universität wien

MASTERARBEIT / MASTER THESIS

Titel der Masterarbeit / Title of the Master Thesis

"Overcoming the curse of dimensionality for deep learning"

verfasst von / submitted by

Clemens Eckl, BSc. BEd.

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 / Master Mathematics
Betreut von / Supervisor:	Ass.-Prof. Dr. Philipp Christian Petersen, MSc.

Abstract (English)

Classical mathematical approximation methods suffer from the "curse of dimensionality". This term is used to describe an exponentially increasing difficulty of problems with increasing dimension.

However, in the beginning of the 1990s it was shown by Barron that approximation with neural networks can, under specific assumptions on the function to be approximated, overcome this curse to some extent. This means that the number of needed parameters (layers and weights) when approximating a d-dimensional function (up to some small error) with neural networks does not grow exponentially with the input dimension. Since then, various authors have stated and rigorously proved different results in this direction.

In this thesis for a nowadays common definition of neural network and its size (measured in the number of layers and non-zero weights) selected results are clustered into four topics. For each of these four areas results are stated and proved showing under which assumptions and to which extent approximation with neural networks overcomes the curse of dimensionality.

In the first area (in Chapter 3) it is shown that neural networks can efficiently approximate hierarchically compositional functions, which can be thought of functions of smooth, lower-dimensional functions.

The second cluster in Chapter 4 handles functions in a high-dimensional space which are assumed to be evaluated only on a lower-dimensional smooth manifold.

Chapter 5 deals with functions whose weighted Fourier transform $\int_{\mathbb{R}^d} \|\omega\|_2 \left|\hat{f}(\omega)\right| d\omega$ is bounded and is based on the results of Barron.

In the last part (Chapter 6) solutions of certain PDEs are shown to be approximable by neural networks without the curse of dimensionality.

Abstract (Deutsch)

Klassische mathematische Approximationsmethoden sind vom sogenannten „Fluch der Dimensionalität“ betroffen. Mit diesem Begriff wird eine exponentiell zunehmende Schwierigkeit des Lösens von Problemen mit zunehmender Dimension beschrieben.

Anfang der 1990er Jahre zeigte Barron jedoch, dass die Approximation mit neuronalen Netzen unter bestimmten Annahmen über die zu approximierende Funktion diesen Fluch einigermaßen überwinden kann. Dies bedeutet, dass die Anzahl der benötigten Parameter (Schichten und Gewichte) bei der Approximation einer d-dimensionalen Funktion (abgesehen von einer kleinen Ungenauigkeit) mit neuronalen Netzen nicht exponentiell mit der Dimension des Defintionsbereichs wächst. Seitdem haben verschiedene Autoren unterschiedliche Ergebnisse in diese Richtung gezeigt und rigoros bewiesen.

In dieser Arbeit werden ausgewählte Ergebnisse für eine heutzutage übliche Definition eines neuronalen Netzwerks und seiner Größe (gemessen in der Anzahl der Schichten und Gewichte ungleich 0) in vier Themenbereiche gruppiert. Für jeden dieser vier Bereiche werden Ergebnisse dargelegt und bewiesen, die zeigen, unter welchen Annahmen und in welchem Ausmaß die Approximation mit neuronalen Netzen den Fluch der Dimensionalität überwinden kann.

Im ersten Bereich (in Kapitel 3) wird gezeigt, dass neuronale Netze hierarchisch zusammengesetzte Funktionen, die man sich als Funktionen glatter, niedrigdimensionaler Funktionen vorstellen kann, effizient approximieren können.

Der zweite Bereich (Kapitel 4) behandelt Funktionen in einem hochdimensionalen Raum, von denen angenommen wird, dass sie nur auf einer niedrigerdimensionalen, glatten Mannigfaltigkeit ausgewertet werden.

Kapitel 5 befasst sich mit Funktionen, deren gewichtete Fourier-Transformation

$$\int_{\mathbb{R}^d} \|\omega\|_2 \left|\hat{f}(\omega)\right| d\omega$$

beschränkt ist und basiert auf den Ergebnissen von Barron.

Im letzten Teil (Kapitel 6) wird gezeigt, dass Lösungen bestimmter partieller Differentialgleichungen durch neuronale Netze annäherbar sind, ohne dass dabei der Fluch der Dimensionalität auftritt.

Contents

1 Introduction

In this thesis a mathematical structure called neural network is studied. These networks are the basis for deep learning algorithms, which are part of the area of machine learning (ML) and artificial intelligence (AI), see [29]. As the name indicates a neural network consists of several neurons which are linked with each other in a certain way.

Historically, neural networks were introduced to model the functionality of the brain. In the 1940s McCulloch and Pitts [52] modelled a biological neuron for $x \in \mathbb{R}^d$ ($d \in \mathbb{N}$) - and given a_i (weights), b (threshold) $\in \mathbb{R}$ where $1 \leq i \leq d$ and $1_{\mathbb{R}^+} : \mathbb{R} \to \mathbb{R}$, (activation function) where $1_{\mathbb{R}^+}(x) = 0$ for $x < 0$ and $1_{\mathbb{R}^+}(x) = 1$ else - as the function

$$x \mapsto \mathbf{1}_{\mathbb{R}^+}\Big(\sum_{i=1}^{d} a_i x_i - b\Big). \tag{1.1}$$

If the combined weighted input reaches or exceeds b the neuron returns 1, one says it "fires". Otherwise the neuron returns 0 and remains inactive. By linking multiple neurons together in the sense that the output of one neuron acts as (part of the) input to another neuron a network of neurons – a "neural network" – is constructed, see e.g. the "multilayer perceptron" (MLP) from Rosenblatt in the 1950s [72]. In mathematical terms, this MLP with L layers is an alternating composition of affine linear maps and the (typically non-linear and throughout the network identical) activation function, so

$$x \mapsto T_L(\rho(T_{L-1}(\rho \dots (\rho(T_1(x))) \dots))), \tag{1.2}$$

where $T_\ell = A_\ell(\cdot) + b_\ell$ denotes an affine linear map from $\mathbb{R}^{d_{\ell-1}}$ to $\mathbb{R}^{d_\ell}$ for $1 \leq \ell \leq L$, $d_0 = d, d_1, \dots, d_L \in \mathbb{N}$ and ρ is applied component-wise. Note that the ρ above can be, but is not necessary, the function $\mathbf{1}_{\mathbb{R}^+}$ as defined for the biological neuron (1.1) above. Nowadays the rectified linear unit (short: ReLU) defined as $\rho(x) := \max\{0, x\}$ is used often. For the remaining part of the introduction the terms "MLP" and "neural network" are used ambiguously, however from Chapter 2 on there is a clear distinction between the neural network (as sequence of weight matrices A_ℓ and bias vectors b_ℓ in the affine linear maps T_ℓ for $1 \leq \ell \leq L$) and its realization function as defined in (1.2).

For compact $K \subset \mathbb{R}^d$ MLPs with two layers are dense in $C(K)$ with respect to the uniform norm $\|f\|_\infty = \sup_{x \in K} |f(x)|$, if the activation function ρ is continuous and not a polynomial [49]. This holds especially for the ReLU. MLPs with this property are called "universal".

Recently, especially deep neural networks (so those with many layers), have received increased attention, since they have shown to be able to be trained efficiently. This means that for given training data in the form of input and output pairs, there exist highly

efficient training algorithms that adapt a network in a way that the trained network interpolates the training data well. Moreover, it even generalizes well to new data points for many problems that occur in practice [64]. This training procedure is typically referred as "deep learning" [46].

Among others, deep learning algorithms are used for image classification (see e.g. [44], where in some cases they even outperform humans in image labelling tasks [33]. Another famous application is the area of (board or computer) games [81, 58], where also world-class humans are beaten [6]. Moreover, deep-learning algorithms are heavily used and are very popular in the field of natural language processing, which includes the subtasks of summarizing, understanding or generating text [94]. The currently probably most prominent example for this kind of tasks is ChatGPT, a variant of the GPT-3 model, trained by OpenAI and specifically designed for chatbot applications [11, 35], which is based on a so-called transformer neural network [88].

While neural networks trained by deep learning prove to be powerful, it is currently not entirely understood why these methods work so well. Therefore, it is reasonable to search for purely mathematical arguments why and under which conditions/assumptions a neural networks is an adequate architecture in practice [64].

In this thesis, the focus is on the approximation property of neural networks for high-dimensional functions. Stated simply, it is studied, how complex neural networks (measured in terms of layers and number of non-zero entries in the matrices A_ℓ and b_ℓ $(1 \leq \ell \leq L)$ in (1.2)) need to be in order to approximate certain functions well. Henceforth, algorithmic aspects of deep learning will **not** be covered in this thesis.

In approximation theory it is well-known that for various types of functions (e.g. smooth functions, piecewise constant functions or functions in L^p for $1 \leq p \leq \infty$) approximation methods with continuous parameter dependence (e.g. approximation with multivariate B-splines or linear finite elements, but also approximation with neural networks) suffer from the so-called curse of dimensionality. This term, which was introduced by Bellman, is used to describe an exponentially increasing difficulty of problems with increasing dimension [5, 19, 61].

Simplified for approximation problems this curse can be stated as follows: To achieve an error (without specifying here how this error is measured, so without stating a norm) of at most $\varepsilon > 0$ for an approximation problem with d-dimensional input and 1-dimensional output $\mathcal{O}(\varepsilon^{-d})$ parameters are needed. Typically smooth functions (so s-times differentiable for some $s \in \mathbb{N} \cup \{\infty\}$) or piecewise constant functions (for classification problems) are considered. It is well known that smoothness of the function to approximate reduces the needed parameters. To be more precise, for s-times differentiable objective $\mathcal{O}(\varepsilon^{-d/s})$ parameters are needed [61].

Among others, in Chapter 2 it is shown that B-spline and linear finite element approximation (which both suffer from the curse of dimensionality) can be also realized by ReLU neural networks with the same approximation rates, see [53, 84] and [34], respectively. In the course of proving the emulation of B-splines with ReLU neural networks specific operations and statements for ReLU neural networks, which are needed and used later on

in various chapters, are given. Beforehand, neural networks, their realization functions, their size and operations with neural networks are defined as in [64].

However, in comparison to other methods and under additional assumptions on the function to be approximated neural networks are shown to overcome the curse of dimensionality, meaning that there is no exponential dependence of the number of needed layers and non-zero weights on the input dimension. In this thesis, various assumptions and resulting statements, how to provably overcome this curse with neural networks, are given. These assumptions are clustered into four parts (Chapters 3, 4, 5 and 6) and are briefly described underneath.

In Chapter 3, functions of lower-dimensional differentiable functions, which are called hierarchically compositional functions, are approximated by ReLU neural networks without the curse dimensionality [70]. This is based on the fact that neural networks by definition are also hierarchical and compositional and can therefore make use of this special structure of these functions, whereas classical approximation methods are somehow blind to it. Each of the appearing lower-dimensional smooth functions can be approximated by ReLU neural networks up to some small error ε (resembling B-spline approximation with ReLU neural networks as shown in Chapter 2) with number of layers and weights depending exponentially on the dimension of the underlying space. Moreover, the error is shown to grow only linearly in ε from one function evaluation to the next and the total needed weights are of order $\mathcal{O}(\varepsilon^{-k})$, where k is the maximum of the input dimensions of the lower-dimensional functions. Hence the number of needed weights does not grow exponentially in the input dimension.

Additionally, it is stated why it is reasonable to assume that (non-smooth) classification functions are of compositional structure and that under this assumption one can approximate functions of this type without the curse of dimensionality with ReLU neural networks.

Chapter 4 shows that functions which are defined in a high-dimensional space, but are assumed to be evaluated only on a lower-dimensional compact, smooth manifold $\mathcal{M}$, can be approximated by ReLU neural networks, where the number of needed layers and weights of these networks grow exponentially only in the dimension of the manifold and not in the dimension of the underlying space [13].

By assumption $\mathcal{M}$ is compact and smooth and can therefore be covered by $n \in \mathbb{N}$ many sets U_i, where $1 \leq i \leq n$. Considering for $1 \leq i \leq n$ P_i as the projection from $U_i \cap \mathcal{M}$ to the respective tangent space and using a partition of unity $(\phi_i)_{i=1}^n$ with $\operatorname{supp}(\phi_i) \subset U_i$ (resembling the finite element method with ReLU neural networks as shown in Chapter 2) the function $f : \mathcal{M} \to \mathbb{R}$ can be written as

$$f(x) = \sum_{i=1}^{n} \phi_i(x) \cdot \big(\underbrace{f(x) \circ P_i^{-1}}_{=:f_i} \circ P_i(x) \big). \tag{1.3}$$

For $1 \leq i \leq n$, the P_i can be realized by ReLU neural networks as they are affine linear maps and ϕ_i can be realized by ReLU neural networks as they are linear finite

elements. Additionally, for $1 \leq i \leq n$ the f_i can be approximated using the resembled B-spline approximation with ReLU neural networks on the lower-dimensional tangent space. Hence, using ReLU calculus shown in Chapter 2 the right-hand side of equation (1.3) can be approximated by ReLU neural networks without the curse of dimensionality.

The statements of Chapter 5 yield that functions, whose weighted Fourier transform $\int_{\mathbb{R}^d} \|\omega\|_2 \left|\hat{f}(\omega)\right| d\omega$ is bounded by some $C > 0$, can be approximated by neural networks with sigmoidal activation function and one hidden layer with N neurons on the compact set B_1^d (the d-dimensional unit ball around 0) to an error measured in the L^2-norm up to a fixed multiple of $\frac{1}{N}$ [3].

The main proof in Section 5.1 is based on a random sampling idea (sometimes referred as "Approximate Caratheodory Theorem"), which states that an element f in the closure of the convex hull of G (subset of a Hilbert space H with elements bounded by a constant $B > 0$) can for any $N \in \mathbb{N}$ be approximated up to $\frac{B^2}{N}$ by a convex combination of N elements in G in the squared norm induced by the inner product of H.

In the language of neural networks: Each function that can be represented by an arbitrary wide two-layer neural network (so with an arbitrary number of neurons in the first layer) with bounded activation function and where the weights in the last layer are positive and sum up to one can also be approximated with a neural networks with only N neurons in the first layer and an approximation rate of $\frac{1}{N}$.

Henceforth, it is shown that f with weighted Fourier transform as defined above bounded by C fulfills $f_{|B_1^d} - f(0) \in \overline{\text{co}}(G_C)$, where $G_C := \{g : B_1^d \to \mathbb{R} : g(x) = \gamma \mathbf{1}_{\mathbb{R}^+}(\langle a, x\rangle + b), a \in \mathbb{R}^d, b, \gamma \in \mathbb{R}, |\gamma| \leq 2C\}$, $\mathbf{1}_{\mathbb{R}^+}$ denotes the indication function on $\mathbb{R}^+ := [0, \infty)$ and the closure is taken with respect to the L^2-norm on B_1^d.

Combining these two results and using the dominated convergence theorem yields the stated claim.

Additionally, in Section 5.2 extensions of what is stated above are sketched. Content-wise these extensions (two generalizations of what is described above, the relation between these two generalizations and an application) are parts of [12, 21].

Lastly, in Chapter 6, solutions of PDEs were shown to be approximable by ReLU neural networks overcoming the curse of dimensionality.

In Section 6.1 the (hyperbolic) homogeneous parametric linear transport equation (where $d, d', s, k \in \mathbb{N}$ and $T > 0$)

$$\begin{cases} \partial_t u(t,x,\mu) + V(t,x,\mu) \cdot \nabla_x u(t,x,\mu) = 0, \\ u(0,x,\mu) = u_0(x) \end{cases} \tag{1.4}$$

for given vector field $V \in C^k([0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \mathbb{R}^d)$ fulfilling a growth condition in the C^k-norm and initial condition $u_0 \in C^s(\mathbb{R}^d)$ where $t \in [0,T]$, $x \in \mathbb{R}^d$ and $\mu \in [0,1]^{d'}$) is studied. In [28] the PDE (1.4) is shown to have an unique solution of the form $u(t,x,\mu) = u_0(X(0,t,x,\mu)) \in C^{\min\{s,k\}}$, where $X(s,t,x,\mu) := \gamma(s) \in C^k$ and where γ is

the solution of the characteristic systems of ordinary differential equations

$$\begin{cases} \dot{\gamma}(s) = V(s, \gamma(s), \mu), \\ \gamma(t) = x. \end{cases}$$

The unique solution u is then approximated by approximating u_0 and X by using the resembling of B-spline approximation with ReLU neural networks from Chapter 2, respectively, and using the output of the network approximating X as input for the one approximating u_0 [45]. No exponential dependence on the input dimension appears as the method of characteristics doesn't suffer from the curse of dimensionality, which is resembled here.

Similar statements are sketched for the non-homogeneous parametric linear transport equation, so with right-hand side $f(t, x, \mu) \in C^{s'}$ for $s' \in \mathbb{N}$ in (1.4). The unique solution $u \in C^{\min\{s,s',k\}}$ is then of the form $u(t, x, \mu) = u_0(X(0, t, x, \mu)) + \int_0^t f(s, X(s, t, x, \mu), \mu) ds$, so additionally the integral has to be approximated using ReLU neural networks [45].

Section 6.2 studies the linear Kolmogorov PDE with affine linear diffusion and drift, so

$$\begin{cases} \partial_t u(t, x) = \frac{1}{2} tr\left(\sigma(x)\sigma(x)^T \nabla_x^2 u(t, x)\right) + \langle \mu(x), \nabla_x u(t, x) \rangle \\ u(0, x,) = u_0(x), \end{cases} \tag{1.5}$$

for $t \in [0, T]$ and $x \in \mathbb{R}^d$, where ∇_x and ∇_x^2 denote the gradient and Hessian with respect to x, respectively and where $d \in \mathbb{N}$ and $T > 0$ and initial value $u_0 \in C(\mathbb{R}^d, \mathbb{R})$, $\mu \in C(\mathbb{R}^d, \mathbb{R}^d)$ (drift coefficeint) and $\sigma \in C(\mathbb{R}^d, \mathbb{R}^{d \times d})$ (diffusion coeffcient) are given. It is stated that there exists a unique so-called viscosity solution for (1.5) and that if u_0 can be approximated well by a ReLU neural network, also the unique viscosity solution can be approximated on a cube $[u, v]^d$ with respect to the squared L^2-norm on $[u, v]^d$ scaled by $(v - u)^{-d}$ up to $\varepsilon > 0$ without the curse of dimensionality [8, 30]. The proof is based on the Feynman-Kac formula (which establishes a link between parabolic PDEs and stochastic processes) and, as in Chapter 5, a random sampling approach.

As a special case, for the Black-Scholes equation in option pricing, where $u_0(x) = \min\{\max\{D - y^T x, 0\}, D\}$ for given $D > 0$ and $y \in \mathbb{R}^d$, is treated.

2 Neural network approximation and the curse of dimensionality

In this chapter neural networks and their realisation are defined.

Additionally, it is shown, how neural networks can be connected with each other and that ReLU neural networks are as good as B-splines in the task of approximating smooth functions.

Moreover, ReLU neural networks are shown to be able to exactly emulate the linear finite element method.

Lastly, the in the introduction mentioned curse of dimensionality is explicitly stated.

2.1 Definitions of neural networks and operations with neural networks

The definitions given in this chapter are stated mostly as in [64], where this formalism was developed. Starting with the definition of the objects in scope of this thesis: neural networks, their size and for given activation function their realisation.

Definition 2.1.1. Let $d, L \in \mathbb{N}$. A neural network Φ with input dimension d and L layers is a sequence of matrix vector tuples

$$\Phi = \big((A_1, b_1), (A_2, b_2), \ldots, (A_L, b_L)\big)$$

where $N_0 := d$, $N_1, \ldots, N_L \in \mathbb{N}$, $A_\ell \in \mathbb{R}^{N_\ell \times \mathbb{N}_{\ell-1}}$ and $b_\ell \in \mathbb{R}^{N_\ell}$ for $\ell = 1, \ldots, L$. Moreover, N_L is called output dimension of a neural network Φ and the vector $A(\Phi) := (d, N_1, \ldots, N_L) \in \mathbb{N}^{L+1}$ is called architecture of Φ.

For a neural network Φ and a given activation function $\rho : \mathbb{R} \to \mathbb{R}$ the realisation of the neural network Φ is defined as

$$R(\Phi) : \mathbb{R}^d \to \mathbb{R}^{N_L} : x \mapsto x_L := R(\Phi)(x),$$

where the output $x_L \in \mathbb{R}^{N_L}$ results from

$$\begin{aligned} x_0 &:= x, \\ x_\ell &:= \rho(A_\ell x_{\ell-1} + b_\ell) \quad \text{for } \ell = 1, \ldots, L-1, \\ x_L &:= A_L x_{L-1} + b_L. \end{aligned}$$

The activation function ρ is understood to act component-wise.

