4096, 8192, 16384, 32768\}$, while choosing the amount for temporal and boundary samples as $m_s = m_t \in \{64, 128, 256, 512, 1024\}$. The neural network employed

for training uses neurons

$$N = (2, 20, 20, 20, 20, 20, 20, 20, 20, 1) \tag{5.19}$$

for one optimizer step.

We present the numerical results in form of mean squared relative error and maximum error for various combinations of the sampling amounts in tables 5.15-5.18 for both the viscous and inviscid case. We notice immediately that only high numbers for m_d and auxiliary conditions result in saturating errors for the viscous case. Further, in the inviscid case, we face saturation sooner, but at relatively high level. Therefore, we can rule out PINNs as a valid choice for an approximator of this problem with our chosen setting. Neither the mean squared relative error nor the maximum error seem to decrease by raising the number of samples. Moreover, the maximum error exceeds the solution's absolute value range. The gap between the performance for the different ν values might be explained by the nature of the physical phenomenon, which develops in our inviscid case shocks and seams to be much harder to learn (cf. Figure 5.5 and Figure 5.6). Here again, we see that there are some PDEs where a PINN is not capable of capturing the conditions. Moreover, it is reasonable to classify the Burgers' equation as a more complex task to learn compared to the earlier problems, which gives an indication of the requirement of higher data sample amounts in more complex PDEs.

Table 5.19 shows our selected errors and computing time for the best performing combinations of sampling amounts. Even that the errors for the viscous case are somewhat promising, we must take into account the high computational effort and the fact that $m_d = 32768$ and $m_s = m_t = 1024$ include unreasonable high amounts of details for the chosen domain $U \times (0, T) = (-1, 1) \times (0, 1)$.

m_d \ m_t/m_s	128	256	512	1024
1024	0.1098+/-0.0290	0.0694+/-0.0385	0.0770+/-0.0396	0.0652+/-0.0292
2048	0.0557+/-0.0339	0.0559+/-0.0179	0.0693+/-0.0618	0.0599+/-0.0408
4096	0.0502+/-0.0203	0.0484+/-0.0169	0.0392+/-0.0049	0.0388+/-0.0145
8192	0.0473+/-0.0283	0.0189+/-0.0094	0.0439+/-0.0437	0.0166+/-0.0161
16382	0.0693+/-0.0203	0.0467+/-0.0202	0.0168+/-0.0098	0.0075+/-0.0025
32768	0.0643+/-0.0158	0.0257+/-0.0151	0.0235+/-0.0140	0.0071+/-0.0021

Table 5.15: Burger MSRE with $\nu = \frac{0.01}{\pi}$.

m_d \ m_t/m_s	128	256	512	1024
1024	1.9400+/-0.6537	1.4190+/-0.5577	1.5660+/-0.5441	1.5730+/-0.3001
2048	1.2500+/-0.6175	1.3240+/-0.3079	0.9926+/-0.5393	1.2100+/-0.5838
4096	1.1030+/-0.4395	1.2460+/-0.3872	0.9299+/-0.1763	1.0130+/-0.3489
8192	0.9361+/-0.5111	0.5367+/-0.2879	0.7458+/-0.4221	0.4298+/-0.4277
16382	1.5000+/-0.2994	1.2470+/-0.3780	0.4583+/-0.2715	0.1691+/-0.0659
32768	1.3890+/-0.3960	0.7290+/-0.4341	0.5292+/-0.2722	0.2002+/-0.0812

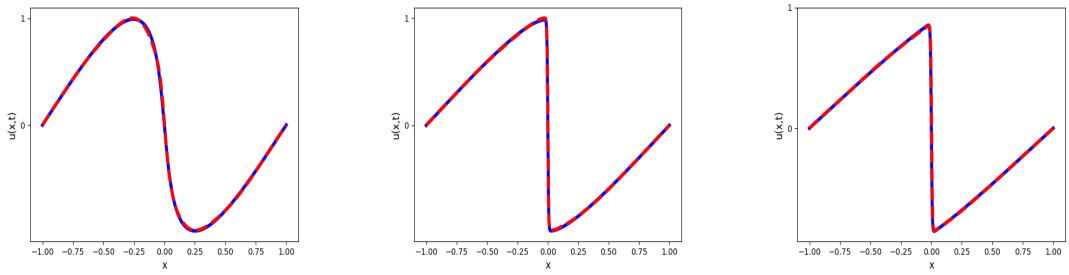
Table 5.16: Burger L_∞ with $\nu = \frac{0.01}{\pi}$.