In order to make precise statements on the size of a given neural network Φ it is distinguished between the number of neurons $N(\Phi)$, number of layers (also called depth) $L(\Phi)$ and number of weights $M(\Phi)$ given as

$$\begin{aligned} N(\Phi) &:= d + \sum_{\ell=1}^{L} N_\ell, \\ L(\Phi) &:= L \quad \text{and} \\ M(\Phi) &:= \sum_{\ell=1}^{L} \Big(\|A_\ell\|_0 + \|b_\ell\|_0 \Big) \leq \sum_{\ell=1}^{L} (N_\ell \cdot N_{\ell-1} + N_\ell), \end{aligned}$$

where $\|\cdot\|_0$ denotes the number of non-zero entries of a matrix or vector.
The layers in a neural network, which are not the input or ouput layer are called hidden layers, the number of hidden layers of a neural network Φ is therefore $L(\Phi) - 1$. Networks with one hidden layer, so $L(\Phi) = 2$, are called shallow neural networks and the realizations of such networks have the form

$$x \mapsto \sum_{i=1}^{N} \big(c_i \rho(\langle a_i, x \rangle + b_i) \big) + e,$$

where $N := N_1 \in \mathbb{N}$ is the number of neurons in the hidden layer and $b_i, c_i, e \in \mathbb{R}$ and $a_i \in \mathbb{R}^d$ for $1 \leq i \leq N$. Neural networks with more than one hidden layer are called deep neural networks. The illustration of a (deep) neural network is given in Figure 2.1.

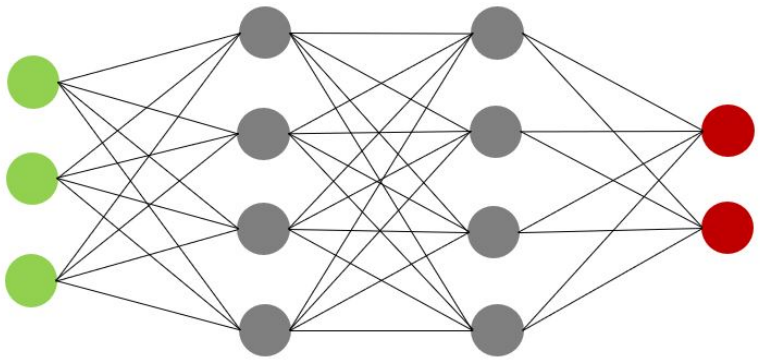

Figure 2.1: Illustration of a neural network Φ with architecture $A(\Phi) = (3, 4, 4, 2)$, so with 3 layers, 3 input neurons (green), 4 neurons in the 2 hidden layers, respectively (gray) and 2 output neurons (red). Note that only at the neurons in the hidden layers the activation function is applied.

In practice a specific, quite simple, piecewise linear activation function, the so-called ReLU, is used very often.

Definition 2.1.2. The function $\rho : \mathbb{R} \to \mathbb{R}$ with

$$\rho(x) := (x)_+ = \max\{0, x\} = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{else} \end{cases}$$

is called rectified linear unit (short: ReLU).

The ReLU fulfills the property

$$x = \rho(x) - \rho(-x), \tag{2.1}$$

so the identity function can be rebuild using the ReLU, compare to Lemma 2.3.5 in Subsection 2.3.1. Clearly the ReLU is idempotent, since $\rho(\rho(x) = \rho(x)$ and the realisation of a ReLU neural network is continuous and piecewise affine linear, see e.g. [18]. Moreover, apart from 0, where the ReLU is not differentiable, the derivative is given by

$$\rho'(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x < 0 \end{cases},$$

so in comparison to other often used activation functions like the logistic sigmoid function $\gamma(x) = \frac{1}{1+e^{-x}}$ or the hyperbolic tangent function $\tanh x$, the ReLU is not saturated and does therefore not suffer from vanishing gradients, compare to Section 3.10 in [29]. Lastly, it is, compared to the two functions mentioned above, very fast to compute, as no exponential function is involved. A smoothened version of the ReLU is given via the softplus function

$$\mu_a = \frac{\log(1 + e^{ax})}{a}$$

for $a > 0$, compare to [32]. For an overview and comparison of different activation functions see [26, 9].

Remark 2.1.3. Comments to (ReLU) neural networks.

- In the literature, mostly in articles, there often is no clear distinction between a neural network Φ and its realisation $R(\Phi)$, both are called "neural network". Typically, it is clear from the context, what is meant in the current situation. However, at least in the Definitions, Lemmata, Propositions etc. in this thesis, there is distinguished between the sequence of matrix vector tuples Φ and the function $R(\Phi)$ in order to formulate precise statements.

- Each neural network uniquely induces a realisation function. However, in general there can be multiple non-trivially different neural networks which induce the same realisation function. This can be seen for example by looking at the neural network $\Phi = \big((A_1, b_1), \ldots, (A_{L-1}, b_{L-1}), (0, b_L)\big)$. For every activation function and every choice of (A_ℓ, b_ℓ) for $1 \leq \ell \leq L - 1$ it holds that $R(\Phi) = b_L$. A deeper insight and analysis in this respect for ReLU neural networks refer to [7].

Prior to starting the analysis of approximation of functions with (realisations of - compare to Remark 2.1.3 above) neural networks, elementary operations that one can perform with given neural networks are defined. These operations are quite natural and it is their meaning to "link" given neural networks in different ways.

Definition 2.1.4. Let $L_1, L_2 \in \mathbb{N}$ and let

$$\Phi^1 = \Big((A_1^1, b_1^1), \dots, (A_{L_1}^1, b_{L_1}^1)\Big) \quad \text{and} \quad \Phi^2 = \Big((A_1^2, b_1^2), \dots, (A_{L_2}^2, b_{L_2}^2)\Big)$$

be two neural networks such that the input layer of Φ^1 has the same dimension as the output layer of Φ^2. Then the concatenation of Φ^1 and Φ^2, denoted by $\Phi^1 \bullet \Phi^2$, is the following $(L_1 + L_2 - 1)$-layered neural network $\Phi^1 \bullet \Phi^2 :=$

$$\Big((A_1^2, b_1^2), \dots, (A_{L_2-1}^2, b_{L_2-1}^2), (A_1^1 A_{L_2}^2, A_1^1 b_{L_2}^2 + b_1^1), (A_2^1, b_2^1), \dots, (A_{L_1}^1, b_{L_1}^1)\Big). \tag{2.2}$$

It holds that

$$R(\Phi^1 \bullet \Phi^2) = R(\Phi^1) \circ R(\Phi^2), \tag{2.3}$$

where $\circ$ denotes the composition of two functions. For an illustration of the concatenation of two neural networks see Figure 2.2.

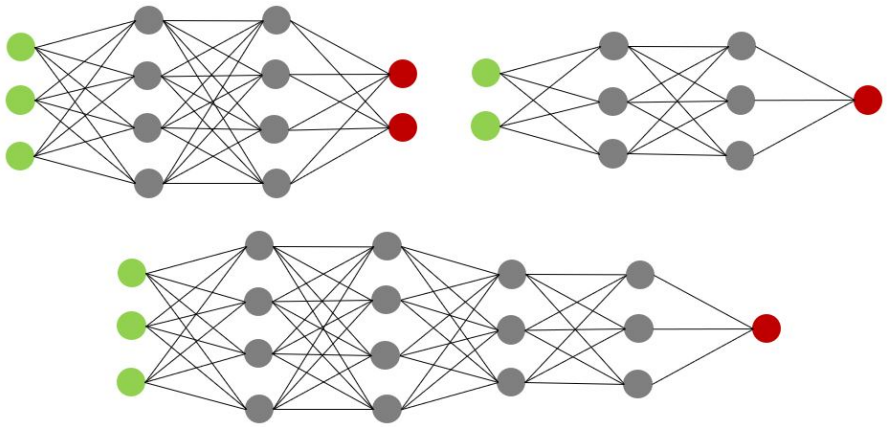

Figure 2.2: **Top:** Two neural networks Φ^1 (right) and Φ^2 (left), where the input layer of Φ^1 has the same dimension as the output layer of Φ^2.
Bottom: The concatenation of Φ^1 and Φ^2 according to Definition 2.1.4.

Definition 2.1.5. Let $L, d_1, d_2 \in N$ and let

$$\Phi^1 = \Big((A_1^1, b_1^1), \dots, (A_L^1, b_L^1)\Big) \quad \text{and} \quad \Phi^2 = \Big((A_1^2, b_1^2), \dots, (A_L^2, b_L^2)\Big)$$

be two neural networks with L layers and with d_1-dimensional and d_2-dimension input, respectively. Moreover, let

$$\widehat{A_1} := \begin{pmatrix} A_1^1 \\ A_1^2 \end{pmatrix}, \widehat{b_1} := \begin{pmatrix} b_1^1 \\ b_1^2 \end{pmatrix}, \widetilde{A_\ell} := \begin{pmatrix} A_\ell^1 & 0 \\ 0 & A_\ell^2 \end{pmatrix} \text{ and } \widetilde{b_\ell} := \begin{pmatrix} b_\ell^1 \\ b_\ell^2 \end{pmatrix} \text{ for } 1 \leq \ell \leq L.$$

For $d_1 = d_2 =: d$ the parallelisation with shared inputs of Φ^1 and Φ^2, denoted by $P(\Phi^1, \Phi^2)$, is a neural network with L layers and d-dimensional input defined as

$$P(\Phi^1, \Phi^2) := \left(\left(\widehat{A_1}, \widehat{b_1}\right), \left(\widetilde{A_2}, \widetilde{b_2}\right), \dots \left(\widetilde{A_L}, \widetilde{b_L}\right) \right).$$

For arbitrary $d_1, d_2 \in \mathbb{N}$ the parallelisation without shared inputs (also called full parallelisation) of Φ^1 and Φ^2, denoted by $FP(\Phi^1, \Phi^2)$, is a neural network with L layers and $d_1 + d_2$-dimensional input defined as

$$FP(\Phi^1, \Phi^2) := \left(\left(\widetilde{A_1}, \widetilde{b_1}\right), \dots \left(\widetilde{A_L}, \widetilde{b_L}\right) \right).$$

The parallelisation with and without shared inputs of two neural networks is visualised in Figure 2.3 and it holds that

$$\begin{aligned} M(P(\Phi^1, \Phi^2)) &= M(FP(\Phi^1, \Phi^2)) = M(\Phi^1) + M(\Phi^2), \\ R(P(\Phi^1, \Phi^2))(x) &= (R(\Phi^1)(x), R(\Phi^2)(x)) \quad \forall\, x \in \mathbb{R}^d \quad \text{and} \\ R(FP(\Phi^1, \Phi^2))(x_1, x_2) &= (R(\Phi^1)(x_1), R(\Phi^2)(x_2)) \quad \forall\, x_1 \in \mathbb{R}^{d_1}, \forall\, x_2 \in \mathbb{R}^{d_2}. \end{aligned} \tag{2.4}$$

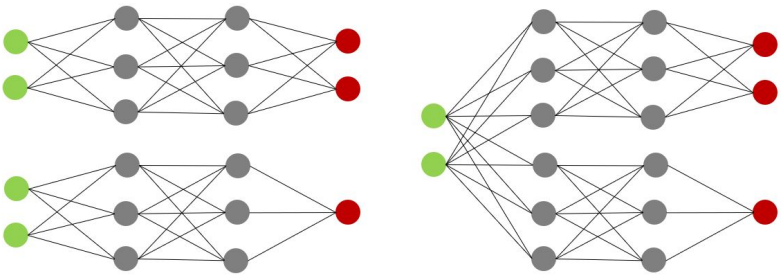

Figure 2.3: **Left:** Two neural networks Φ^1 (top) and Φ^2 (bottom) with the same depth and input dimension.
Right: The parallelisation with shared inputs of Φ^1 and Φ^2 according to Definition 2.1.5.
The parallelisation without shared inputs of Φ^1 and Φ^2 can be interpreted as the neural network on the left side consisting of Φ^1 *and* Φ^2, so without any connections between these two neural networks.

2.2 Reapproximation of dictionaries

In this section the procedure of lifting theoretical approximation results (which **do** suffer from the curse of dimensionality) for N-term approximation to theoretical results for neural networks is shown, compare to Section 2.4 in [66]. In a nutshell, it will be seen that if elements of a dictionary can be approximated well with neural networks, also elements which can be approximated by a linear combinations of N elements out of this dictionary can be approximated well with neural networks, with cN weights for a constant c. This seems reasonable and not really surprising, as (affine) linear maps can be easily realized with 1-layered neural networks. In order to proceed the definitions of N-term approximation and the worst case error, a measurement for the error made using N-term approximation, are given.

Definition 2.2.1. Let H be a normed space, $(A_N)_{N\in\mathbb{N}}$ a nested sequence (i.e. $A_N \subset A_{N+1}$ for every $N \in \mathbb{N}$) of subsets of H and $C \subset H$. Then for $N \in \mathbb{N}$ the worst case error when approximating every element of C by the closest element in A_N, denoted by $\sigma(A_N, C)$, is defined as

$$\sigma(A_N, C) := \sup_{f\in C} \inf_{g\in A_N} \|f - g\|_H .$$

If it is not possible to compute $\sigma(A_N, C)$ precisely, but it is able to write

$$\sigma(A_N, C) = \mathcal{O}(h(N)) \text{ for } N \to \infty \tag{2.5}$$

for some $h : \mathbb{N} \to \mathbb{R}$, then $(A_N)_{N\in\mathbb{N}}$ is said to achieve an approximation rate of h for C.

A natural example of nested sequences are sparse representations in a basis or a dictionary.

Definition 2.2.2. Let $D := (f_i)_{i=1}^{\infty}$ be a dictionary (assuming that D contains only countably many elements) and let

$$A_N := \left\{ \sum_{i=1}^{\infty} a_i f_i : \|a\|_0 \leq N \right\}. \tag{2.6}$$

With A_N like in (2.6), $\sigma(A_N, C)$ is called the best N-term approximation error of C with respect to D. Here $\|a\|_0$ is the cardinality of $\{i \in \mathbb{N} : a_i \neq 0\}$. Moreover, if h satisfies (2.5) D is said to achieve a rate of best N-term approximation error of h for C.

Now all the needed tools and terms are available to formulate the statement mentioned in the beginning of this section.

Theorem 2.2.3. *Let $d \in \mathbb{N}, H \subseteq \{f : \mathbb{R}^d \to \mathbb{R}\}$ be a normed space, $\rho : \mathbb{R} \to \mathbb{R}$ and $D = (f_i)_{i=1}^{\infty}$ be a dictionary. Assume that there exist $L, c \in \mathbb{N}$ such that, for every $i \in \mathbb{N}$ and every $\varepsilon > 0$, there exists a neural network Φ_i^ε such that*

$$L(\Phi_i^\varepsilon) = L, \quad M(\Phi_i^\varepsilon) \leq c \text{ and } \|R(\Phi_i^\varepsilon) - f_i\|_H \leq \varepsilon. \tag{2.7}$$

For every $C \subset H$ define A_N as is (2.6) *and define*

$$B_N := \{R(\Phi) : \Phi \text{ is a neural network with } L(\Phi) = L \text{ and } M(\Phi) \leq N\}.$$

Then, for every $C \subset H$, it holds that $\sigma(B_{cN}, C) \leq \sigma(A_N, C)$.

Proof. Let $a \in A_N$, so $a = \sum_{j=1}^{N} c_{i(j)} f_{i(j)}$ and $\varepsilon > 0$. Then, by (2.7), it holds that there exist neural networks $(\Phi_j)_{j=1}^{N}$ such that

$$L(\Phi_j) = L, \quad M(\Phi_j) \leq c \quad \text{and} \quad \|R(\Phi_j) - f\|_H \leq \frac{\varepsilon}{N\hat{c}}, \text{ where } \hat{c} = \max_{j=1,\dots,N} |c_j|. \quad (2.8)$$

With $\Phi^c := (([c_i(1), c_i(2), \dots, c_i(N)], 0))$ and $\Phi^{a,\varepsilon} := \Phi^c \bullet P(\Phi_1, \Phi_2, \dots, \Phi_N)$ one yields, using the triangle inequality and the approximation property in (2.8),

$$\|R(\Phi^{a,\varepsilon}) - a\|_H = \left\| \sum_{j=1}^{N} c_{i(j)} (f_{i(j)} - R(\Phi_j)) \right\|_H \leq \sum_{j=1}^{N} |c_{i(j)}| \left\| f_{i(j)} - R(\Phi_j) \right\|_H \leq N\hat{c} \frac{\varepsilon}{N\hat{c}} = \varepsilon.$$

Clearly, due to (2.8) and Definition 2.1.4,

$$L(\Phi^{a,\varepsilon}) = L(\Phi^c \bullet P(\Phi_1, \Phi_2, \dots, \Phi_N) = 1 + L(P(\Phi_1, \Phi_2, \dots, \Phi_N) - 1 = L$$

and, due to (2.8), Definition 2.1.4 and (2.4)

$$\begin{aligned} M(\Phi^{a,\varepsilon}) = M(\Phi^c \bullet P(\Phi_1, \Phi_2, \dots, \Phi_N)) &= M(P(\Phi_1, \Phi_2, \dots, \Phi_N)) \\ &\leq N \max_{j=1,\dots,N} M(\Phi_j) = Nc, \end{aligned}$$

which proofs the claim. □

Remark 2.2.4. For the moment it is not clear, why there should be dictionaries and neural networks/activation functions such that condition (2.7) holds for all elements in the underlying dictionary.

However, in Section 2.3 it is shown that for smooth functions on a nonempty and bounded set assumption (2.7) holds for the dictionary of B-splines and ReLU neural networks. Moreover, (2.7) holds for the dictionary of affine linear functions on a locally convex mesh, where each function in the dictionary can be written as realisation of a ReLU neural network, used for the finite element method, see Subsection 2.3.2.

Various other combinations of dictionaries and activation functions for which (2.7) holds can be found on p.27 in [71].

2.3 Function approximation with ReLU neural networks

In this section it will be shown that assumption (2.7) in Theorem 2.2.3 holds for for the dictionary of multivariate B-splines for ReLU neural networks, see Subsection 2.3.1. This was first shown in Lemma 1 in [84] and is presented here as a summarized version of [66].

Together with a result about the best N-term approximation with multivariate B-splines (see [63]) quantitative approximation results for neural networks, which show the curse of dimensionality, are derived. Additionally in this section, ReLU specific network operations, which will be used in several proofs in upcoming chapters, will be defined and some properties of them are shown.

In Subsection 2.3.2 it is shown how the linear finite element method can be realized by using ReLU neural networks, based on [34].

2.3.1 Emulation of B-Spline approximation and ReLU calculus

Definition 2.3.1. The univariate cardinal B-spline on $[0, k]$ of order $k \in \mathbb{N}$ is defined as

$$N_k(x) := \frac{1}{(k-1)!} \sum_{\ell=0}^{k} (-1)^\ell \binom{k}{\ell} (x-\ell)_+^{k-1}, \tag{2.9}$$

with $(x)_+ := \max\{0, x\}$ and the convention $0^0 = 0$.

For $t \in \mathbb{R}$ and $\ell \in \mathbb{N}$ defining $N_{\ell,t,k} := N_k(2^\ell(\dot{-}t))$, so shifted and scaled splines. For $d \in \mathbb{N}, \ell \in \mathbb{N}$ and $t = (t_1, \ldots, t_d) \in \mathbb{R}^d$ the multivariate B-splines is defined by

$$N^d_{\ell,t,k}(x) := \prod_{i=1}^{d} N_{\ell,t_i,k}(x_i) \quad \text{for } x = (x_1, \ldots, x_d) \in \mathbb{R}^d. \tag{2.10}$$

For $d \in \mathbb{N}$ the dictionary of dyadic B-splines of order k is given by

$$B^k := \{N^d_{\ell,t_\ell,k} \,|\, \ell \in \mathbb{N}, t_\ell \in 2^{-\ell}\mathbb{Z}^d\}. \tag{2.11}$$

Clearly $\operatorname{supp} N_k = [0, k]$, and one can show that $\|N_k\|_\infty \leq 1$, see [78].

The following (simplified version of a) statement about best N-term approximation with multivariate B-Splines from Oswald, published in [63], is stated without proof. The given domain there is $[0, 1]^d$ for some $d \in N$, but the statement also holds for any other given cube in $\mathbb{R}^d$ or more general, for any nonempty and bounded set in $\mathbb{R}^d$, compare to Remark 2.3.12 underneath.