m_d \ m_t/m_s	128	256	512	1024
1024	0.1300+/-0.0164	0.1255+/-0.0083	0.1224+/-0.0456	0.1140+/-0.0175
2048	0.1391+/-0.0072	0.1295+/-0.0067	0.1348+/-0.0129	0.1335+/-0.0321
4096	0.1198+/-0.0298	0.1383+/-0.0128	0.1340+/-0.0124	0.1391+/-0.0078
8192	0.1247+/-0.0414	0.1299+/-0.0067	0.1355+/-0.0017	0.1399+/-0.0009

Table 5.17: Burger MSRE with $\nu = 0$.

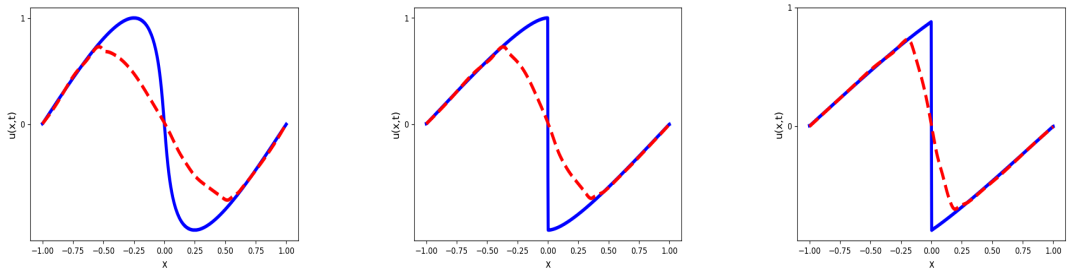
m_d \ m_t/m_s	128	256	512	1024
1024	1.6680+/-0.4316	1.4840+/-0.4489	1.6400+/-0.1254	1.767+/-0.1274
2048	1.4950+/-0.4095	1.2790+/-0.2770	1.5220+/-0.1770	1.612+/-0.1105
4096	1.2500+/-0.2692	1.4140+/-0.2663	1.6910+/-0.2661	1.511+/-0.4115
8192	1.4510+/-0.4710	1.6390+/-0.1760	1.5050+/-0.0874	1.395+/-0.0341

Table 5.18: Burger L_∞ with $\nu = 0$.



(a) $t = 0.25$ (b) $t = 0.5$ (c) $t = 0.75$

Figure 5.5: Exact (blue) and PINN (red) solutions of the viscious Burgers' equation. PINNs were trained with $m_d = 32786$ and $m_s = m_t = 1024$ samples.



(a) $t = 0.25$ (b) $t = 0.5$ (c) $t = 0.75$

Figure 5.6: Exact (blue) and PINN (red) solutions of the inviscid Burgers' equation. PINNs were trained with $m_d = 32786$ and $m_s = m_t = 1024$ samples.

ν	Time [s]	MSE	MSRE	L_∞	$\mathcal{E}_T^2$
$0.01/\pi$	486.39	0.0113+/-0.0024	0.0071+/-0.0021	0.2002+/-0.0812	0.6237+/-0.1953
0	375.82	0.2442+/-0.0302	0.1289+/-0.0115	1.2380+/-0.2210	0.5163+/-0.2316

Table 5.19: Selected errors and average wall time for the Burgers' equation with sample amounts $m_d = 32768$ and $m_s = m_t = 1024$.

6 Conclusion

In this thesis, we demonstrated physics-informed neural networks as a special case of regularization techniques in neural networks and embedded them within the framework of mathematical learning theory. Our numerical analysis in Chapter 5 on three different types of partial differential equations revealed promising results for simple problems, such as the convection equation with a low coefficient and heat equation, with a relatively small amount of training data. However, we also encountered limitations in capturing the underlying phenomena and approximating solutions to a satisfactory degree of accuracy for more complex examples. Despite our efforts to improve the performance by scaling up the amount of training data, we found that this method alone does not result in arbitrarily good error rates, which seemingly saturated at certain levels for all discussed examples. We note that further optimization through hyper-parameter tuning and different initialization methods could achieve even better results then we presented. The more complex problems also presented challenges in terms of the unreasonable high amount of training data required, where traditional numerical mathematical methods are more practical at present. Nevertheless, our results confirmed the relationship between the use of a regularized loss and the approximation capabilities of physics-informed neural networks, making them a promising avenue for future research.
In conclusion, we can state that for the model problems considered in this study, physics-informed neural networks should not be considered as a main approximator and analytical solutions or efficient numerical approximation methods are preferable.
It is worth noting that the application of neural networks as solvers for physical phenomena is a relatively new field of research when compared to the long history of efforts in classical numerical mathematics. Nevertheless, the promising results achieved thus far and the growing trend in applying deep learning to PDEs (cf. for instance Mishra [56], Ryck [73] and Shin [62]), offer great hope for future advancements. Some efforts aim to overcome current limitations (cf. Krishnapriyan [66]) and other studies explore different strategies for improvement (cf. Beck [63]). Additionally, ongoing theoretical developments continue to deepen our understanding of the capabilities and potential of these methods (cf. for instance Mishra [71], Ryck [73] and Shin [62]).

Appendices

.1 Hyper-parameters used in Numerical Experiments

Description	Value	Variable
Architecture		
input	2	N_0
output	1	N_L
parameter initialization $(W^{(\ell)}, b^{(\ell)})$	$\mathcal{U}([-\sqrt{1/N_{\ell-1}}, \sqrt{1/N_{\ell-1}}])$	
activation function	tanh	ϱ
Optimization		
algorithm	L-BFGS	
maximal number of iterations	10^6	
tolerance on first order optimality	1.0e-7	
tolerance on function changes	1.0e-7	
update history size	50	
line search function	Wolfe	
optimization steps	1	
Evaluation		
number of samples	10^5	
evaluation errors	MSE, MSRE, L_∞	

Table 1: General hyper-parameters of the experiments in Chapter 5.

Description	Value	Variable
Architecture		
depth	5	L
width	50	N_ℓ
Experiment		
spatial samples	{128, 256, 512, 1024, 2048}	m_d
boundary condition samples	{32, 64, 128, 256}	m_s
initial condition samples	{32, 64, 128, 256}	m_t
learning rate	1.5	
spatial domain	$(0, 2\pi)$	U
time maximum	1	T

Table 2: Hyper-parameters of the experiments in Section 5.1.

Description	Value	Variable
Architecture		
depth	5	L
width	20	N_ℓ
Experiment		
spatial samples	{32, 64, 128, 512}	m_d
boundary condition samples	{16, 32, 64, 128, 256}	m_s
initial condition samples	{16, 32, 64, 128, 256}	m_t
learning rate	1	
spatial domain	$(-5, 5)$	U
time maximum	1	T
Regularization		
$L2$-regularization	10^{-6}	λ

Table 3: Hyper-parameters of the experiments in Section 5.2.

Description	Value	Variable
Architecture		
depth	9	L
width	20	N_ℓ
Experiment		
spatial samples	{1024, 2048, 4096, 8192, 16382, 32768}	m_d
boundary condition samples	{64, 128, 256, 512, 1024}	m_s
initial condition samples	{64, 128, 256, 512, 1024}	m_t
learning rate	1	
spatial domain	$(-1, 1)$	U
time maximum	1	T

Table 4: Hyper-parameters of the experiments in Section 5.3.

Bibliography

[1] W. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5(4) (1943), pp. 115–133.

[2] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. In: *Psychological Review* 65.6 (1958), pp. 386–408.

[3] A. Friedman. *Partial differential equations of the parabolic type*. prentice hall, 1964.

[4] J. R. Cannon. *The one-dimensional heat equation*. Vol. 23. Cambridge University Press, 1984.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[6] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Math. Control Signal Systems* 2 (1989), pp. 303–314.

[7] K. Hornik, M. Stinchcombe, and H. White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2(5) (1989), pp. 359–366.