Theorem 2.3.2. *Let $d, k \in \mathbb{N}$, $p \in [1, \infty]$ and $0 < s \leq k$. Then exists a constant $C > 0$ such that for every $f \in C^s([0, 1]^d)$, every $\delta > 0$ and every $N \in N$ there exists $c_i \in \mathbb{R}$ with $|c_i| \leq C \|f\|_\infty$ and $B_i \in B^k$ for $i = 1, \ldots, N$ such that*

$$\left\| f - \sum_{i=1}^{N} c_i B_i \right\|_{L^p([0,1]^d)} \leq C N^{\frac{\delta - s}{d}} \|f\|_{C^s([0,1]^d)}.$$

In particular, for $\{f \in C^s([0, 1]^d) : \|f\|_{C^s([0,1]^d)} \leq 1\}$ one sees that B^k achieves a rate of best N-term approximation error of order $N^{\frac{\delta - s}{d}}$ for every $\delta > 0$.

In order to rebuild 2.9 monomials need to be approximated by ReLU neural networks. This is done via concatenations of saw-tooth functions and an estimation from Yarotsky published in [93]. There it is shown that the square function $x \mapsto x^2$ (and therefore also polynomials) can be approximated by realisations of sufficiently deep ReLU neural networks.

Now starting with the mentioned saw-tooth functions and their efficient construction with ReLU neural networks

Lemma 2.3.3. *Let* $\Phi^\wedge := \big((A_1, b_1), (A_2, 0)\big)$ *be a ReLU neural network where*

$$A_1 := \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}, b_1 := \begin{pmatrix} 0 \\ -1 \\ -2 \end{pmatrix} \text{ and } A_2 := (1, -2, 1).$$

Then

$$R(\Phi^\wedge)(x) = \rho(2x) - 2\rho(2x-1) + \rho(2x-2)$$

is a hat function on $[0,1]$*, equals to* 0 *on* $[0,1]^c$ *and it holds that* $A(\Phi^\wedge) = (1,3,1)$*,* $L(\Phi^\wedge)$*=2 and* $M(\Phi^\wedge) = 8$*.*

Proof. One immediately notes that $L(\Phi^\wedge)$=2, $M(\Phi^\wedge) = 8$ and that the architecture of $\Phi^\wedge$ is as written in the formulation of the Lemma above.
Let $x \leq 0$, then $R(\Phi^\wedge) = 0 - 2 \cdot 0 + 0 = 0$.
Let $x \in (0, \frac{1}{2}]$, then $R(\Phi^\wedge) = 2x - 2 \cdot 0 + 0 = 2x$.
Let $x \in (\frac{1}{2}, 1)$, then $R(\Phi^\wedge) = 2x - 2(2x-1) = 1 - 2x$.
Let $x \geq 1$, then $R(\Phi^\wedge) = 2x - 2(2x-1) + (2x-2) = 0$. □

The composition of $R(\Phi^\wedge)$ with itself is particularly interesting. It holds that

$$F_n := R(\underbrace{\Phi^\wedge \bullet \cdots \bullet \Phi^\wedge}_{n-\text{times}}) = \underbrace{R(\Phi^\wedge) \circ \cdots \circ R(\Phi^\wedge)}_{n-\text{times}} \tag{2.12}$$

is a saw-tooth function on $[0,1]$ with 2^{n-1} hats of width $\frac{1}{2^{n-1}} = 2^{1-n}$ and height 1 each, see Figure 2.4 This result was published by Telgarsky in [85] and is stated here without proof.

The key observation from Yarotsky in [93] is that the map $x \mapsto x^2$ on $[0,1]$ can be approximated by linear combinations of the identity function and the functions $F_1, \dots F_N$ with F_n as in (2.12) for $1 \leq n \leq N$, see Figure 2.5. The proof is omitted here.

Proposition 2.3.4. *Let* $x \in [0,1]$ *and* $N \in N$*. Then the following holds:*

$$\left| x^2 - x + \sum_{n=1}^{N} \frac{F_n(x)}{2^{2n}} \right| \leq 2^{-2(N+1)}. \tag{2.13}$$

In order to describe the main ideas of the upcoming proof in this section, two more (ReLU specific) features are needed, they are cited here as in [64]. First, it is shown that with ReLU neural networks the identity function on $\mathbb{R}^d$ can be rebuild exactly.

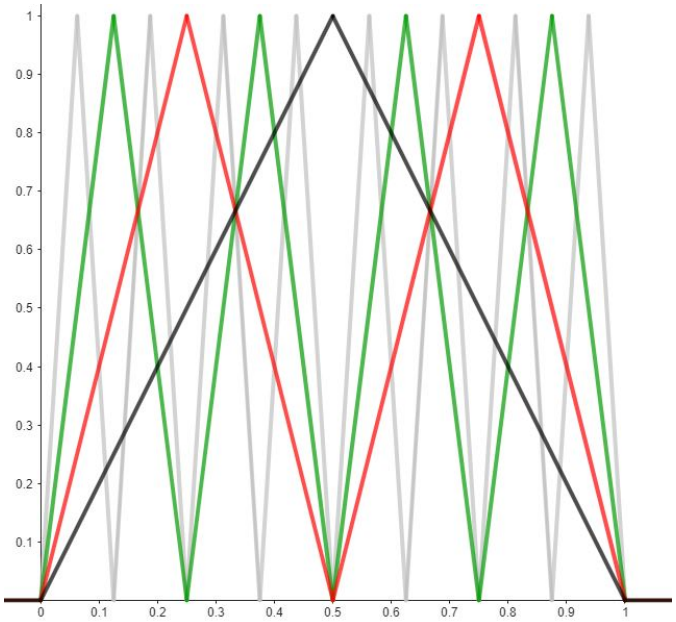

Figure 2.4: Visualisaton of $R(\Phi^\wedge) = F_1$ (black, hat function) and its compositions with itself $R(\Phi^\wedge \bullet \Phi^\wedge) = R(\Phi^\wedge) \circ R(\Phi^\wedge) = F_2$ (red, saw-tooth function with $2^1 = 2$ hats), F_3 (green, $2^2 = 4$ hats) and F_4 (gray, $2^3 = 8$ hats).

Lemma 2.3.5. *Let $d \in \mathbb{N}$ and $\Phi^{id} := \big((A_1, 0), (A_2, 0)\big)$ be a ReLU neural network where*

$$A_1 := \begin{pmatrix} Id_{\mathbb{R}^d} \\ -Id_{\mathbb{R}^d} \end{pmatrix} \text{ and } A_2 := (Id_{\mathbb{R}^d}, -Id_{\mathbb{R}^d}).$$

Then $R(\Phi^{id}) = Id_{\mathbb{R}^d}$.

Proof. Let $x \in \mathbb{R}^d$, then $R(\Phi^{id})(x) = \begin{pmatrix} \rho(x_1) - \rho(-x_1) \\ \vdots \\ \rho(x_d) - \rho(-x_d) \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} = x$, see (2.1). □

Remark 2.3.6. The lemma above can be generalized to yield emulations of the identity function with arbitrary numbers of layers. Let $d \in \mathbb{N}$ and $L \in \mathbb{N}$ with $L \geq 2$, then the neural network

$$\Phi^{id}_{d,L} := \Bigg(\Bigg(\begin{pmatrix} \mathrm{Id}_{\mathbb{R}^d} \\ -\mathrm{Id}_{\mathbb{R}^d} \end{pmatrix}, 0 \Bigg), \underbrace{(\mathrm{Id}_{\mathbb{R}^{2d}}, 0), \ldots, (\mathrm{Id}_{\mathbb{R}^{2d}}, 0)}_{L-2 \text{ times}}, ((\mathrm{Id}_{\mathbb{R}^d}, -\mathrm{Id}_{\mathbb{R}^d}), 0) \Bigg)$$

fulfills $R(\Phi^{id}_{d,L}) = \mathrm{Id}_{\mathbb{R}^d}$ and clearly $L(\Phi^{id}_{d,L}) = L$ and by counting $M(\Phi^{id}_{d,L}) = 2Ld$.

The second needed ReLU specific feature is a different kind of concatenation of ReLU networks, in order to control the number of resulting weights.

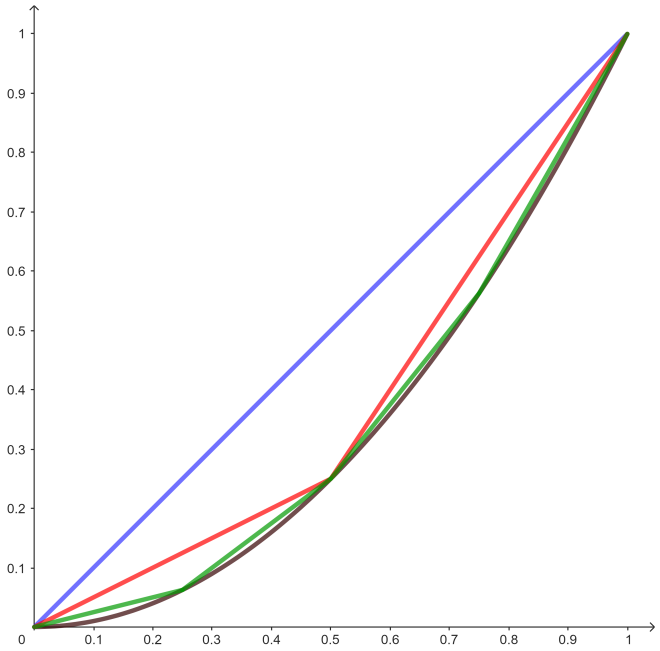

Figure 2.5: Approximation of the square function (black) with linear combinations $H_N(x) := x - \sum_{n=1}^{N} F_n(x)/2^{2n}$, compare to (2.13). H_0 is shown in blue, H_1 in red and H_2 in green. Note that the functions H_N can be approximated by ReLU neural network, see Proposition 2.3.9.

Definition 2.3.7. Let $L_1, L_2 \in \mathbb{N}$ and let

$$\Phi^1 = \Big((A_1^1, b_1^1), \dots, (A_{L_1}^1, b_{L_1}^1)\Big) \quad \text{and} \quad \Phi^2 = \Big((A_1^2, b_1^2), \dots, (A_{L_2}^2, b_{L_2}^2)\Big)$$

be two ReLU neural networks such that the input layer of Φ^1 has the same dimension as the ouput layer of Φ^2. Let Φ^{id} as in Lemma 2.3.5, then the sparse concatenation of Φ^1 and Φ^2 is defined as

$$\Phi^1 \odot \Phi^2 := \Phi^1 \bullet \Phi^{id} \bullet \Phi^2.$$

Remark 2.3.8. Comments to Definition 2.3.7.

- With Φ^1 and Φ^2 as in the Definition above, the following properties are fulfilled:

$$1) \quad R(\Phi^1 \odot \Phi^2) = R(\Phi^1) \circ R(\Phi^2), \tag{2.14}$$
$$2) \quad L(\Phi^1 \odot \Phi^2) = L(\Phi^1) + L(\Phi^2) \quad \text{and} \tag{2.15}$$
$$3) \quad M(\Phi^1 \odot \Phi^2) \leq 2M(\Phi^1) + 2M(\Phi^2). \tag{2.16}$$

- In comparison to the concatenation defined as in Definition 2.1.4 the sparse concatenation for ReLU neural networks has one layer more, but the number of weights

in the resulting network grows only linearly (and not probably quadratic as in the "normal" concatenation, see underneath) in the number of weights of the concatenated ReLU neural networks. For $N \in \mathbb{N}$ consider the case

$$\Phi := \Phi^1 := \Phi^2 := \big((A_1, 0), (A_2, 0)\big)$$

with $A_1 := (1, \dots, 1)^T \in \mathbb{R}^{N \times 1}$ and $A_2 := (1, \dots, 1) \in \mathbb{R}^{1 \times N}$. Then clearly $M(\Phi) = 2N$ and $\Phi \bullet \Phi = \big((A_1, 0), (A_1 A_2, 0), (A_2, 0)\big)$ by Definition 2.1.4, where $A_1 A_2 \in \mathbb{R}^{N \times N}$ and each entry in $A_1 A_2$ equals to 1. Comparing now the "normal" concatenation with the sparse concatenation yields

$$M(\Phi \bullet \Phi) = N + N^2 + N = N^2 + 2N, \quad \text{but} \quad M(\Phi \odot \Phi) \leq 2 \cdot 2N + 2 \cdot 2N = 8N.$$

For a visualisation of the sparse concatenation compared to the "normal" concatenation see Figure 2.6.

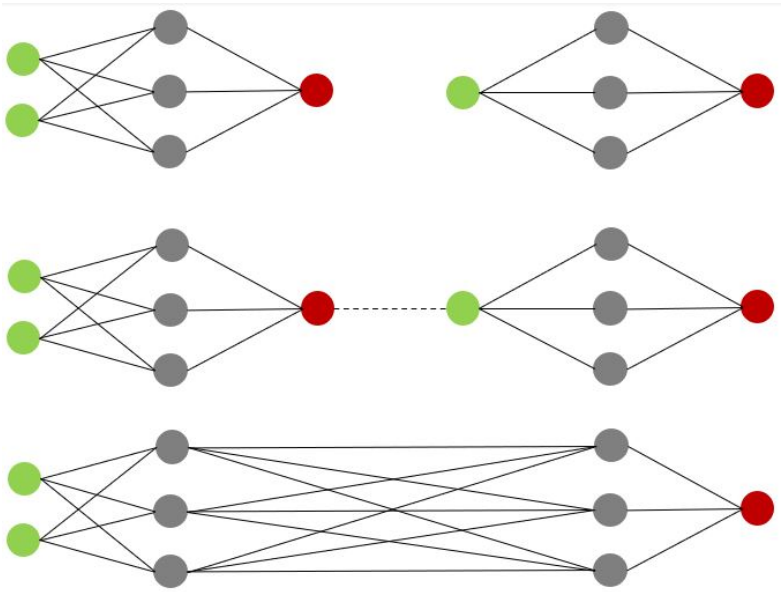

Figure 2.6: **Top:** Two neural networks Φ^1 (right) and Φ^2 (left), where the input layer of Φ^1 has the same dimension as the output layer of Φ^2.
Middle: The sparse concatenation of Φ^1 and Φ^2 according to Definition 2.3.7, where the dotted line represents the intercalated network Φ^{id} from Lemma 2.3.5.
Bottom: The "normal" concatenation of Φ^1 and Φ^2 according to Definition 2.1.4.

In order to achieve the desired goal for this subsection by ReLU neural networks, to approximate B-splines, following the idea of Yarotsky, a statement for the approximation of the map $x \mapsto x^2$ (Proposition 2 in [93]) is given.

Proposition 2.3.9. *Let $\varepsilon \in (0, \frac{1}{2})$. Then there exists a ReLU neural network $\Phi^{sq,\varepsilon}$ such that for all $x \in [0,1]$ and for $\varepsilon \to 0$*

$$\begin{aligned} L(\Phi^{sq,\varepsilon}) &= \mathcal{O}\Big(\log_2 \left(\frac{1}{\varepsilon}\right) \Big), \\ M(\Phi^{sq,\varepsilon}) &= \mathcal{O}\Big(\log_2^2 \left(\frac{1}{\varepsilon}\right) \Big) \quad \textit{and} \\ \big|R(\Phi^{sq,\varepsilon}) - x^2\big| &\leq \varepsilon. \end{aligned}$$

Proof (sketch). Choosing $N := \lceil -\frac{\log_2(\varepsilon)}{2} \rceil$ yields, that for all $x \in [0,1]$ due to (2.13) holds that

$$\left| x^2 - x + \sum_{n=1}^{N} \frac{F_n(x)}{2^{2n}} \right| \leq \varepsilon.$$

Therefore a network with realisation is $x - \sum_{n=1}^{N} \frac{F_n(x)}{2^{2n}}$ is constructed in two steps:

Step 1: Constructing N neural networks with same length $N+1$ and realisation F_n for $1 \leq n \leq N$ as follows:

$$\Phi_n := \Phi^{id}_{1,N-n} \odot \underbrace{\Phi^{\wedge} \bullet \cdots \bullet \Phi^{\wedge}}_{n-\text{times}}.$$

Step 2: Constructing a neural network which assigns the correct factors to each term in $x - \sum_{n=1}^{N} \frac{F_n(x)}{2^{2n}}$ and rebuilds the sum:

$$\Phi^{sq,\varepsilon} := \Big(\big(1, -2^{-2}, \ldots, -2^{-2N}\big), 0 \Big) \odot P(\Phi^{id}_{1,N+1}, \Phi_1, \ldots, \Phi_N),$$

then per construction $R(\Phi^{sq,\varepsilon}) = x - \sum_{n=1}^{N} \frac{F_n(x)}{2^{2n}}$.
The properties for $L(\Phi^{sq,\varepsilon})$ and $M(\Phi^{sq,\varepsilon})$ follow from (2.15) and (2.16). □

The Proposition above shows a way of emulating $x \mapsto x^2$, and therefore also $x \mapsto x_+^2$, by ReLU neural networks. Using the identity

$$\begin{aligned} 2x_1x_2 &= (x_1 + x_2)^2 - x_1^2 - x_2^2 \\ &= (x_1 + x_2)_+^2 + (-x_1 - x_2)_+^2 - (x_1)_+^2 - (-x_1)_+^2 - (x_2)_+^2 - (-x_2)_+^2 \end{aligned} \tag{2.17}$$

for $x = (x_1, x_2) \in \mathbb{R}^2$ yields that on $[-K, K]$ for $K \in \mathbb{N}$ there exists a neural network $\Phi^{mult,p,\varepsilon}$ that approximates the multiplication $\prod_{i=1}^{d} x_i$ on $[-K, K]^d$ with precision $\varepsilon \in (0, \frac{1}{2})$ (see Proposition 3 in [93]), with

$$L(\Phi^{mult,p,\varepsilon}) = \mathcal{O}\Big(\log_2(K) \cdot \log_2 \left(\frac{1}{\varepsilon}\right) \Big) \quad \text{and} \tag{2.18}$$

$$M(\Phi^{mult,p,\varepsilon}) = \mathcal{O}\Big(\log_2(K) \cdot \log_2^2 \left(\frac{1}{\varepsilon}\right) \Big). \tag{2.19}$$

Clearly (choosing $x_1 = x_2 = \cdots = x_d =: x$) also a neural network $\Phi^{pow,p,\varepsilon}$ that approximates the powers x^p (and x_+^p) for $p \in \mathbb{N}$ on $[-K, K]$ up to $\varepsilon \in (0, \frac{1}{2})$ exists, with number of layers and weights of the same order.

Using $\Phi^{pow,p,\varepsilon}$ it is possible to rebuild cardinal B-splines of arbitrary order to arbitrary precision, as it is stated underneath.

For proofs of these three statements (approximation of multiplication, powers and B-splines, respectively) see Propositions 3.16-3.18 in [66].