[8] D. C. Liu and J. Nocedal. “On the Limited Memory Method for Large Scale Optimization”. In: *Mathematical Programming* 45(3) (1989), pp. 503–528.

[9] E. K. Blum and L. K. Li. “Approximation theory and feedforward networks”. In: *Neural networks* 4(4) (1991), pp. 511–515.

[10] W. Rudin. *Functional analysis*. second. International Series in Pure and Applied Mathematics. New York: McGraw-Hill, Inc., 1991.

[11] M. Leshno et al. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. In: *Neural networks* 6(6) (1993), pp. 861–867.

[12] H. N. Mhaskar. “Approximation properties of a multilayered feedforward artificial neural network”. In: *Adv. Comput. Math.* 1(1) (1993), pp. 61–80.

[13] M. Dissanayake and N. Phan-Thien. “Neural-network-based approximations for solving partial differential equations”. In: *Communications in Numerical Methods in Engineering* (1994).

[14] G. Folland. *Introduction to Partial Differential Equations*. second. Princeton University Press, 1995.

[15] I. Lagaris, A. Likas, and D. Fotiadis. “Artificial neural networks for solving ordinary and partial differential equations”. In: *IEEE Transactions on Neural Networks* 9(5) (1998), pp. 987–1000.

[16] F. Scarselli and A. C. Tsoi. "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results". In: *Neural networks* 11(1) (1998), pp. 15–37.

[17] J. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. second. Vol. 22. Texts in Applied Mathematics. Springer, 1998.

[18] V. Vapnik. *Statistical learning theory*. Vol. 3. New York: Wiley, 1998.

[19] M. Anthony and P. Bartlett. *Neural network learning: theoretical foundations*. Cambridge: Cambridge University Press, 1999.

[20] V. Maiorov and A. Pinkus. "Lower bounds for approximation by MLP neural networks". In: *Neurocomputing* 25(1-3) (1999), pp. 81–91.

[21] N. Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural Networks* 12(1) (1999), pp. 145–151.

[22] R. Ash and C. Doleans-Dade. *Probability and Measure Theory*. Harcourt/Academic Press, 2000.

[23] D. P. Bertsekas and J. N. Tsitsiklis. "Gradient convergence in gradient methods with errors". In: *SIAM Journal on Optimization* 10 (2000), pp. 627–642.

[24] F Cucker and S Smale. "On the Mathematical Foundations of Learning". In: *Bulletin of the American mathematical society* 39 (2002), pp. 1–49.

[25] Y. LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[26] A.N. Iusem. "On the convergence properties of the projected gradient method". In: *Computational Applied Mathematics* 22(1) (2003), pp. 37–52.

[27] T Poggio and S Smale. "The Mathematics of Learning: Dealing with Data". In: *Notice of the American mathematical Society* 50 (2003), pp. 537–544.

[28] M. Renardy and R. Rogers. *An Introduction to Partial Differential Equations*. second. Vol. 13. Texts in Applied Mathematics. Springer, 2004.

[29] K. Morton and D. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. second. Cambridge: Cambridge University Press, 2005.

[30] C. M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. New York: Springer, 2006.

[31] W. Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 2006.

[32] C. Aliprantis and K. Border. *Infinite Dimensional Analysis: A Hitchhiker's Guide*. Springer, 2007.

[33] S. Keller A. Heinrich and H. Niederreiter. *Monte Carlo and Quasi-Monte Carlo Methods*. Springer Berlin Heidelberg, 2007.

[34] L. Evans. *Partial Differential Equations*. Vol. 19. American Mathematical Society, 2010.

[35] H. Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. New York: Springer, 2011.

[36] A. Ruszczynski. *Nonlinear Optimization*. Princeton University Press, 2011.

[37] P. Billingsley. *Probability and Measure*. Wiley Series in Probability and Statistics. Wiley, 2012.

[38] C. Graham and D. Talay. *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2013.

[39] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. New York: Springer, 2013.

[40] O. Shamir and T. Zhang. "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes". In: *International Conference on Machine Learning* (2013), pp. 71–79.

[41] I. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems* 3(11) (2014).

[42] D Kingma and J Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (2014).