Proposition 2.3.10. *Let $d, k, \ell \in \mathbb{N}$, $k \geq 2$, $t \in \mathbb{R}^d$ and $\varepsilon \in (0, \frac{1}{2})$. Then there exists a ReLU neural network $\Phi^d_{\ell,t,k,\varepsilon}$ such that for all $x \in \mathbb{R}^d$ and for $\varepsilon \to 0$*

$$L(\Phi^d_{\ell,t,k,\varepsilon}) = \mathcal{O}\Big(\log_2 \left(\frac{1}{\varepsilon}\right) \Big), \tag{2.20}$$

$$M(\Phi^d_{\ell,t,k,\varepsilon}) = \mathcal{O}\Big(\log_2^2 \left(\frac{1}{\varepsilon}\right) \Big) \quad \textit{and} \tag{2.21}$$

$$\Big| R(\Phi^d_{\ell,t,k,\varepsilon}) - N^d_{\ell,t,k} \Big| \leq \varepsilon. \tag{2.22}$$

Proof (sketch). It is sufficient to show the statement for $\ell = 0$ and $t = 0$. Defining a neural network $\Phi_{k,\delta}$ that rebuilds N_k as given in (2.9), so for $\delta > 0$

$$\Phi_{k,\delta} := \Big((S_1, 0) \Big) \odot FP\Big(\underbrace{\Phi^{pow,k-1,\delta}, \ldots, \Phi^{pow,k-1,\delta}}_{(k+1)-\text{times}} \Big) \odot \Big((A_1, b_1), (Id_{\mathbb{R}^{k+1}}, 0) \Big), \text{ where}$$

$$S_1 := \left(\frac{1}{(k-1)!} \left(\binom{k}{0}, -\binom{k}{1}, \ldots (-1)^k \binom{k}{k} \right), 0 \right),$$

$$A_1 := (1, \ldots, 1)^T \in \mathbb{R}^{k+1} \text{ and } b_1 := (0, -1, \ldots, -k)^T.$$

Then it is possible to find $\delta_\varepsilon > 0$ such that for $x \in [-k-1, k+1]$

$$\Big| R(\Phi_{k,\delta_\varepsilon})(x) - N_k(x) \Big| \leq \frac{\varepsilon}{4d2^{d-1}}.$$

To extend the approximation property of $\Phi_{k,\delta_\varepsilon}$ from $[-k-1, k+1]$ to $\mathbb{R}$ defining

$$\Phi^{\text{cut}} := \Big(\big((1,1,1,1)^T, (1, 0, -k, -k-1) \big), \big((1, -1, -1, 1), 0 \big) \Big),$$

which is a piecewise linear spline fulfilling $R(\Phi^{\text{cut}}) = 1$ on $[0, k]$, vanishing on $[-1, k+1]^c$ and being non-negative and bounded by 1, compare to Figure 2.7. With

$$\tilde{\Phi}_{k,\delta} := \Phi^{mult,d,\varepsilon/(4d2^{d-1})} \odot P(\Phi_{k,\delta_\varepsilon}, \Phi^{id}_{1,L(\Phi_{k,\delta_\varepsilon})-2} \odot \Phi^{cut})$$

follows for $x \in \mathbb{R}$

$$\Big| R(\tilde{\Phi}_{k,\delta_\varepsilon})(x) - N_k(x) \Big| \leq \frac{\varepsilon}{2d2^{d-1}}.$$

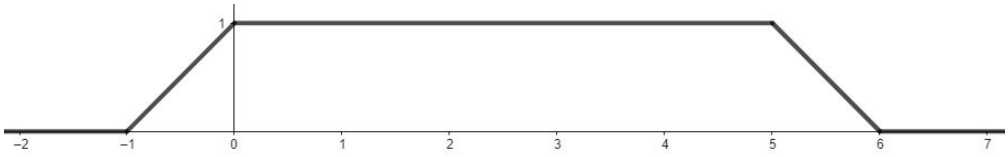

Figure 2.7: Visualization of $R(\Phi^{\text{cut}})$ for $k = 5$.

Now defining the desired neural network, $\Phi^d_{0,0,k,\varepsilon}$ rebuilding $N^d_{0,0,k}(x) = \prod_{j=1}^d N_k(x_j)$, see (2.10), as

$$\Phi^d_{0,0,k,\varepsilon} := \Phi^{mult,d,\varepsilon/2} \odot FP\Big(\underbrace{\tilde{\Phi}_{k,\delta_\varepsilon}, \dots, \tilde{\Phi}_{k,\delta_\varepsilon}}_{d-\text{times}}\Big),$$

then one can show (inserting $\prod_{j=1}^d R(\tilde{\Phi}_{k,\delta_\varepsilon})(x_j)$ and using the triangle inequality) that for $x \in \mathbb{R}^d$ it holds that

$$\left| R(\Phi^d_{0,0,k,\varepsilon}) - N^d_{0,0,k} \right| \leq \varepsilon,$$

which yields (2.22). Properties (2.20) and (2.21) follow from (2.15) and (2.16). □

It was shown above in Proposition 2.3.10 that B-splines can be approximated arbitrary well by ReLU neural networks. In other words: (2.7) in Theorem 2.2.3 holds also for the dictionary B^k with ReLU neural networks. Combining this with Theorem 2.3.2 yields the following statement:

Theorem 2.3.11. *Let $d \in \mathbb{N}$, $s > \delta > 0$ and $p \in [1, \infty]$. Then there exists a constant $C > 0$ such that for every $f \in C^s([0,1]^d)$ with $\|f\|_{C^s([0,1]^d)} \leq 1$ and every $\varepsilon \in (0, \frac{1}{2})$ there exists a ReLU neural network Φ fulfilling*

$$L(\Phi) \leq C \log_2\left(\frac{1}{\varepsilon}\right), \tag{2.23}$$

$$M(\Phi) \leq C\varepsilon^{-\frac{\delta-s}{d}} \quad \text{and} \tag{2.24}$$

$$\|f - R(\Phi)\|_{L^p([0,1]^d)} \leq \varepsilon. \tag{2.25}$$

Proof (sketch). Let $f \in C^s([0,1]^d)$ with $\|f\|_{C^s([0,1]^d)} \leq 1$ and $s > \delta > 0$. By Theorem 2.3.2 there exists a constant $C' > 0$ and, for every $N \in N$, there exists $c_i \in \mathbb{R}$ with $|c_i| \leq C'$ and $B_i \in B^k$ for $i = 1, \dots, N$ and $k := \lceil s \rceil$ such that

$$\left\| f - \sum_{i=1}^N c_i B_i \right\|_{L^p[0,1]^d)} \leq C' N^{\frac{\delta-s}{d}}. \tag{2.26}$$

With $C := \max\{1, C'\}$, (2.26) also holds with C instead of C'. By Proposition 2.3.10 each spline B_i can be approximated up to an error of $N^{\frac{\delta-s}{d}}/(CN)$ with a neural network Φ_i

with $L(\Phi_i) = \mathcal{O}(\log_2\left(N^{\frac{\delta-s}{d}}/(CN)\right)) = \mathcal{O}(\log_2(N))$ and $M(\Phi_i) = \mathcal{O}(\log_2^2\left(N^{\frac{\delta-s}{d}}/(CN)\right)) = \mathcal{O}(\log_2^2(N))$ for $N \to \infty$. With

$$\Phi_f^N := ((c_1, \dots, c_N), 0) \bullet P(\Phi_1, \dots, \Phi_N)$$

it holds for $N \to \infty$ that $L(\Phi_f^N) = \mathcal{O}(\log_2(N))$ and $M(\Phi_f^N) = \mathcal{O}(N \log_2^2(N))$. Moreover,

$$\begin{aligned}
\left\|f - R(\Phi_f^N)\right\|_{L^p([0,1]^d)} &\leq \underbrace{\left\|f - \sum_{i=1}^N c_i B_i\right\|_{L^p([0,1]^d)}}_{\leq CN^{\frac{\delta-s}{d}}, \text{ see (2.26)}} + \left\|\sum_{i=1}^N c_i B_i - R(\Phi_f^N)\right\|_{L^p([0,1]^d)} \\
&\leq CN^{\frac{\delta-s}{d}} + \sum_{i=1}^N \underbrace{\left\|c_i B_i - R(\Phi_f^N)\right\|_{L^p([0,1]^d)}}_{\leq \|\cdot\|_{L^\infty([0,1]^d)} = \|\cdot\|_\infty \text{ on } [0,1]^d} \\
&\leq CN^{\frac{\delta-s}{d}} + \sum_{i=1}^N \underbrace{\left\|c_i B_i - R(\Phi_f^N)\right\|_\infty}_{\leq N^{\frac{\delta-s}{d}}/(CN), \text{ under (2.26)}} \\
&\leq CN^{\frac{\delta-s}{d}} + \underbrace{N \cdot \frac{N^{\frac{\delta-s}{d}}}{CN}}_{= N^{\frac{\delta-s}{d}}/C} = \underbrace{\left(C + \frac{1}{C}\right)}_{\leq C+1} N^{\frac{\delta-s}{d}} \leq (C+1) N^{\frac{\delta-s}{d}},
\end{aligned}$$

choosing $N = N_\varepsilon := \lceil \left(\frac{\varepsilon}{C+1}\right)^{\frac{d}{\delta-s}} \rceil$ yields (2.25). For estimates of $L(\Phi_f^{N_\varepsilon})$ and $M(\Phi_f^{N_\varepsilon})$, so (2.23) and (2.24), see the proof of Theorem 3.19 in [66]. □

Remark 2.3.12. Features of the result given in Theorem 2.3.11.

- With increasing smoothness of the functions, ReLU networks with fewer weights are needed to achieve a certain approximation accuracy. This can be probably seen easier if one rewrites the needed weights in the statement above as $M(\Phi^\varepsilon) \leq C\varepsilon^{-\frac{d}{s-\delta}}$, for larger s this is a smaller number.
- It can be shown that Theorem (2.3.11) holds for $\delta = 0$, but with

  $$M(\Phi) \leq C\varepsilon^{-\frac{0-s}{d}} \log_2\left(\frac{1}{\varepsilon}\right) = C\varepsilon^{-\frac{d}{s}} \log_2\left(\frac{1}{\varepsilon}\right),$$

  weights and also for $f \in C^s([-K, K]^d)$ for $K > 0$, $f \in C^s(B_K(0))$ (where $B_K(0)$ denotes the closed ball with radius K around 0 in $\mathbb{R}^d$) for $K > 0$ or, more general for $f \in C^s(\Omega)$ for nonempty and bounded $\Omega \subset \mathbb{R}^d$ but with C depending on K or Ω, respectively, see [45, 93] where Taylor approximation instead of approximation with B-splines is used. This fact will be used later on in various proofs.

- In order to achieve certain accuracy for fixed s and fixed (or ignored) δ, one needs $M = \mathcal{O}(\varepsilon^{-d})$ weights, so there is an exponential dependence on the input-dimension d: the previously introduced curse of dimensionality.

A second approximation method, the linear finite element method, is rebuilt exactly by ReLU neural networks in the upcoming subsection.

2.3.2 Linear finite elements and ReLU neural networks

In order to show that the linear finite element method can be emulated by ReLU neural networks [34] a special type of considered meshes on a compact subset of $\mathbb{R}^d$ are defined.

Definition 2.3.13. Let $d \in \mathbb{N}$ and $\Omega \subset \mathbb{R}^d$ be compact. A set $\mathcal{T} \subset \mathcal{P}(\Omega)$, fulfilling

- $\mathcal{T} = (\pi_i)_{i=1}^{M_\mathcal{T}}$ for $M_\mathcal{T} \geq 2$, where each π_i is a d-simplex, which is the convex hull of $d+1$ affinely independent points, denoted by $\mathrm{co}(\pi_i)$,
- $\bigcup_{i=1}^{M_\mathcal{T}} \mathrm{co}(\pi_i) = \Omega$ and
- $(\pi_i \cap \pi_j) \subset (\partial\pi_i \cap \partial\pi_j)$ is an n-simplex with $n < d$ for every $i \neq j$,

is called simplicial mesh of Ω. For $1 \leq i \leq M_\mathcal{T}$ the d-simplices π_i are called cells of the mesh $\mathcal{T}$ and the extremal points of π_i are called nodes of the mesh $\mathcal{T}$. The set of all nodes is denoted by $(\mu_k)_{k=1}^{M_N}$. In order to prevent degenerated meshes, additionally assuming that if a node is in a cell, so $\mu_k \in \pi_i$ for some $1 \leq i \leq M_\mathcal{T}$ and $1 \leq k \leq, M_N$, then the node μ_k is an extremal point of the cell π_i. Moreover, assuming $M_\mathcal{T} \geq 2$, so $\pi_1 \subsetneq \Omega$.

For a given mesh $\mathcal{T} = (\pi_i)_{i=1}^{M_\mathcal{T}}$ defining the linear finite element space

$$V_\mathcal{T} := \{f \in C(\Omega) : f|_{\pi_i} \text{ affine linear for } i = 1, \ldots, M_\mathcal{T}\}.$$

Clearly, an affine linear function from (a subset Ω of) $\mathbb{R}^d \to \mathbb{R}$ is uniquely defined through values on $d+1$ linearly independent points. Therefore, for a given mesh $\mathcal{T} = (\pi_i)_{i=1}^{M_\mathcal{T}}$, every $f \in V_\mathcal{T}$ is uniquely defined through the function values at the nodes, so through $(f(\mu_k))_{k=1}^{M_N}$. The other way round, for every choice of $(y_k)_{k=1}^{M_N}$ there exists a $f \in V_\mathcal{T}$ such that $f(\mu_k) = y_k$ for $1 \leq k \leq M_N$.

For $1 \leq k \leq M_N$ defining the Courant elements $\phi_{k,\mathcal{T}} \in V_\mathcal{T}$ to be those functions that satisfy

$$\phi_{k,\mathcal{T}}(\mu_\ell) = \delta_{k,\ell}. \tag{2.27}$$

Then the following (compare to (3.3) in [34]) holds:

Lemma 2.3.14. *Let $d \in \mathbb{N}$ and $\mathcal{T}$ be a simplicial mesh of compact $\Omega \subset \mathbb{R}^d$. Then for every $f \in V_\mathcal{T}$ in holds that*

$$f = \sum_{k=1}^{M_N} f(\mu_k)\phi_{k,\mathcal{T}}. \tag{2.28}$$

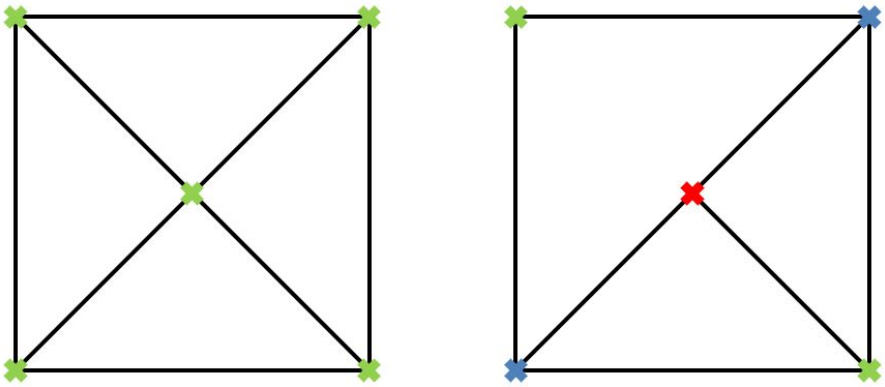

Figure 2.8: Visualization of a simple non-degenerated (left) and degenerated (right) simplicial mesh on a square in $\mathbb{R}^2$ as defined in Definition 2.3.13, where the cells are triangles. The nodes of the mesh are shown as crosses. The red node of the right mesh makes the mesh degenerated, as it is a node of the right and bottom triangular cell, but not of the bigger triangular cell in the upper left corner. The left mesh is locally convex according to Definition 2.3.15, the right mesh is not locally convex due to the blue marked nodes, as the union of the cells where these nodes are part of, respectively, are not convex sets.

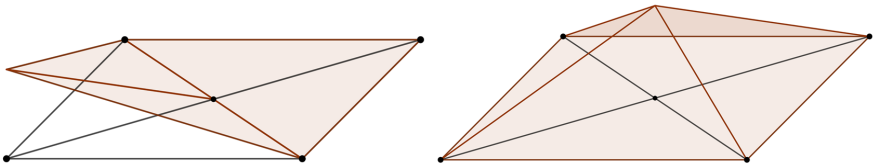

Figure 2.9: Visualization of two Courant for the simplicial mesh given on the left-hand side in Figure 2.8.

Therefore, a ReLU neural network with realisation $f \in V_{\mathcal{T}}$ can be constructed, if for $1 \leq k \leq M_N$ every Cournot element $\phi_{k,\mathcal{T}}$ can rebuild with a ReLU neural network. In order to rebuild the Cournot elements as realisation of a ReLU neural network, an additional assumption on the mesh has to be made. Moreover, for the sake of convenience two more sets are defined.

Definition 2.3.15. A mesh $\mathcal{T} = (\pi_k)_{k=1}^{M_{\mathcal{T}}}$ is called locally convex, if for every node μ_i with $1 \leq i \leq M_N$ it holds that the union of all cells containing this node, so $\cup\{\pi_k : \mu_i \in \pi_k\}$, is convex.

For $1 \leq k, \ell \leq M_N$ defining

$F(k) := \{i \in \{1, \dots, M_{\mathcal{T}}\} : \mu_k \in \pi_i\}$, the set of indices of the cells containing μ_k, and
$H(i,k) := \{\mu_\ell \in \pi_i : \mu_\ell \neq \mu_k\}$, the set of all nodes in cell π_i except μ_k for $1 \leq i \leq M_{\mathcal{T}}$.

The mesh given in Figure 2.8 on the left-hand side is locally convex.

Now one can show that the following holds (see Lemma 3.1 in [34], where also the proof is stated):

Lemma 2.3.16. *Let $d \in \mathbb{N}$ and $\mathcal{T}$ be a locally convex simplicial mesh of compact $\Omega \subset \mathbb{R}^d$. Then, for every $1 \leq k \leq M_N$, it holds that*

$$\phi_{k,\mathcal{T}} = \max\Big\{0, \min_{\ell \in F(k)} g_\ell\Big\}, \tag{2.29}$$

where g_ℓ is the unique affine linear map with $g_\ell(\mu_i) = 0$ for $\mu_i \in H(\ell, k)$ and $g_\ell(\mu_k) = 1$.

Now it is possible to rebuild the Cournot elements with ReLU neural networks using (2.29) and a clever construction to rebuild the min-operation (see Theorem 3.1 in [34]):

Theorem 2.3.17. *Let $d \in \mathbb{N}$ and $\mathcal{T}$ be a locally convex simplicial mesh of compact $\Omega \subset \mathbb{R}^d$. Let $k_\mathcal{T}$ denote the maximum number of neighbouring cells of the mesh, so*

$$k_\mathcal{T} := \max_{1 \leq k \leq M_N} \#F(k). \tag{2.30}$$

Then there exists a constant $C > 0$ such that, for every $1 \leq k \leq M_N$, there exists a ReLU neural network $\Phi_k^\mathcal{T}$ with

$$L(\Phi_k^\mathcal{T}) = \lceil \log_2(k_\mathcal{T}) \rceil + 2, \tag{2.31}$$

$$M(\Phi_k^\mathcal{T}) \leq C \cdot (k_\mathcal{T} + d) k_\mathcal{T} (\lceil \log_2(k_\mathcal{T}) \rceil) \quad \text{and} \tag{2.32}$$

$$R(\Phi_k^\mathcal{T}) = \phi_{k,\mathcal{T}}. \tag{2.33}$$

Proof. Constructing a network with realisation (2.29): One observes that, for $x, y \in \mathbb{R}$,

$$\min\{x, y\} = \frac{x+y}{2} - \frac{|x-y|}{2} = \frac{1}{2}(\rho(x+y) - \rho(-x-y) - \rho(x-y) - \rho(y-x)).$$

Therefore, the neural network $\Phi^{\min,2} := \big((A_1, 0), (A_2, 0)\big)$ with

$$A_1 := \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} \text{ and } A_2 := \begin{pmatrix} \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2} \end{pmatrix}$$

yields $R(\Phi^{\min,2})(x, y) = \min\{x, y\}$, with $L(\Phi^{\min,2}) = 2$ and $M(\Phi^{\min,2}) = 12$. The neural network $\Phi^{\min,2}$ is shown in Figure 2.10.

Now with a binary tree construction for $p \in \mathbb{N}$ even considering the networks

$$\tilde{\Phi}^{\min,p} := FP(\underbrace{\Phi^{\min,2}, \ldots, \Phi^{\min,2}}_{\frac{p}{2}\text{-times}}),$$

for p uneven adding an artificial 0 at the input. One notes that in the first layer of $\tilde{\Phi}^{\min,p}$ the number of neurons is bounded by $4 \cdot \frac{p}{2} = 2p$. For $p = 2^q$ and $q \in \mathbb{N}$ defining

$$\Phi^{\min,p} := \tilde{\Phi}^{\min,2} \bullet \tilde{\Phi}^{\min,2^2} \bullet \cdots \bullet \tilde{\Phi}^{\min,p},$$

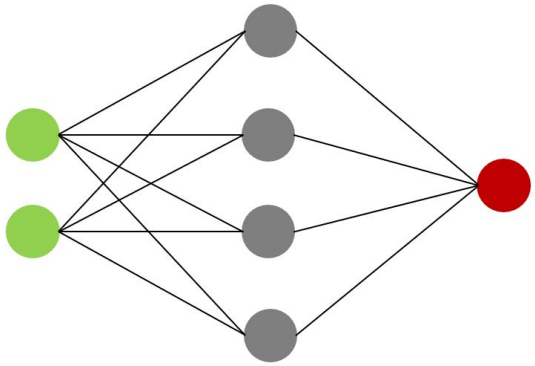

Figure 2.10: Visualization of the neural network $\Phi^{\min,2}$.

see Figure 2.11. Now clearly $R(\Phi^{\min,p})(x_1,\dots,x_p) = \min\{x_1,\dots,x_d\}$. If p is not a power of 2, a small adaption in the binary tree construction is necessary, which is omitted here. One counts that $L(\Phi^{\min,p}) = \lceil \log_2(p) \rceil + 1$ and by construction each layer (apart from the input layer) of $\Phi^{\min,p}$ has less than $2p$ neurons. Also by construction all affine maps appearing in the construction of $\Phi^{\min,p}$ are linear, so

$$\Phi^{\min,p} = \big((B_1,0),\dots,(B_{L(\Phi^{\min,p})},0)\big). \tag{2.34}$$

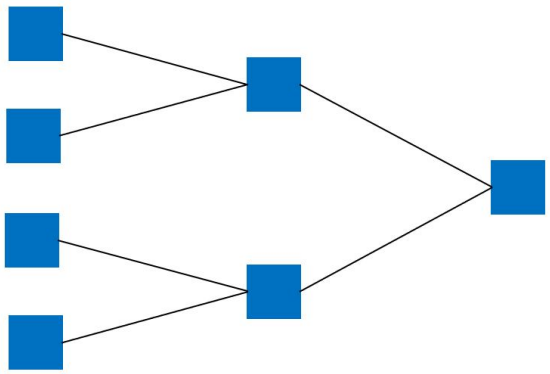

Figure 2.11: Visualization of the neural network $\Phi^{\min,8}$, where each of the blue squares does **not** display a neuron but the neural network $\Phi^{\min,2}$ shown in Figure 2.10.