[43] A. Klenke. *Probability theory*. second. Universitext. London: Springer, 2014.

[44] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press, 2014.

[45] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[46] P. Cannarsa and D'Aprile. *Introduction to Measure Theory and Functional Analysis*. Unitext. Springer International Publishing, 2015.

[47] Y. LeCun, Y. Bengio, and G. Hinton. "Deep Learning". In: *Nature* 521 (2015), pp. 436–444.

[48] S. Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[49] L. N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *IEEE Winter Conference on Applications of Computer Vision* (2017), pp. 464–472.

[50] L. N. Smith and N. Topin. "Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates". In: *arXiv preprint arXiv:1708.07120* (2017).

[51] K. Eykholt et al. "Robust Physical-World Attacks on Deep Learning Visual Classification". In: *Conference on Computer Vision and Pattern Recognition* (2018), pp. 1625–1634.

[52] A. Jentzen et al. "Strong error analysis for stochastic gradient descent optimization algorithms". In: *arXiv preprint arXiv:1801.09324* (2018).

[53] M. Raissi and G. Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357 (2018), pp. 125–141.

[54] D. Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362(6419) (2018), pp. 1140–1144.

[55] C. Berner et al. "Dota 2 with Large Scale Deep Reinforcement Learning". In: *Preprint, available from arXiv:1912.06680* (2019).

[56] S. Mishra. "A machine learning framework for data driven acceleration of computations of differential equations". In: *Mathematics in Engineering* 1 (2019), pp. 118–146.

[57] M. Raissi, P. Perdikaris, and G. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[58] C. Shorten and T. M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Big Data* 6.60 (2019).

[59] O. Yadan. *Hydra - A framework for elegantly configuring complex applications*. Github. 2019. URL: `https://github.com/facebookresearch/hydra`.

[60] T. Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[61] K. Lye, S. Mishra, and D. Ray. "Deep learning observables in computational fluid dynamics". In: *Journal of Computational Physics* 410 (2020), p. 109339.

[62] Y. Shin, J. Darbon, and G. Karniadakis. "On the Convergence and generalization of Physics Informed Neural Networks". In: *Preprint, available from arXiv:2004.01806v1* (2020).

[63] C. Beck et al. "Artificial neural networks for solving ordinary and partial differential equations". In: *Journal of Scientific Computing* 88(73) (2021).

[64] A. Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations* (2021).

[65] J. Jumper, R. Evans, A. Pritzel, et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596 (2021), pp. 583–589.

[66] A. S. Krishnapriyan et al. "Characterizing possible failure modes in physics-informed neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021).

[67] P. Petersen, M. Raslan, and F. Voigtlaender. "Topological properties of the set of functions generated by neural networks of fixed size". In: *Foundations of Computational Mathematics* 21 (2021), pp. 375–444.

[68] J. Berner, P. Grohs, and Voigtlaender F. "Training ReLU networks to high uniform accuracy is intractable". In: *arXiv preprint arXiv:2205.13531* (2022).

[69] J. Berner et al. "The Modern Mathematics of Deep Learning". In: *Mathematical Aspects of Deep Learning*. Ed. by P. Grohs and G. Kutyniok. Cambridge: Cambridge University Press, 2022, pp. 1–111.

[70] A. Fawzi, M. Balog, et al. "Discovering faster matrix multiplication algorithms with reinforcement learning". In: *Nature* 610 (2022), pp. 47–53.

[71] S. Mishra and R. Molinaro. "Estimates on the generalization error of physics-informed neural networks for approximating PDEs". In: *IMA Journal of Numerical Analysis* 43(1) (2022), pp. 1–43.

[72] R. Rombach et al. "High-Resolution Image Synthesis With Latent Diffusion Models". In: *Conference on Computer Vision and Pattern Recognition* (2022), pp. 10684–10695.

[73] T. Ryck and S. Mishra. "Error analysis for physics-informed neural networks (PINNs) approximating Kolmogorov PDEs". In: *Advances in Computational Mathematics* 48 (2022).

[74] R. Thoppilan et al. "Lamda: Language models for dialog applications". In: *arXiv preprint arXiv:2201.08239* (2022).