For $1 \le \ell \le M_N$ each of the maps g_ℓ in (2.29) is affine, so has the form $g_\ell(x) = \langle a_\ell, x\rangle + b_\ell$ with $a_\ell \in \mathbb{R}^d, b_\ell \in \mathbb{R}$. For $1 \le k \le M_N$ let Φ_k^{aff} be the one-layered neural network defined as

$$\Phi_k^{\text{aff}} := P\big(((a_1^T,b_1)),((a_2^T,b_2)),\dots,((a_{\#F(k)}^T,b_{\#F(k)}))\big),$$

with d-dimensional input and $\#F(k)$-dimensional output, compare to Figure 2.12.

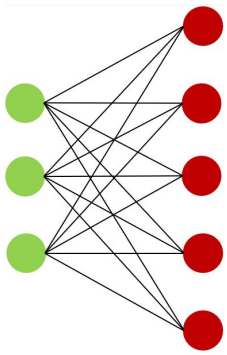

Figure 2.12: Visualization of the one-layered neural network Φ_k^{aff} with input dimension $d = 3$ and output dimension $\#F(k) = 5$, whose realisation is an affine map.

Finally, for $p := \#F(k)$ defining

$$\Phi_k^{\mathcal{T}} := \big((1,0),(1,0)\big) \bullet \Phi^{\min,p} \bullet \Phi_k^{\mathrm{aff}},$$

then by construction $R(\Phi_k^{\mathcal{T}}) = \max\Big\{0, \min_{\ell \in F(k)} g_\ell\Big\} = \phi_{k,\mathcal{T}}$, see (2.29). The neural network $\big((1,0),(1,0)\big)$ is displayed in Figure 2.13. Moreover, $L(\Phi_k^{\mathcal{T}}) = L(\Phi^{\min,p}) + 1 = \lceil \log_2(p) \rceil + 2 =: L$.

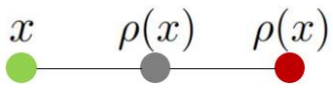

Figure 2.13: Visualization of the neural network $\big((1,0),(1,0)\big)$. Next to the two neurons in the hidden (gray) and output layer (red) the current realization after the respective neuron for input $x \in \mathbb{R}$ is shown.

Also, by construction, the number of neurons in each layer, apart from the input layer, is bounded by $2p$. Due to (2.34) and the definition of concatenation (see (2.2)) it holds that $\Phi_k^{\mathcal{T}}$ is of the form

$$\big((C_1, d_1)), (C_2, 0), \ldots, (C_L, 0)\big),$$

with $C_j \in \mathbb{R}^{N_j \times N_{j-1}}$ for $1 \le j \le L$ and $d_1 \in \mathbb{R}^p$, where $N_0 = d$ and $N_L = 1$. One follows

$$M(\Phi_k^{\mathcal{T}}) \le p + \sum_{j=1}^{L} \big(\|C_j\|_0 \big) \le p + \sum_{j=1}^{L} N_{j-1} N_j \le p + dN_1 + \sum_{j=2}^{L-1} N_{j-1} N_j + N_{L-1}$$

and due to $N_j \le 2p$ for $1 \le j \le L-1$

$$M(\Phi_k^{\mathcal{T}}) \le p + 2dp + (L-2)(2p)^2 + 2p = 3p + 2dp + (2p)^2 \big(\lceil \log_2(p) \rceil\big).$$

Due to the definition of $k_{\mathcal{T}}$, it holds that $k_{\mathcal{T}} \geq p$ for $1 \leq k \leq M_N$. Moreover, by assumption on the mesh it holds that $M_{\mathcal{T}} \geq 2$, hence clearly $k_{\mathcal{T}} \geq 2$ yielding $\lceil \log_2(k_{\mathcal{T}}) \rceil \geq 1$. It follows that

$$\begin{aligned} M(\Phi_k^{\mathcal{T}}) &\leq k_{\mathcal{T}}\Big(3 + 2d + 4k_{\mathcal{T}}(\lceil \log_2(k_{\mathcal{T}}) \rceil)\Big) \leq Ck_{\mathcal{T}}\Big(d + k_{\mathcal{T}}(\lceil \log_2(k_{\mathcal{T}}) \rceil)\Big) \\ &\leq Ck_{\mathcal{T}}(\lceil \log_2(k_{\mathcal{T}}) \rceil)(d + k_{\mathcal{T}}), \end{aligned}$$

which was claimed. □

Of course, one could ask, what happens with respect to the needed numbers of layers and weights in the proof of the Theorem above using the sparse concatenation instead of the "normal" concatenation to connect the constructed networks. One notes regarding the layers, that there is no benefit in using the sparse concatenation, as the resulting network would be have much more layers, compare (2.2) to (2.15). Nevertheless, the number of resulting weights using the sparse concatenation would be less, see (2.16).

Now combining Lemma 2.3.14 and Theorem 2.3.17 yields that every continuous, piecewise linear function on a locally compact mesh with M_N nodes can be represented as the realisation of a neural network with CM_N weights, where the constant C only depends on the input dimension d and the maximum number of neighbouring cells $k_{\mathcal{T}}$.

Theorem 2.3.18. *Let $d \in \mathbb{N}$ and $\mathcal{T}$ be a locally convex simplicial mesh of compact $\Omega \subset \mathbb{R}^d$ with nodes $(\mu_k)_{k=1}^{M_N}$. Let $k_{\mathcal{T}}$, as before (see (2.30)), denote the maximum number of neighbouring cells of the mesh. Then there exists a constant $C > 0$ such that, for every $f \in V_{\mathcal{T}}$, there exists a ReLU neural network Φ^f with*

$$\begin{aligned} L(\Phi^f) &= \lceil \log_2(k_{\mathcal{T}}) \rceil + 2, \\ M(\Phi^f) &\leq CM_N \cdot (k_{\mathcal{T}} + d)k_{\mathcal{T}}(\lceil \log_2(k_{\mathcal{T}}) \rceil) \quad \text{and} \\ R(\Phi^f) &= f. \end{aligned}$$

Proof. For $1 \leq k \leq M_N$ let $\Phi_k^{\mathcal{T}}$ be as is the proof of Theorem 2.3.17. Defining

$$\Phi^f := \Big(\big(f(\mu_1), \dots, f(\mu_{M_N})\big), 0)\Big) \bullet P\big(\Phi_1^{\mathcal{T}}, \dots, \Phi_{M_N}^{\mathcal{T}}\big)$$

yields $R(\Phi^f) = \sum_{k=1}^{M_N} f(\mu_k)\phi_{k,\mathcal{T}} = f$, see (2.28). By construction and (2.31) it holds that

$$L(\Phi^f) = 1 + L\Big(P\big(\Phi_1^{\mathcal{T}}, \dots, \Phi_{M_N}^{\mathcal{T}}\big)\Big) - 1 = L(\Phi_1^{\mathcal{T}}) = \lceil \log_2(k_{\mathcal{T}}) \rceil + 2.$$

The statement about the weights follows from (2.32) and the fact, that the mesh has M_N nodes. □

Remark 2.3.19. Remarks to Theorem 2.3.18 and the finite element method.

- One notes that for ReLU neural networks and the space $V_{\mathcal{T}}$ assumption (2.7) in Theorem 2.6 is fulfilled. Hence, whatever can be approximated by piecewise affine linear, continuous functions with N degrees of freedom can be approximated to the same accuracy by realisations of ReLU neural networks with $C \cdot N$ degrees of freedom, for a constant $C > 0$. Therefore, realisations of ReLU neural networks achieve the same approximation rate as linear finite element spaces.

- The exponential dependence on the input dimension does not appear directly in the number of needed weights in Theorem 2.3.18. However the maximum number of neighbouring cells $k_{\mathcal{T}}$ of the (locally convex simplicial) mesh $\mathcal{T}$ in compact $\Omega \subset \mathbb{R}^d$ grows exponentially in the input dimension d. Hence the curse of dimensionality is existent, but somehow hidden in $k_{\mathcal{T}}$.

- A generalization of what is shown in this section for ReLU neural networks but for higher-order finite elements can be found [62].

2.4 The curse of dimensionality

The curse of dimensionality is a term which was introduced by Bellman in the 1950s, see [5]. It is commonly used to describe an exponentially increasing difficulty of problems with increasing dimension.

The with input-dimension d exponentially deteriorating approximation rate for ReLU neural networks was shown above in Theorem 2.3.11 and Theorem 2.3.18, compare to Remarks 2.3.12 and 2.3.19. The first statement is based on approximation with B-splines, see Theorem 2.3.2 from Oswald, which suffers from the curse of dimensionality, see [20]. The second approach resembles the linear finite element method, which also suffers from the curse of dimensionality, see [4].

Way more general, it can be shown that every classical approximation method with continuous parameter dependence (including spline approximation, the linear finite element method and also approximation with neural networks) for high-dimensional function approximation suffers from the curse of dimensionality, see [61].

In fact, the approximation rates in Theorem 2.3.2 from Oswald are, apart from the δ, optimal under some very reasonable assumptions on the approximation scheme, see [19, 20].

Also the approximation rates achieved with neural networks are, without any additional assumptions on the function or the setting, optimal, see [1].

Hence, there is a fundamental lower bound on approximation capabilities of all approximation schemes that increases exponentially with the dimension, if there are no other assumptions on the function to be approximated: the curse of dimensionality.

On an abstract level it was shown here in Chapter 2 that with deep enough neural networks it is possible to approximate C^s-functions as well as with classical approximation schemes. However, at the moment it is not clear, why one should use neural networks to approximate functions. There was until here no benefit shown compared to classical

approximation tools, such as e.g. via B-splines. The exponential dependence on the dimension occurs also for the neural network approach.

The important point is that under certain additional assumptions on the function to approximate with neural networks, the dependence on the dimension is not as terrible (meaning: not exponential in terms of the needed layers and weights), one says it is possible to "overcome the curse of dimensionality". Which assumptions yield to overcoming the curse of dimensionality for neural network approximation and to which extent will be shown in the upcoming chapters.

3 Hierarchical compositionality

In the examples for approximation of C^s-functions with for neural networks via approximation of B-splines (Section 2.3) the appearing curse of dimensionality was already noted.

If one now considers a d-dimensional function f as sum of 1-dimensional functions s-times differentiable functions g_i $(i = 1, \dots, d)$, so

$$f(x) = \sum_{i=1}^{d} g_i(x_i), \tag{3.1}$$

then f can be approximated using $d\mathcal{O}(\varepsilon^{-1/s})$ many weights, which is for $\varepsilon \to 0$ asymptotically less than $\mathcal{O}(\varepsilon^{-d/s})$.

Therefore, it sounds reasonable that high-dimensional functions, which are somehow "built" out of smooth lower-dimensional functions, can be approximated better (to be exact: more efficient in terms of needed layers, neurons and weights) by neural networks than high-dimensional function which do not have this structure. One theoretical framework for this "built" is based on a special type of (directed, acyclic) graph, see Subsection 3.1.2 and the arising functions possess a hierarchical compositionality.

This idea was studied/reviewed in [70]. The most important basis for the theorems and their proofs there are [54, 55]. In [70] neural networks are defined in a slightly different way using ridge functions, meaning the neurons (there they are called nodes) of the neural networks are considered to be linear combinations of ReLU-neurons, so for $x \in \mathbb{R}^d$, each node is of the form $\sum_{i=1}^{r} c_i(\langle x, t_i\rangle + a_i)_+$ for some $r \in \mathbb{N}$, $c_i, a_i \in \mathbb{R}$ and $t_i \in \mathbb{R}^d$. Therefore the proofs in [70] can't be stated in this thesis in the way they are stated there. Instead statements from Section 2.3 as in [66] are used. Additionally, in [70] the idea is shown for infinitely often differentiable and non-polynomial activation function, which are not treated in this thesis.

A different to (3.1) but also somehow compositional approach can be found in [25], where functions of the the form

$$f(x) = \sum_{j=1}^{n} a_j \prod_{i=1}^{d} g_i^j(x_i), \tag{3.2}$$

for $n, d \in \mathbb{N}$, $x \in \mathbb{R}^d$, $a_j \in \mathbb{R}$ and $g_i^j \in C^s(\mathbb{R})$ $(1 \le j \le n$ and $1 \le i \le d)$ are considered. Functions of form (3.2) arise as solutions of several classes of high-dimensional PDEs, see [79]. Proposition 6.4 in [25] shows that these functions can be approximated by ReLU neural networks without the curse of dimensionality. However, functions of type (3.2) are not studied further on in this thesis.

Note that the smoothness of the functions which build the objective f is crucial. One could argue that each continuous d-dimensional function is built out of $2d^2 + d$ one-dimensional and continuous functions. This statement is nowadays known as Kolmogorov–Arnold representation theorem. It is stated underneath and was proofed in the 1950s, see [43].

Theorem 3.0.1. *For every $d \in N$ there are $2d^2+d$ univariate, continuous and increasing functions $\phi_{p,q}$ ($1 \leq p \leq d$, $1 \leq q \leq 2d+1$) such that for every $f \in C([0,1]^d)$ and for all $x \in [0,1]^d$ it holds that*

$$f(x) = \sum_{q=1}^{2d+1} g_q \left(\sum_{p=1}^{d} \phi_{p,q}(x) \right), \tag{3.3}$$

where g_q ($1 \leq q \leq 2d+1$) are univariate, continuous functions depending on f.

However, the "inner functions" $\phi_{p,q}$ are highly non-smooth, see [89]. Moreover, for differentiable objective f the "outer functions" g_q are non-smooth, see [27]. In fact it is even worse, as apart from some special cases (where e.g. f is constant) the smoother f is, the more non-smooth g_q has to be to cancel out the singularities of the inner functions $\phi_{p,q}$. Hence, rebuilding or approximation of the right-hand side in (3.3) with neural networks doesn't yield better approximation rates than approximating f directly.

Nevertheless, for some subsets of $C^s([0,1]^d)$ the Theorem above yields approximation results with ReLU neural networks without the curse of dimensionality, see [59]. Approximation with ReLU neural networks, using a modified version of the Theorem above, which transfers smoothness properties of the represented function to the outer functions, is studied in [77].

3.1 Hierarchically compositional functions

In this section the relevance of a special type of functions, the so-called compositional functions, is shown. In Subsection 3.1.1 the definition of these functions as well as theorems and proof ideas for the easiest case of compositionality, the so-called binary case, are shown. In Subsection 3.1.2 these ideas are extended and generalized to arbitrary compositional functions.

3.1.1 The binary case

In order to get the main ideas of the practicability of the theory, for the moment the restriction to the so-called binary case is done. So for now a function $f : \mathbb{R}^d \to \mathbb{R}$ for $d = 2^i$ for $i \in \mathbb{N}$ is considered, which is a composition of functions with two-dimensional input, so from $\mathbb{R}^2$ to $\mathbb{R}$. For $d = 8$ (so $i = 3$) this is

$$f(x_1, \ldots, x_8) = f_7\Big(f_5\big(f_1(x_1, x_2), f_2(x_3, x_4)\big), f_6\big(f_3(x_5, x_6), f_4(x_7, x_8)\big)\Big). \tag{3.4}$$

For a visualisation of the function f defined in (3.4) as a (binary) tree see Figure 3.1. If a function f is like in (3.4) it is said to be binary compositional. The appearing functions $f_j\colon \mathbb{R}^2 \to \mathbb{R}$, $1 \leq j \leq 7$ in (3.4) are called constituent functions of f. Obviously the function f is a composition, but it also contains a kind of hierarchy, sometimes also called locality, meaning that it is a composition of lower-dimensional (in the binary case: 2-dimensional) functions, whose output is again input for a lower-dimensional (in this case: 2-dimensional) function.

For a visualisation of the function f defined in (3.4) as a (binary) tree see Figure 3.1.

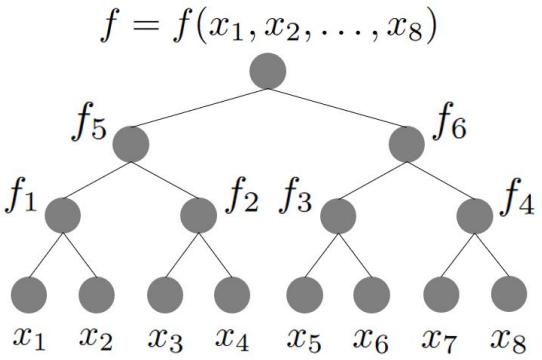

Figure 3.1: Visualization of the function f defined in (3.4) as a binary tree with constituent functions $f_j\colon \mathbb{R}^2 \to \mathbb{R}$, $1 \leq j \leq 7$.

Note that in a binary tree with input dimension d there are always $d-1$ nodes: $1 = 2^0$ nodes on top, $2 = 2^1$ in the layer underneath, $4 = 2^2$ nodes in the layer underneath, $\ldots$, $\frac{d}{2} = 2^{i-1}$ in the last layer and therefore (telescoping sum) the total number of nodes is

$$\sum_{j=0}^{i-1} 2^k = \sum_{j=0}^{i-1} 2^k \cdot (2-1) = \sum_{j=0}^{i-1} (2^{k+1} - 2^k) = \underbrace{2^i}_{=d} - 1 = d - 1. \tag{3.5}$$

After having depicted, what is meant by the binary case, the formalities and the needed definitions and restrictions on the considered function f can be given. As stated in the intro of this chapter, the smoothness of the constituent functions is the most important restriction here.

Definition 3.1.1. Let $s, d \in \mathbb{N}$ and $I^d := [-1,1]^d$ and let $C(I^d)$ be the space of all continuous functions on I^d equipped with the infinity-norm $\|f\|_\infty := \sup_{x \in I^d} |f(x)|$. Let

$$W_s^d := \{f \in C^s(I^d) : \|f\|_{C^s(I^d)} \leq 1\}. \tag{3.6}$$

Additionally, for $d = 2^i (i \in \mathbb{N})$ defining

$$W_s^{d,2} := \{f : \mathbb{R}^d \to \mathbb{R} : f \text{ binary compositional with constituent functions } f_i \in W_s^2\}.$$

Remark 3.1.2. Comments to the previous definition.

- The class of functions W_s^d is a normalized (due to the ≤ 1 in (3.6)) C^s-space. The upcoming statements also hold (up to some constants) in a more general setting, so with $\|f\|_{C^s(I^d)} \leq \kappa$ for some $\kappa > 1$. Also the restriction to I^d is not necessary but decreases the technicalities in some of the upcoming proofs without obfuscating the main ideas.

- Obviously $W_s^{d,2} \subset W_s^d$ holds, as by definition each constituent function h is in $C^s(I^2)$ and fulfills $\|h\|_{C^s(I^2)} \leq 1$, which implies $|h(x)| \leq 1$ for all $x \in I^2$. Therefore the image of each h is again in $I = [-1, 1]$, which is in all cases, except for the last constituent function, one of the two inputs for the next constituent function.

- It seems reasonable that neural networks can approximate (binary) compositional functions better than classical approximation schemes, as neural networks are (by definition) also compositional and hierarchical.

Next, a statement about approximation of functions from $W_s^{d,2}$ with ReLU neural networks is given. The idea is to lift the statement about approximation of W_s^d-functions with ReLU networks to $W_s^{d,2}$. The proof will be sketched as the binary case is a special case of what is shown underneath in more generality. Contentwise this is Theorem 4 in [70].

Theorem 3.1.3. *Let $d, s \in \mathbb{N}$. Then, for every $f \in W_s^{d,2}$ and every $\varepsilon \in (0, \frac{1}{2})$ there exists a ReLU neural network Φ_f^ε with*

$$M(\Phi_f^\varepsilon) = (d-1)\mathcal{O}(\varepsilon^{-2/s}) \quad \text{and}$$
$$\left\| f - R(\Phi_f^\varepsilon \right\|_\infty \leq \varepsilon.$$

Proof (sketch). By definition of $W_s^{d,2}$ each of the constituent functions is in W_s^2. Therefore, applying Theorem 2.3.11 (see also Remark 2.3.12) implies that each of this constituent function f_i can be approximated arbitrary well by the realisation of a ReLU neural network Φ_i up to accuracy ε. Since all the constituent functions are in W_s^2 they have bounded partial derivatives and are therefore also Lipschitz continuous with Lipschitz constant K. Now consider f_1, f_2 and f_3 such that in the binary tree of f the input for f_1 is the output from f_2 and f_3 and approximations P_1, P_2 and P_3 (which can be seen as realisations of ReLU neural networks) for f_1, f_2 and f_3 up to accuracy ε, respectively, so $\|f_i - P_i\|_\infty \leq \varepsilon$ for $1 \leq i \leq 3$. Then, by the Minkowski inequality, one follows

$$\|f_1(f_2, f_3) - P(P_1, P_2)\|_\infty \leq \|f_1(f_2, f_3) - f_1(P_2, P_3)\|_\infty + \|f_1(P_2, P_3) - P_1(P_2, P_3)\|_\infty ,$$

where $P_1(P_2, P_3)$ can be seen as realisation of a neural network, which is (sparsely) concatenated with a parallelization of 2 neural networks. Since f_1 is Lipschitz continuous with Lipschitz constant K and P_i are approximations of f_i $(i = 2, 3)$ up to ε, one yields

$$\|f_1(f_2, f_3) - f_1(P_2, P_3)\|_\infty \leq K \|(f_2, f_3) - (P_2, P_3)\|_\infty = K \max_{i=2,3} \|f_i - P_i\|_\infty \leq K\varepsilon.$$

Additionally, since P_1 is an approximation of f_1 up to accuracy ε, it holds that

$$\|f_1(P_2, P_3) - P_1(P_2, P_3)\|_\infty \leq \varepsilon.$$

So one yields with $C := K + 1$

$$\|f_1(f_2, f_3) - P_1(P_2, P_3)\|_\infty \leq C\varepsilon. \tag{3.7}$$

From the already mentioned fact 3.5 (a binary tree with d roots has $d - 1$ nodes) the claim follows. □

Remark 3.1.4. Reflection on what was shown above.

- The dependence on the needed weights in Theorem 3.1.3 is not any longer exponential in d, but of order $\varepsilon^{-2/s}$, so the curse of dimensionality has been overcome. In other words: the number 2 has replaced the input dimension d.

- The neural network considered in the proof of Theorem 3.1.3 has the same (binary) architecture than the compositional function to be approximated. The number of needed weights $M(\Phi)$ is therefore a lower bound on what could be achieved with a neural network exactly matching the architecture of the function. From the proof one could think that, to avoid the curse of dimensionality, one would have to exactly rebuild the structure of the function, which is typically not known. In fact, it is sufficient that the graph representing the structure of the function is a subgraph of the graph representing the neural network, see [70].

- The constructed deep network in the proof of Theorem 3.1.3 considers the special structure of f, whereas shallow networks (and also classical approximation schemes) are somehow blind to it. One could say that for shallow networks functions in $W_s^{d,2}$ are just functions in W_s^d.

- A similar statement to Theorem 3.1.3 holds for neural networks with activation function with order of sigmoidality $q \geq 2$, based on a statement similar to Theorem 2.3.11, see [53]. Moreover, it holds for neural networks with infinitely differentiable, non-polynomial activation function, see Theorem 2 in [70].

3.1.2 Extension to the general hierarchical case with directed acyclic graphs

The hierarchical compositionality of functions mentioned at the binary case before will now get a broader mathematical framework, based on directed acyclic graphs, which handles constituent functions with input-dimensions > 2 and more complex hierarchies in the structure of f.

The upcoming definitions are based mostly on [66], the theorem with ReLU as activation function is Theorem 5.3 there.

Definition 3.1.5. Let $d, k, N \in \mathbb{N}$ and let $G(d, k, N)$ be the set of directed acyclic graphs with N vertices, where the indegree of every vertex is at most k and the outdegree of all but one vertex is at least 1 and the indegree of exactly d vertices is 0.

For $G \in G(d, k, N)$, let $(\eta_i)_{i=1}^N$ be a topological ordering of G, so every edge $\eta_i\eta_j$ in G satisfies $i < j$.

Moreover, for $i > d$, so for the vertices with positive indegree, defining

$$T_i := \{j : \eta_j\eta_i \text{ is an edge of } G\},$$

so the indices of all nodes from where arrows go to vertex i.

Finally, defining $d_i := \#T_i$, so the number of vertices which have arrows to vertex i, then obviously $d_i \leq k$ holds for $i > d$.

Example 3.1.6. For $d = 2^j$ with $j \in \mathbb{N}$, $N = d-1$ and $d_i = 2$ for all $i > d$ the definition above yields (the graph of) the binary case.

Definition 3.1.7. Let $d, k, N, s \in \mathbb{N}$. Let $G \in G(d, k, N)$ and $I^d = [-1, 1]^d$. Additionally, let, for $i = d+1, \ldots, N$ (so for the vertices with positive indegree) $h_i \in W_s^{d_i}$, so $h_i \in C^s(I^{d_i})$ with $\|h_i\|_{C^s(I^{d_i})} \leq 1$.

For $x \in \mathbb{R}^d$ defining for $i = 1, \ldots, d$: $v_i = x_i$ and for $i = d+1, \ldots, N$: $v_i(x) = h_i(v_{j_1}(x), \ldots, v_{j_{d_i}}(x))$, where $j_1, \ldots, j_{d_i} \in T_i$ and $j_1 < \cdots < j_{d_i}$.

The function

$$f : I^d \to \mathbb{R}, \quad x \mapsto v_N(x)$$

is called a (hierarchically) compositional function associated to G with regularity s.

The set of all compositional functions associated to any G in $G(d, k, N)$ with regularity s is denoted by $CF(d, k, N; s)$.

Three different graphs, which are a valid basis for compositional functions, are displayed in Figure 3.2.

Note that compositional functions are functions that are either local (meaning they have small input-dimension) or the composition of other functions or both.

Now stating the key statement for this chapter, giving sharp upper bounds for the approximation of compositional functions belonging to an arbitrary $G \in G(d, k, N)$ with regularity s with ReLU neural networks. The statement and an extensive proof using various results from Subsection 2.3.1 is content-wise Theorem 3.19 in [66].

Theorem 3.1.8. *Let $d, k, N, s \in \mathbb{N}$ and let ρ be the ReLU. Then there exists a constant $C > 0$ such that for every $f \in CF(d, k, N; s)$ and every $\varepsilon \in (0, \frac{1}{2})$ there exists a neural network Φ_f^ε with*

$$L(\Phi_f^\varepsilon) \leq CN^2 \log_2\left(\frac{k}{\varepsilon}\right), \tag{3.8}$$

$$M(\Phi_f^\varepsilon) \leq CN^4 (2k)^{\frac{kN}{s}} \varepsilon^{-\frac{k}{s}} \log_2\left(\frac{k}{\varepsilon}\right) \quad \text{and} \tag{3.9}$$

$$\left\|f - R(\Phi_f^\varepsilon)\right\|_\infty \leq \varepsilon. \tag{3.10}$$

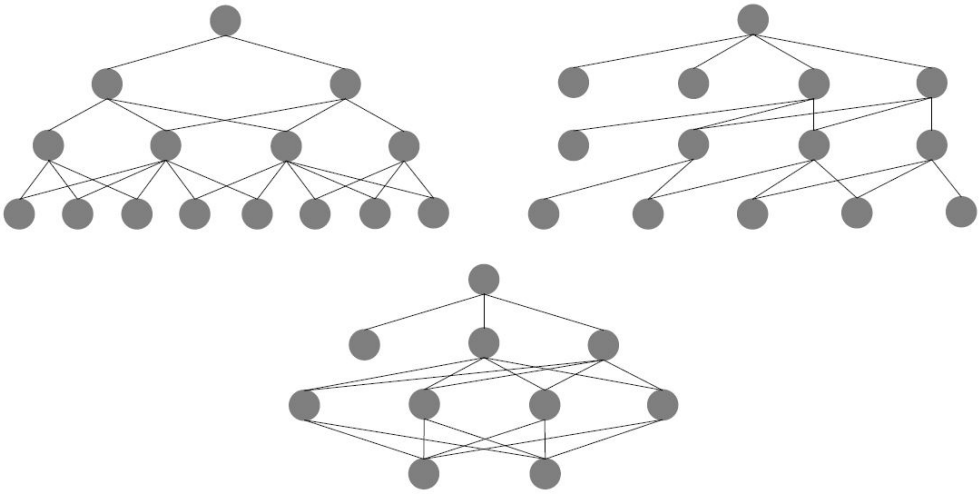

Figure 3.2: Three graphs of compositional functions.

Proof. The neural network Φ_f^ε will be constructed directly and the properties (3.8)-(3.10) for the constructed network will be shown. Therefore, let $f \in CF(d,k,N;s)$ and let, for $i=1,\dots,N$, $f_i \in W_s^{d_i}$ be according to Definition 3.1.5. Now the statements of Theorem 2.3.11 and Remark 2.3.12 with $p=\infty$ for any constituent function f_i is used (where ε is replaced by $\frac{\varepsilon}{(2k)^N}$). Hence, one can follow that there exists neural networks Φ_i and $C_i' > 0$ for $i=1,\dots,N$ such that with $C' := \max\{C_1',\dots,C_N'\} > 0$

$$|R(\Phi_i)(x) - f_i(x)| \leq \frac{\varepsilon}{(2k)^N} \quad \text{for all } x \in [-2,2]^{d_i}, \tag{3.11}$$

$$L(\Phi_i) \leq C' N \log_2\left(\frac{k}{\varepsilon}\right) \quad \text{and} \tag{3.12}$$

$$M(\Phi_i) \leq C' \varepsilon^{-d_i/s} (2k)^{\frac{d_i N}{s}} N \log_2\left(\frac{k}{\varepsilon}\right) \leq C' \varepsilon^{-k/s} (2k)^{\frac{kN}{s}} N \log_2\left(\frac{k}{\varepsilon}\right), \tag{3.13}$$

where the last inequality follows from $d_i \leq k$ for all $i=1,\dots,N$. The restriction $x \in [-2,2]^{d_i}$ will be used later within the proof.

Note that it is allowed to make implication (3.11) (which is the L^∞-norm on $[-2,2]^{d_i}$) as in the used Theorem 2.3.11 the L^p-norm is stated, including the case $p=\infty$.

For $i=d+1,\dots,N$ (so for the vertices with positive indegree) let P_i be the orthogonal projection from $\mathbb{R}^{i-1}$ to the components in T_i, so for $T_i = \{j_1,\dots,j_{d_i}\}$, where $j_1 < \dots < j_{d_i}$, defining

$$P_i\Big((x_k)_{k=1}^{i-1}\Big) := (x_{j_k})_{k=1}^{d_i}.$$

Now, for $j=d+1,\dots,N-1$ (so for the vertices which are neither input vertices nor output vertices) defining

$$\Phi_j^* := P\Big(\Phi_{j-1,L(\Phi_j)}^{id}, \Phi_j \bullet P_j\Big),$$

which is a neural network with input dimension $j-1$, output-dimension j and length $L(\Phi_j)$, whose realisation is the identity in the first $(j-1)$ components (see Remark 2.3.6) and an approximation for f_j in the j-th component, as the P_j picks the correct d_j-many indices and Φ_j approximates f_j (compare to (3.11)).

For the remaining output vertex defining the network

$$\Phi_N^* := \Phi_N \bullet P_N.$$

Now all these currently defined networks are (sparsely) concatenated to Φ_f^ε, so

$$\Phi_f^\varepsilon := \Phi_N^* \odot \Phi_{N-1}^* \odot \cdots \odot \Phi_{d+1}^*$$

for which the fulfillment of (3.8)-(3.10) is shown underneath.

First analyzing the size of Φ_f^ε. For the weights the following holds due to (2.16):

$$\begin{aligned} M(\Phi_f^\varepsilon) =& M(\Phi_N^* \odot \Phi_{N-1}^* \odot \cdots \odot \Phi_{d+1}^*) \\ \leq& 2M(\Phi_N^* \odot \cdots \odot \Phi_{\lceil (N+d+1)/2 \rceil}^*) + 2M(\Phi_{\lceil (N+d-1)/2 \rceil - 1}^* \odot \cdots \odot \Phi_{d+1}^*) \end{aligned}$$

and one can conclude inductively, by repeating this "bisection trick", that

$$M(\Phi_f^\varepsilon) \leq 2^{\lceil \log_2(N) \rceil} N \max_{j=d+1}^{N} M(\Phi_j^*) = N^2 \max_{j=d+1}^{N} M(\Phi_j^*). \tag{3.14}$$

Furthermore,

$$\max_{j=d+1}^{N} M(\Phi_j^*) \leq \max_{j=d+1}^{N-1} M(\Phi_{j-1,L(\Phi_j)}^{id}) + \max_{j=d+1}^{N} M(\Phi_j),$$

and by using Remark 2.3.6 and $j < N$ for the first term, so for the identity part of each network, one yields

$$\max_{j=d+1}^{N} M(\Phi_j^*) \leq 4N \max_{j=d+1}^{N-1} L(\Phi_j) + \max_{j=d+1}^{N} M(\Phi_j).$$

By using (3.12) and (3.13) for $L(\Phi_j)$ and $M(\Phi_j)$, respectively, one follows

$$\begin{aligned} \max_{j=d+1}^{N} M(\Phi_j^*) \leq& 2C' N^2 \log_2\left(\frac{k}{\varepsilon}\right) + C' \varepsilon^{-k/s} (2k)^{\frac{kN}{s}} N \log_2\left(\frac{k}{\varepsilon}\right) \\ \leq& C' N^2 \log_2\left(\frac{k}{\varepsilon}\right)\left(2 + \varepsilon^{-k/s}(2k)^{\frac{kN}{s}}\right). \end{aligned}$$

With an appropriate choice of $C \geq C' > 0$ one yields

$$\max_{j=d+1}^{N} M(\Phi_j^*) \leq C N^2 (2k)^{\frac{kN}{s}} \varepsilon^{\frac{k}{s}} \log_2\left(\frac{k}{\varepsilon}\right) \tag{3.15}$$

and combining (3.14) and (3.15) yields (3.9):

$$M(\Phi_f^\varepsilon) \leq C N^4 (2k)^{\frac{kN}{s}} \varepsilon^{\frac{k}{s}} \log_2\left(\frac{k}{\varepsilon}\right).$$

For the layers one yields

$$L(\Phi_f^\varepsilon) \leq N \max_{j=d+1}^{N} L(\Phi_j^*) = N \max_{j=d+1}^{N} L(\Phi_j) \leq C'N^2 \log_2\left(\frac{k}{\varepsilon}\right) \leq CN^2 \log_2\left(\frac{k}{\varepsilon}\right),$$

where the second inequality holds due to (3.12) and the last inequality due to $C' \leq C$. Hence, (3.8) holds for Φ_f^ε.

Now coming to the open approximation property (3.10) of Φ_f^ε:
The following claim is shown by induction: For $N > j > d, x \in [-1,1]^d$ and v_j as well as f_j as in Definition 3.1.7 it holds that

$$\left\| R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - [v_1(x), v_2(x), \ldots, v_j(x)] \right\|_\infty \leq \frac{\varepsilon}{(2k)^{N-j}}. \tag{3.16}$$

Starting with the base case, so $j = d+1$: Since the realisation of $\Phi_{d,L(\Phi_j)}^{id}$ is the d-dimensional identity, by construction, for $k \leq d$ holds

$$\Big(R(\Phi_{d+1}^*)(x)\Big)_k = v_k(x).$$

Moreover, by (3.11) it holds that

$$\left|\Big(R\Big(\Phi_{d+1}^*\Big)(x)\Big)_{d+1} - v_{d+1}(x)\right| = \left|\Big(R\Big(\Phi_{d+1}\Big)\Big(P_{d+1}(x)\Big)\Big)_{d+1} - f_{d+1}\Big(P_{d+1}(x)\Big)\right| \leq \frac{\varepsilon}{(2k)^N}.$$

Note that it is allowed to use (3.11) due to $x \in [-1,1]^d$ and therefore $P_{d+1}(x) \in [-1,1]^{d_{d+1}}$.

Now assuming for the induction step that (3.16) holds for $N-1 > j > d$. Then, since the identity (which is by definition used for all components of Φ_{j+1}^*, apart from the last one) is implemented exactly, by the induction hypothesis, it holds for $k \leq j$ that

$$\left|\Big(R\Big(\Phi_{j+1}^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\Big)_k - v_k(x)\right| \leq \frac{\varepsilon}{(2k)^{N-j}}.$$

By definition it holds that $v_{j+1}(x) = f_{j+1}(P_{j+1}([v_1(x), \ldots, v_j(x)]))$. Hence,

$$\begin{aligned}
&\left|\Big(R\Big(\Phi_{j+1}^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\Big)_{j+1} - v_{j+1}(x)\right| \\
&= \left|R(\Phi_{j+1}) \circ P_{j+1} \circ R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - v_{j+1}(x)\right| \\
&\leq \underbrace{\left|R(\Phi_{j+1}) \circ P_{j+1} \circ R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - f_{j+1} \circ P_{j+1} \circ R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\right|}_{:=\,\mathrm{I}} \\
&\quad + \underbrace{\left|f_{j+1} \circ P_{j+1} \circ R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - f_{j+1} \circ P_{j+1} \circ [v_1(x), \ldots, v_j(x)]\right|}_{:=\,\mathrm{II}}.
\end{aligned}$$

An upper bound for I is given by (3.11), so $\mathrm{I} \leq \frac{\varepsilon}{(2k)^N}$. Note that is it allowed to use (3.11), since

$$R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) \in [-2,2]^j$$

(and therefore, by definition, $P_{j+1} \circ R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) \in [-2,2]^{d_{j+1}}$), which holds due to the following: For $i = 1, \ldots, d$, by definition, $v_i = x_i \in [-1,1] \subseteq [-2,2]$. For $i = d+1, \ldots, j$, by definition, $f_i \in C^s(I^{d_i})$ with $\|f_i\|_{C^s(I^{d_i})} \leq 1$, therefore $|v_i(x)| \leq 1$ by the definition of the C^s-norm. By induction hypothesis (3.16) one yields

$$\left|\Big(R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\Big)_i - v_i(x)\right| \leq \frac{\varepsilon}{(2k)^{N-j}} \leq \frac{\frac{1}{2}}{2} = 1,$$

since $k \in \mathbb{N}$, $j < N$ and $\varepsilon \in (0, \frac{1}{2})$. Using the triangle inequality yields

$$\left|\Big(R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\Big)_i\right| \leq \left|\Big(R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\Big)_i - v_i(x)\right| + \Big|v_i(x)\Big| \leq 2,$$

therefore $R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) \in [-2,2]^j$.

For an estimation of II the properties of the constituent function f_{j+1} is used: Since $f_{j+1} \in C^s(I^{d_{j+1}})$, all the partial derivatives are bounded in absolute value by 1 and one yields, using (3.16) and $d_{j+1} \leq k$,

$$\begin{aligned}\mathrm{II} &\leq \sum_i^{d_{j+1}} \underbrace{\left|\Big(P_{j+1} \circ R\Big(\Phi_j^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\Big)_i - \Big(P_{j+1} \circ [v_1(x), \ldots, v_j(x)]\Big)_i\right|}_{\leq \frac{\varepsilon}{(2k)^{N-j}}} \\ &\leq \underbrace{d_{j+1}}_{\leq k} \frac{\varepsilon}{(2k)^{N-j}} \leq k \frac{\varepsilon}{(2k)^{N-j}} = \frac{\varepsilon}{2(2k)^{N-j-1}}.\end{aligned}$$

One verifies $\mathrm{I} \leq \mathrm{II}$ and therefore

$$\mathrm{I} + \mathrm{II} \leq 2 \cdot \mathrm{II} = \frac{\varepsilon}{(2k)^{N-j-1}}.$$

Now it is possible to show (3.10):

$$\begin{aligned}&\left|R\Big(\Phi_N^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - v_N(x)\right| \\ &= \left|R(\Phi_N) \circ P_N \circ R\Big(\Phi_{N-1}^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - v_N(x)\right| \\ &\leq \underbrace{\left|R(\Phi_N) \circ P_N \circ R\Big(\Phi_{N-1}^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - f_N \circ P_N \circ R\Big(\Phi_{N-1}^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x)\right|}_{:= \mathrm{III}} \\ &\quad + \underbrace{\left|f_N \circ P_N \circ R\Big(\Phi_{N-1}^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - f_N \circ P_N \circ [v_1(x), \ldots, v_{N-1}(x)]\right|}_{:= \mathrm{IV}}.\end{aligned}$$

III and IV can be bounded similar to I and II above, respectively (just by changing $j+1$ to N in all calculations above). So one finally yields ($k \geq 1$)

$$\text{III} + \text{IV} \leq 2 \cdot \text{IV} = 2\frac{\varepsilon}{2k} \leq \varepsilon,$$

and therefore, since $x \in [-1,1]^d$ was arbitrary,

$$\left\|R(\Phi_f^\varepsilon) - f\right\|_\infty = \left\|R\Big(\Phi_N^* \odot \cdots \odot \Phi_{d+1}^*\Big)(x) - v_N(x)\right\|_\infty \leq \varepsilon.$$

□

Remark 3.1.9. The seen overcome of the curse of dimensionality and comments on the previous Theorem 3.1.8 and the belonging proof.

- In the statement in Theorem 3.1.8 one notes that, similar to the binary case, the input dimension d does not appear in the needed numbers of layers or weights, but on the maximum indegree k of the underlying acyclic graph. One could say that "k takes the role of d". The curse of dimensionality therefore does not appear! This also shows the key property that makes deep neural networks exponentially better than shallow neural networks for the approximation of compositional functions: The locality (low dimensionality) of the constituent functions together with the fact, that compositionality (the mentioned generalization of (3.7) in the proof above) is not a big issue. Nevertheless, while the convergence rate doesn't depend on d, the constants in (3.9) are potentially very large. In particular, for fixed s, they grow superexponentially with k.
- In [66] the statement and proof of Theorem 3.1.8 is content-wise given with $[0,1]^d$ as the domain of f. In this thesis, in order to being consistent, $I = [-1,1]^d$ is used as domain for f and the proof is adapted accordingly.
- As for the binary case a similar statement holds for neural networks with activation function with order of sigmoidality $q \geq 2$ or infinitely differentiable, non-polynomial activation function (Theorem 3 in [70]).

3.1.3 Why hierarchically compositional functions are relevant

Even if it is reasonable to expect efficient approximations with neural networks for this type of functions, it is currently not clear, why compositional functions are a relevant function class to study at all. According to [56, 68], more detailed explanations are given in Chapter 5 in [69], these functions model quite good the functionality of the visual cortex in a human body, as this functionality is also strongly hierarchical. At first, the visual cortex analyzes only localized parts of what is currently visible, then it combines the outcomes of this (probably many) responses in some way (this combination is often modelled via different kinds of "pooling", see [68] or with already mentioned "ridge functions", see [70]) and repeats this procedure to get a more and more abstract and

high-level description of the current scene.

Moreover, the (strong) solution of a hyperbolic partial differential equation (assuming that the problem is well-posed, so meaning that the solution exists and is unique) can often be written as initial condition u_0 composed with another function, which is evaluated at the origin of a curve. This curves are found using the methods of characteristics and they have the property that the solution of the PDE is constant along this curves. In order to have a strong solution the initial condition u_0 and the characteristic curves, which are solutions of a system of ODEs, are assumed to be differentiable. Therefore the solution is a composition of smooth functions, but not necessarily with smoothness of some order. For details refer to Section 6.1.

3.2 Compositionality in classification problems

The main idea for this section stems from [64], where in Chapter 5 another kind of compositionality, reasonable for classification problems, is given and it is shown, that for a specific setting it is again possible to overcome the curse of dimensionality. In order to keep the length of this thesis under control, an informal version is cited and the proof is omitted, for details refer to [64]. In a nutshell, the final statement there is the following:

Theorem 3.2.1 (informal version)**.** *For $p > 0$ there exist $c > 0$ and $L \in \mathbb{N}$ such that for a classification function $f : J^d = [-1/2, 1/2]^d \to \mathbb{R}$ with regions where f take different values are seperated by smooth, say C^s for some $s \in \mathbb{N}$, hypersurfaces and f can be written as $f = g \circ \tau$ with a smooth feature map $\tau : \mathbb{R}^d \to \mathbb{R}^{d'}$ for $d' < d$ and a piecewise constant classifier function $g : \mathbb{R}^d \to \{1, \dots, N\}$ (one could also consider piecewise smooth classifiers) fulfilling $\|g \circ \tau\|_{L^p(J^d)} \leq \kappa \|g\|_{L^p(J^d)}$ for some $\kappa > 0$, then there exists a ReLU neural network Φ with*

$$\begin{aligned} L(\Phi) &\leq L, \\ M(\Phi) &\leq c \cdot \varepsilon^{-p(d'-1)/s} \quad \text{and} \\ \|R(\Phi) - f\|_{L^p(J^d)} &\leq \varepsilon. \end{aligned}$$

Note the d' in the number of needed weights, the underlying dimension d is not appearing here. Of course, one could ask why the L^p-norm instead of L^∞-norm is now used in order to measure the approximation error. This is the case as realisations of ReLU neural networks are always continuous, hence is doens't make sense to look at the L^∞-norm in order to measure the error between the realization and a discontinuous classification function, see also p.3 in [64]. For further insights refer to [36, 37].

Remark 3.2.2. Two remarks on why the decomposition of the classification function f in Theorem 3.2.1 is reasonable for classification.

- Classifier functions that occur in practice exhibit invariances, while such invariances are not incorporated into classical function spaces, like C^s-spaces on a compact domain. For instance, an image classifier should be translation and scaling invariant.

Moreover it should be invariant to small smooth deformations and small changes in color, brightness, or contrast, see e.g. [51, 92].

Therefore, a reasonable way to model such a function class is the following: Consider τ as a smooth, so C^s dimension-reducing "feature map" that incorporates the above mentioned invariances and g as piecewise constant (or smooth) function responsible for the classification after the invariances have been removed by τ, then the classifier f can be written as $f = g \circ \tau$.

- A different, also reasonable assumption for e.g. image classification is that not random arrangements of pixels are considered but only images from the "real world", which are assumed to form a manifold, see Section 4.1.

4 Approximation on manifolds

Realisations of neural networks are functions on the d-dimensional euclidean space $\mathbb{R}^d$, see Definition 2.1.1. Of course, one can only care about values that this function takes on a subset of this space, in this thesis until here most of the results are shown for compact subsets of the form $[a,b]^d$ for $a < b$, mostly $[0,1]^d$, $[-1,1]^d$ or $[-K,K]^d$ for some $K > 0$.

Another approach is to only consider functions, that are not defined on $\mathbb{R}^d$, but on a (lower dimensional) manifold $\mathcal{M} \subset \mathbb{R}^d$. For a d'-dimensional manifold $\mathcal{M}$ with $d' < d$ and, like before, a C^s-function on $\mathcal{M}$ (probably it is not clear at the moment what this should be, refer to Definition 4.1.2) one can expect an approximation rate, when using neural networks, which depends only (exponentially) on d', and not on d.

In this thesis, the just mentioned claim is shown for ReLU neural networks based on [66]. The main idea can be found e.g. in [13, 76, 80], but using the linear finite element method with ReLU neural networks, see Subsection 2.3.2 based on [34], instead of Taylor approximation (as in [13, 76] or approximation with wavelets (as in [80]).

4.1 Manifold assumptions

As stated in [76], in many machine learning applications the inputs are lying in a "small" subset compared to $\mathbb{R}^d$. As these subsets are not able to be parameterized, manifolds are a natural way to model the input space. In particular this holds for image classification problems (see [66]), where the input are vectors of typically quite big dimension, e.g. for images with 64 pixels in each direction the input dimension is $64 \cdot 64 = 4096$. It is conceivable that for the use case of image classification the input comes from the (potentially) lower dimensional manifold of natural images. Under this assumption only approximation on the manifold and not on images consisting of unstructured combinations of pixel values is considered.

The same reasoning can be used for other applications, one can think of e.g speech recognition, where only existing words in a chosen language are considered, and not random combinations of letters or phonemes or arbitrary sounds.

For further applications apart from image classification and speech recognition, where the manifold hypothesis is considered, see e.g. the recently published paper by Mhaskar and O'Dowd in 2023 [57] and the references mentioned there, or [73, 86] from 2000, which were kind of the starting point for dimensionality reduction techniques in order to locate and determine the respective lower-dimensional manifolds.

In this section the in Subsection 2.3.2 shown linear finite element method with ReLU neural networks is used to get approximation results for functions on a manifold $\mathcal{M}$ in $\mathbb{R}^d$. In order to do not get to deep into statements of differential geometry and keep the

main idea in scope, assumptions on the considered manifold are given, as in [66]. It will be shown below, why these assumptions are convenient and not too restrictive.

For $d \in \mathbb{N}, x \in \mathbb{R}^d$ and $r > 0$ the open ball with radius r is denoted by $B_r(x)$. Moreover, only smooth manifolds, so manifolds with a C^∞ atlas having C^∞ pairwise compatible charts, are considered.

Definition 4.1.1. Let $\mathcal{M}$ be a smooth d'-dimensional manifold in $\mathbb{R}^d$. For $N \in N$ and $\delta > 0$ the manifold $\mathcal{M}$ is said to be (N, δ)-covered if

(i) $\exists\, x_1, \ldots, x_N$ such that $\mathcal{M} \subset \bigcup_{i=1}^N B_{\delta/2}(x_i)$ and

(ii) the projection on the tangent space $T_{x_i}\mathcal{M}$ of $\mathcal{M}$ at x_i

$$P_i : \mathcal{M} \cap B_\delta(x_i) \to T_{x_i}\mathcal{M}$$

is injective and smooth and

$$P_i^{-1} : P_i(\mathcal{M} \cap B_\delta(x_i)) \to \mathcal{M}$$

is smooth for $1 \leq i \leq N$. In the sequel, T_{x_i} will be identified with $\mathbb{R}^{d'}$ and the atlas $(\mathcal{M} \cap B_\delta(x_i), Pi)_{i=1}^N$ is considered.

At first glance, it probably seems like for a manifold $\mathcal{M}$ being (N, δ)-covered is very restrictive. In fact, one can show that every compact, smooth manifold which is locally diffeomorphic to a disc is (N, δ)-covered for some $N \in \mathbb{N}$ and $\delta > 0$ by using the Theorem of Heine-Borel (via e.g. Theorem 2.41 in in [74]) and using various statements from Chapter 2 and 3 in [87]. For more details see Section 4 in [13] or Subsection 4.2 in [80].

It can also be shown that the number of needed balls (denoted by N) with diameter δ to cover a (compact, smooth, locally diffeomorphic to a disc) manifold $\mathcal{M}$, can be bounded, depending on the shape of $\mathcal{M}$. Writing τ for the reach of $\mathcal{M}$ (which measures how fast the curvature of $\mathcal{M}$ changes), $T_{d'}$ for the thickness of the covering (which corresponds to by how much the balls need to overlap) and denoting the surface area of $\mathcal{M}$ with $SA(\mathcal{M})$, for $\delta < \tau$ the following holds (see [13, 80]):

$$N \leq \left\lceil \frac{2^{d'} SA(\mathcal{M})}{\delta^{d'}} T_{d'} \right\rceil .$$

Next, spaces of smooth functions on a manifold $\mathcal{M}$ are defined.

Definition 4.1.2. Let $d, s \in \mathbb{N}$, I be a set and $\mathcal{M}$ be a smooth manifold with atlas $(U_i, \phi_i)_{i \in I}$ in $\mathbb{R}^d$. A function $f : \mathcal{M} \to \mathbb{R}$ is said to be s-times continuous differentiable, if for $i \in I$ the chart (U_i, ϕ_i) the composition $f \circ \phi_i^{-1} : \phi(U_i) \subset \mathbb{R}^d \to \mathbb{R}$ is s-times continuous differentiable, short: $f \in C^s(\mathcal{M}, \mathbb{R})$ if $f \circ \phi_i^{-1} \in C^s(\phi_i(U_i), \mathbb{R})$ for each $i \in I$. The definition of C^s functions is independent of the choice of the chart (U_i, ϕ_i).

If a manifold $\mathcal{M}$ is (N, δ)-covered, it is now possible to define a C^s-norm on $C^s(\mathcal{M}, \mathbb{R})$:

$$\|f\|_{C^s,\delta,N} := \sup_{1 \leq i \leq N} \left\| f \circ P_i^{-1} \right\|_{C^s(P_i(\mathcal{M} \cap B_\delta(x_i)))} .$$

4.2 Approximation statements on manifolds

Now the main Theorem of this chapter, proving that d' takes the role of d in the number of needed weights, can be given. How to finally overcome the curse of dimensionality as defined in Section 2.4 is shown underneath the proof of Theorem 4.2.1.

Theorem 4.2.1. *Let $d', d, s, N \in \mathbb{N}, d' \leq d, \delta > 0$ and $\mathcal{M} \subset \mathbb{R}^d$ be a (N, δ)-covered d'-dimensional manifold. Then there exists a constant $C > 0$ such that, for every $\varepsilon > 0$ and $f \in C^s(\mathcal{M}, \mathbb{R})$ with $\|f\|_{C^s,\delta,N} \leq 1$ there exists a ReLU neural network Φ_f^ε such that*

$$L(\Phi_f^\varepsilon) \leq C \cdot \log_2\left(\frac{1}{\varepsilon}\right), \tag{4.1}$$

$$M(\Phi_f^\varepsilon) \leq C \cdot \left(\varepsilon^{-\frac{d'}{s}} \log_2\left(\frac{1}{\varepsilon}\right)\right) \quad \text{and} \tag{4.2}$$

$$\left\|f - R(\Phi_f^\varepsilon)\right\|_\infty \leq \varepsilon. \tag{4.3}$$

Proof. The proof is structured in two parts. In the first step an alternative representation of f as sum of products of functions is given. In the second step the associated neural network is build and properties (4.1) - (4.3) are shown.
Step 1. Since the manifold $\mathcal{M}$ is (N, δ)-covered, is holds that $\mathcal{M} \subset [-K, K]^d$ for some $K > 0$. Consider a locally convex simplicial mesh $\mathcal{T}$ on $[-K, K]^d$ with the following property: For each node μ_i of $\mathcal{T}$ it holds that for $1 \leq i \leq M_N$

$$G(i) := \operatorname{supp}(\phi_{i,\mathcal{T}}) \subset B_{\delta/8}(\mu_i),$$

where $\phi_{i,\mathcal{T}}$ denotes a Courant element, see (2.27). By Lemma 2.3.14 on $[-K, K]^d$ holds

$$1 = \sum_{i=1}^{M_N} \phi_{i,\mathcal{T}}. \tag{4.4}$$

Defining $I_\mathcal{M} := \{i \in \{1, \ldots, M_N\} : \mathrm{d}(\mu_i, \mathcal{M}) \leq \frac{\delta}{8}\}$, where $\mathrm{d}(a, \mathcal{M}) := \min_{y \in \mathcal{M}} |a - y|$. Then, by construction it holds for $x \in \mathcal{M}$ that

$$1 = \sum_{i \in I_\mathcal{M}} \phi_{i,\mathcal{T}}(x).$$

By Definition 4.1.2 exist $x_1, \ldots, x_N \in \mathcal{M}$ such that $\mathcal{M} \subset \bigcup_{j=1}^N B_{\delta/2}(x_j)$. Hence, μ_i are less than $\frac{\delta}{8} + \frac{\delta}{2} = \frac{5\delta}{8}$ away from at least one of the points x_j for $1 \leq j \leq N$, formally $\mu_i \in \bigcup_{j=1}^N B_{5\delta/8}(x_j)$ for all $i \in I_\mathcal{M}$ and therefore, for each μ_i exists $j(i) \in \{1, \ldots, N\}$ such that $B_{\delta/8}(\mu_i) \subset B_{6\delta/8}(x_{j(i)}) = B_{3\delta/4}(x_{j(i)})$. In order to be precise, if for $i \in I_\mathcal{M}$ more than one index $j(i)$ exists, an arbitrary but fix one is chosen. Moreover, it holds that

$$\bigcup_{i \in I_\mathcal{M}} G(i) \subset \bigcup_{j=1}^N B_{3\delta/4}(x_j), \tag{4.5}$$

so for $i \in I_{\mathcal{M}}$ the supports of $\phi_{i,\mathcal{T}}$ are covered by open balls with radius $\frac{3\delta}{4}$ around the points $x_1, \dots, x_N$.

Now it is possible to write f, such that for $x \in \mathcal{M}$ it holds that

$$\begin{aligned} f(x) &= \sum_{i \in I_{\mathcal{M}}} \phi_{i,\mathcal{T}}(x) \cdot f(x) \\ &= \sum_{i \in I_{\mathcal{M}}} \phi_{i,\mathcal{T}}(x) \cdot \big(\underbrace{f(x) \circ P_{j(i)}^{-1}}_{=: f_{j(i)}} \circ P_{j(i)}(x) \big) \\ &= \sum_{i \in I_{\mathcal{M}}} \phi_{i,\mathcal{T}}(x) \cdot \big(f_{j(i)} \circ P_{j(i)}(x) \big), \end{aligned} \tag{4.6}$$

where $f_j : P_j(\mathcal{M} \cap B_\delta(x_j)) \to \mathbb{R}$ with $\|f_j\|_{C^s(P_j(\mathcal{M} \cap B_\delta(x_j)))} \leq \|f\|_{C^s,\delta,N} \leq 1$. Each P_j is injective, hence it holds that

$$P_j(\mathcal{M} \cap B_{3\delta/4}(x_j)) \subset \overline{P_j(\mathcal{M} \cap B_{3\delta/4}(x_j))} \subset P_j(\mathcal{M} \cap B_{7\delta/8}(x_j)),$$

where $P_j(\mathcal{M} \cap B_{3\delta/4}(x_j))$ and $P_j(\mathcal{M} \cap B_{7\delta/8}(x_j))$ are, as images of open sets under a continuous map, open. Now using a C^∞ version of the Urysohn lemma (see e.g. Proposition 2.25 in [48]) there exist for $1 \leq j \leq N$ a smooth bump function $\sigma_j : \mathbb{R}^d \to [0,1]$ such that $\sigma_j = 1$ on $\overline{P_j(\mathcal{M} \cap B_{3\delta/4}(x_j))}$ and $\operatorname{supp}(\sigma_j) \subseteq P_j(\mathcal{M} \cap B_{7\delta/8}(x_j))$. Now defining

$$\tilde{f}_j := \begin{cases} \sigma_j f_j & \text{for } x \in P_j(\mathcal{M} \cap B_\delta(x_j)) \\ 0 & \text{else} \end{cases}. \tag{4.7}$$

For each $1 \leq j \leq N$, $\tilde{f}_j$ is the product of a smooth and a C^s-function, therefore it holds that $\tilde{f}_j \in C^s(\mathbb{R}^{d'})$ with $\left\| \tilde{f}_j \right\|_{C^s} \leq C_{i\mathcal{M}}$, for a constant $C_{i,\mathcal{M}}$. Finally $\tilde{f}_j = f_j$ on $P_j(\mathcal{M} \cap B_{3\delta/4}(x_j))$ by construction, therefore rewriting (4.6) yields for $x \in \mathcal{M}$

$$\sum_{i \in I_{\mathcal{M}}} \phi_{i,\mathcal{T}}(x) \cdot \big(\tilde{f}_{j(i)} \circ P_{j(i)}(x) \big), \tag{4.8}$$

since $\phi_{i,\mathcal{T}}(x) = 0$ if $x \notin B_{3\delta/4}(x_j))$ for some $1 \leq j \leq N$, see (4.5).

Step 2. Now a ReLU neural network approximating f as given in (4.8) is constructed, by considering neural networks approximating $P_{j(i)}$, $\tilde{f}_{j(i)}$ and $\phi_{i,\mathcal{T}}$ for $i \in I_{\mathcal{M}}$ and using statements for these neural networks shown before.

Starting with $P_{j(i)}$:
For every $i \in I_{\mathcal{M}}$ the map $P_{j(i)} : [-K, K]^d \to \mathbb{R}^{d'}$ is affine linear, so defining (for $x \in \mathbb{R}^d$)

$$P_{j(i)}(x) = A_i x + b_i, \tag{4.9}$$

with $A_i \in \mathbb{R}^{d' \times d}$ and $b_i \in \mathbb{R}^{d'}$. For $i \in I_{\mathcal{M}}$ defining the neural networks $\Phi_i^P := ((A_i, b_i))$ with A_i and b_i as in (4.9), then obviously

$$R\Big(\Phi_i^P\Big) = P_{j(i)}. \tag{4.10}$$

Proceeding with $\tilde{f}_{j(i)}$:
As $\mathcal{M} \subset [-K, K]^d$ for some $K > 0$ and $P_{j(i)}$ is affine linear it holds that $P_{j(i)}(\mathcal{M}) \subset [-K_i, K_i]^{d'}$ for $K_i > 0$, for $i \in I_{\mathcal{M}}$. With $\tilde{K} = \max_{i \in I_{\mathcal{M}}} K_i$ by Theorem 2.3.11 and Remark 2.3.12 there exist a constant $C_1 > 0$ such that for every $\varepsilon_1 > 0$ and $i \in I_{\mathcal{M}}$ there exist a ReLU neural network Φ_i^f such that for $x \in [-\tilde{K}, \tilde{K}]^{d'}$

$$L(\Phi_i^f) \leq C_1 \log_2\left(\frac{1}{\varepsilon_1}\right) \tag{4.11}$$

$$M(\Phi_i^f) \leq C_1 \varepsilon^{-\frac{d'}{s}} \log_2\left(\frac{1}{\varepsilon_1}\right) \quad \text{and} \tag{4.12}$$

$$|\tilde{f}_{j(i)} - R(\Phi_i^f)| \leq \varepsilon_1. \tag{4.13}$$

Note that it is allowed to use the statement of Theorem 2.3.11, even if $\tilde{f}_{j(i)}$ doesn't fulfill $\left\|\tilde{f}_j\right\|_{C^s} \leq 1$ as considered in the mentioned theorem, but fulfills $\left\|\tilde{f}_j\right\|_{C^s} \leq C_{i\mathcal{M}}$. The used statement from Oswald (see Theorem 2.3.2), which is the basis of Theorem 2.3.11, holds up to the constant $C_{i\mathcal{M}}$ for $\left\|\tilde{f}_j\right\|_{C^s} \leq C_{i\mathcal{M}}$, so one can see the factor $C_{\mathcal{M}} := \max_{i \in I_{\mathcal{M}}} C_{i\mathcal{M}}$ included in C_1.

Finally, coming to the Courant elements $\phi_{i,\mathcal{T}}$:
By assumption the mesh $\mathcal{T}$ is simplicial and locally convex. By Theorem 2.3.18, for every $i \in I_{\mathcal{M}}$, there exists a ReLU neural network Φ_i^ϕ and constants $C_2 = C_2(d)$ and $C_3 = C_3(d)$, which also depend on the considered mesh $\mathcal{T}$, such that

$$L(\Phi_i^\phi) = C_2 \tag{4.14}$$

$$M(\Phi_i^\phi) \leq C_3 \quad \text{and} \tag{4.15}$$

$$R(\Phi_i^\phi) = \phi_{i,\mathcal{T}}. \tag{4.16}$$

Construction of the neural network approximating f:
For $\varepsilon_2 > 0$ and $L^* := L(\Phi_i^f \odot \Phi_i^P) - L(\Phi_i^\phi)$ defining

$$\Phi_i := \begin{cases} \Phi^{mult,2,\varepsilon_2} \odot P(\Phi_{1,L^*}^{id} \odot \Phi_i^\phi, \Phi_i^f \odot \Phi_i^P) & \text{for } L^* > 0 \\ \Phi^{mult,2,\varepsilon_2} \odot P(\Phi_i^\phi, \Phi_{1,|L^*|}^{id} \odot \Phi_i^f \odot \Phi_i^P) & \text{for } L^* < 0 \\ \Phi^{mult,2,\varepsilon_2} \odot P(\Phi_i^\phi, \Phi_i^f \odot \Phi_i^P) & \text{for } L^* = 0 \end{cases},$$

which approximates $\phi_{i,\mathcal{T}} \cdot (\tilde{f}_{j(i)} \circ P_{j(i)})$ and where $\Phi^{mult,2,\varepsilon_2}$ is as underneath the proof of Proposition 2.3.9 and $\Phi_{1,L}^{id}$ is as in Remark 2.3.6. Let $I_{\mathcal{M}} = \{i_1, \ldots, i_Q\}$, so $Q := |I_{\mathcal{M}}|$, then the desired neural network Φ_f^ε is given as

$$\Phi_f^\varepsilon := (((1, \ldots, 1) \bullet P\big(\Phi_{i_1}, \ldots, \Phi_{i_Q}\big). \tag{4.17}$$

Now showing that Φ_f^ε fulfills (4.1) - (4.3):

Due to (4.8) it holds that

$$\begin{aligned}\left\|f - R\left(\Phi_f^\varepsilon\right)\right\|_\infty \leq\ & Q \max_{i\in I_\mathcal{M}} \left\|\phi_{i,\mathcal{T}} \cdot \left(\tilde{f}_{j(i)} \circ P_{j(i)}\right) - R\left(\Phi_i\right)\right\|_\infty \\ \leq\ & \underbrace{Q \max_{i\in I_\mathcal{M}} \left\|\phi_{i,\mathcal{T}} \cdot \left(\tilde{f}_{j(i)} \circ P_{j(i)}\right) - R\left(\Phi_i^\phi\right) \cdot R\left(\Phi_i^f \odot \Phi_i^P\right)\right\|_\infty}_{:=\,\mathrm{I}} \\ & + \underbrace{Q \max_{i\in I_\mathcal{M}} \left\|R\left(\Phi_i^\phi\right) \cdot R\left(\Phi_i^f \odot \Phi_i^P\right) - R\left(\Phi_i\right)\right\|_\infty}_{:=\,\mathrm{II}}.\end{aligned}$$

In order to estimate I, using (4.16), so $R(\Phi_i^\phi) = \phi_{i,\mathcal{T}}$ yields

$$\mathrm{I} = \max_{i\in I_\mathcal{M}} \left\|\phi_{i,\mathcal{T}} \cdot \left(\tilde{f}_{j(i)} \circ P_{j(i)}\right) - \phi_{i,\mathcal{T}} \cdot R\left(\Phi_i^f \odot \Phi_i^P\right)\right\|_\infty.$$

Since $\|\phi_{i,\mathcal{T}}\|_\infty \leq 1$ (see (2.27) and property (2.14) of the sparse concatenation $\odot$ follows

$$\begin{aligned}\mathrm{I} \leq & \max_{i\in I_\mathcal{M}} \left\|\left(\tilde{f}_{j(i)} \circ P_{j(i)}\right) - R\left(\Phi_i^f \odot \Phi_i^P\right)\right\|_\infty \\ = & \max_{i\in I_\mathcal{M}} \left\|\left(\tilde{f}_{j(i)} \circ P_{j(i)}\right) - R\left(\Phi_i^f\right) \circ R\left(\Phi_i^P\right)\right\|_\infty,\end{aligned}$$

using (4.10), so $R\left(\Phi_i^P\right) = P_{j(i)}$ and (4.13) yields

$$\begin{aligned}\mathrm{I} \leq & \max_{i\in I_\mathcal{M}} \left\|\left(\tilde{f}_{j(i)} \circ P_{j(i)}\right) - R\left(\Phi_i^f\right) \circ P_{j(i)}\right\|_\infty \\ \leq & \max_{i\in I_\mathcal{M}} \left\|\left(\tilde{f}_{j(i)}\right) - R\left(\Phi_i^f\right)\right\|_\infty \leq \varepsilon_1.\end{aligned}$$

By definition of Φ_i it holds that $\mathrm{II} \leq \varepsilon_2$. Hence, for $\varepsilon > 0$ and $\varepsilon_1 := \varepsilon_2 = \frac{\varepsilon}{2Q}$

$$\left\|f - R\left(\Phi_f^\varepsilon\right)\right\|_\infty \leq\ Q \cdot (\varepsilon_1 + \varepsilon_2) = Q \cdot \left(\frac{2\varepsilon}{2Q}\right) = \varepsilon,$$

which is (4.3).
Estimating the number of layers (assuming $L^* \geq 0$) using (2.18), (4.11) and (4.14) yields

$$\begin{aligned}L\left(\Phi_f^\varepsilon\right) = & \max_{i\in I_\mathcal{M}} L\left(\Phi_i\right) = \max_{i\in I_\mathcal{M}} L\left(\Phi^{mult,2,\varepsilon_2}\right) + L\left(\Phi_i^f\right) + L\left(\Phi_i^P\right) \\ \lesssim & \log_2\left(\frac{1}{\varepsilon_2}\right) + \log_2\left(\frac{1}{\varepsilon_1}\right) \lesssim \log_2\left(\frac{1}{\varepsilon}\right),\end{aligned}$$

which is (4.1).
For the number of weights (assuming $L^* \geq 0$) one yields using (2.16)

$$\begin{aligned}M\left(\Phi_f^\varepsilon\right) \leq\ & Q \max_{i\in I_\mathcal{M}} M\left(\Phi_i\right) \\ & \leq 2Q \cdot \left(M\left(\Phi^{mult,2,\varepsilon_2}\right) + M\left(\Phi_{1,L^*}^{id} \odot \Phi_i^\phi\right) + M\left(\Phi_i^f \odot \Phi_i^P\right)\right) \\ & \leq 2Q \cdot \left(M\left(\Phi^{mult,2,\varepsilon_2}\right) + M\left(\Phi_{1,L^*}^{id} \odot \Phi_i^\phi\right) + M\left(\Phi_i^f \odot \Phi_i^P\right)\right).\end{aligned}$$

Moreover, using again (2.16) as well as (2.19) and Remark 2.3.6 yields

$$M\Big(\Phi_f^\varepsilon\Big) \leq 2Q \cdot \Big(C_4 \log_2\left(\frac{1}{\varepsilon}\right) + 4L^* + 2M\big(\Phi_i^\phi\big) + 2M\big(\Phi_i^f\big) + 2M(\Phi_i^P)\Big).$$

Finally, using (4.12) yields (4.2):

$$M\Big(\Phi_f^\varepsilon\Big) \lesssim \varepsilon^{-\frac{d'}{s}} \log_2\left(\frac{1}{\varepsilon}\right).$$

□

The method used for the proof above can be generalized to the following procedure (as done/sketched in e.g. [13, 15, 76, 80]), compare to [64]:

1. Defining the regularity of the C^s function f on a compact, smooth manifold $\mathcal{M}$, using its compactness, so that it can be covered by finitely many open sets $(U_i)_{i=1}^Q$ and coordinate charts $(\pi_i)_{i=1}^Q$, via local (smooth) projections $(P_i)_{i=1}^Q$ from $U_i \cap \mathcal{M}$ to the respective tangent spaces, compare to Definition 4.1.2.

2. Writing f as a sum of localized functions, using a partition of unity $(\phi_i)_{i=1}^Q$ of $\mathcal{M}$ with $\operatorname{supp}(\phi_i) \subset U_i$ (the existence of a smooth parition of unity is guaranteed by e.g. Proposition 13.6 in [87]) and using the projection P_i ($1 \leq i \leq Q$), so for $x \in \mathcal{M}$

$$f(x) = \sum_{i=1}^{Q} \phi_i(x) \cdot \Big(f \circ P_i^{-1}\big(P_i(x)\big)\Big), \tag{4.18}$$

   compare to (4.8).

3. For $1 \leq i \leq Q$ using neural networks approximation results for ϕ_i, $f \circ P_i^{-1}$ and P_i – see (4.10), (4.13) and (4.16) – and extending the domain of the inverse projections P_i^{-1} while keeping the regularity of $f \circ P_i^{-1}$ using e.g. Theorem 1 in [91]. In the proof above, the extension of the domain of P_i^{-1} constructed explicitly using the Urysohn lemma, see (4.7).

4. Building a neural network approximating f as in the right hand side of (4.18) by concatenating the respective neural networks arising from 3. and using that multiplication, composition and addition can be implemented by neural networks, see (4.17).

The approximation rate in Theorem 4.2.1 for approximating a C^s-function defined on a d'-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^d$ is, as claimed, not depended on the underlying d-dimensional space but on the dimension d' of the manifold, hence the curse of dimensionality has been overcome.

However, the constant C in the formulation of Theorem 4.2.1 is still dependent on d and may grow very fast with increasing d. In the stated proof, the constants $C_2(d)$ and $C_3(d)$ in (4.14) and (4.15) (which are contained in the constant C), appearing in

the approximation of the Courant elements (which together form a continuous partition of unity), depend on $k_{\mathcal{T}}$, the maximal number of neighbouring cells of the mesh $\mathcal{T}$, see Theorem 2.3.18. For a typical mesh (think of a mesh on $\mathbb{Z}^d$) this number $k_{\mathcal{T}}$ grows exponentially in d, compare to Remark 2.3.19. So is the exponential dependence just shifted into the constant C?

The answer is „no", as if a different (e.g. smooth) partition of unity than the one consisting of Courant elements (see (4.4)), which are continuous but piecewise affine linear and therefore not differentiable, is considered, no exponential dependence on the input-dimension d in the constant C in Theorem 4.2.1 appears. As shown in e.g. Proposition 13.6 in [87], for compact manifolds always a smooth partition of unity exists. Therefore, one could use an approximation result for smooth functions with neural networks, e.g. Theorem 2.3.11 - which was used in the proof for the approximation of $\tilde{f}_j = f \circ P_j^{-1}$ to get rid of the exponential dependence from C on d.

Approximation results on a smooth and compact manifold based on polynomial approximation instead of B-spline approximation and a learning algorithm without using any eigenmap decomposition can be found in [14]. For an introduction and overview about manifold learning including algorithms using different eigenmap decompositions refer to [38].

5 Dimension dependent Fourier transform assumption

The historically first result showing approximation rates for shallow neural networks (almost) independent of the underlying dimensions were stated by Barron in the beginning of the 1990s [2, 3]. There, functions for which $\int_{\mathbb{R}^d} \|\omega\|_2 \left|\hat{f}(\omega)\right| d\omega$ is bounded by some $C > 0$, where $\hat{f}$ denotes the Fourier transform, are considered. It was shown, that these functions are lying in the the convex hull of scaled indicator functions on a half plane and can be approximated by shallow neural networks with N neurons in the hidden layer and sigmoidal activation functions with approximation rate $\frac{1}{N}$. In the first section this statement from Barron and the respective proof is shown for a simpler setting than in [3] and examples of functions which have the mentioned property are given.

One will not be surprised that over the last 30 years the initial statement from Barron was generalized in various ways. In the second section of this chapter a generalized Theorem is stated and selected topics, which show some kind of generalization compared to the statement in [3], are given.

5.1 The classial Barron class

In this section neural networks with a special type of activation functions, being sigmoidal, see [16], are considered.

Definition 5.1.1. Let $\rho : \mathbb{R} \to \mathbb{R}$ be a continuous function. Then ρ is called sigmoidal if

$$\rho(x) \to 1 \text{ for } x \to \infty \quad \text{ and } \rho(x) \to 0 \text{ for } x \to -\infty. \tag{5.1}$$

Clearly the ReLU ρ is not sigmoidal, as $\rho(x) \to \infty$ for $x \to \infty$. Nevertheless the statements shown in this chapter also hold with small adaptions for ReLU neural networks, the approximation rates are identical – see also Theorem 5.2.2, which is a generalization of Theorem 5.1.4, the main statement in this section.

Next, a set of functions, nowadays often called "classical Barron class" or simply "Barron class" for $C > 0$ are defined.

Definition 5.1.2. For $d \in \mathbb{N}$ and $C > 0$ defining

$$\Gamma_C := \left\{ f \in L^1(\mathbb{R}^d) : \left\| \hat{f} \right\|_{L^1(\mathbb{R}^d)} < \infty, \int_{\mathbb{R}^d} \|2\pi\xi\|_2 \left|\hat{f}(\xi)\right| d\xi \leq C \right\}, \tag{5.2}$$

where for $x \in \mathbb{R}^d : \|x\|_2 = \sqrt{x_1^2 + \cdots + x_d^2}$ and $\mathcal{F}(f) := \hat{f}$ denotes the Fourier transform of f, so

$$\mathcal{F}(f(x)) = \hat{f}(\xi) = \int_{\mathbb{R}^d} f(\xi) e^{-2\pi i \langle x, \xi \rangle} d\xi,$$

and $\langle x, y \rangle$ denotes the scalar product of two vectors $x, y \in \mathbb{R}^d$, so $\langle x, y \rangle = x^T \cdot y$. Moreover, in this chapter the closed ball in $\mathbb{R}^d$ with respect to the 2-norm with radius $r > 0$ is given by B_r^d, so

$$B_r^d := \{x \in \mathbb{R}^d : \|x\|_2 \leq r\}$$

and $\left|B_r^d\right|$ denotes the Lebesgue measure of B_r^d.

Remark 5.1.3. Several comments on the definition above:

a) $f \in \Gamma_C$ needs to be a $L^1(\mathbb{R}^d)$-function, as the Fourier transform needs to be defined.

b) By the inverse Fourier theorem (see e.g. Theorem 2.4 in [83]) it holds for $f \in L^1(\mathbb{R}^d)$ that

$$f(x) = \int_{\mathbb{R}^d} \hat{f}(\xi) e^{2\pi i \langle x, \xi \rangle} d\xi. \tag{5.3}$$

c) In [3] the assumption that $\left\|\hat{f}\right\|_{L^1(\mathbb{R}^d)} < \infty$ is not stated. Later on the equation

$$f(x) - f(0) = \int_{\mathbb{R}^d} \hat{f}(\xi)(e^{2\pi i \langle x, \xi \rangle} - 1) d\xi, \tag{5.4}$$

is necessary in order to proof Lemma 5.1.7 and 5.4 holds true if $\hat{f} \in L^1(\mathbb{R}^d)$. In other words: $\left\|\hat{f}\right\|_{L^1(\mathbb{R}^d)} < \infty$ is a convenient (but not necessary) assumption such that (5.4) holds. For an elaboration, what happens without this assumption, see Section III in [3].

d) The last assumption for $f \in \Gamma_C$, so $\int_{\mathbb{R}^d} \|2\pi\xi\|_2 \left|\hat{f}(\xi)\right| d\xi \leq C$, can be interpreted as integrability of the Fourier transform of all first-order partial derivatives of f, as (see e.g. Proposition 2.1 in [83])

$$\mathcal{F}\left(\frac{\partial}{\partial x_i} f(x)\right) = 2\pi i \xi \hat{f}(\xi)$$

and therefore, using (5.3), also

$$\frac{\partial}{\partial x_i} f(x) = \int_{\mathbb{R}^d} 2\pi i \xi \hat{f}(\xi) e^{2\pi i \langle x, \xi \rangle} d\xi$$

and $|e^{2\pi i \langle x, \xi \rangle}| = 1$, by the defintion of the complex exponential function. Clearly, the assumption becomes more restrictive with growing dimension d, in the sense that the complexity of the functions does not, or only grow slowly with increasing dimension.

e) It can be shown (see [82]) that

$$\left|B_r^d\right| = \frac{(\sqrt{\pi}r)^d}{\Gamma\big((d/2)+1\big)}, \tag{5.5}$$

where Γ denotes the Gamma function and that (see again [82])

$$\left|B_r^d\right| \to 0 \quad \text{for} \quad d \to \infty. \tag{5.6}$$

Next, the main Theorem for this section will be given and proofed using two Lemmas also stated underneath.

Theorem 5.1.4. *Let $d, N \in \mathbb{N}, f \in \Gamma_C$ and $\rho : \mathbb{R} \to \mathbb{R}$ be sigmoidal. Then, for every $c > (2C)^2$, there exists a neural network Φ with*

$$L(\Phi) = 2, \tag{5.7}$$

$$M(\Phi) \leq N \cdot (d+2) + 1 \text{ and} \tag{5.8}$$

$$\frac{1}{\left|B_1^d\right|} \int_{B_1^d} |R(\Phi)(x) - f(x)|^2 \, dx \leq \frac{c}{N}. \tag{5.9}$$

In [3] the Theorem above is stated in a much more general way, see also Theorem 5.2.2 underneath. First, it is stated for general bounded sets B in $\mathbb{R}^d$, see Section II and V (and therefore also for open balls B_r^d with arbitrary radius r, see Section III) and not only for the unit ball B_1^d. Moreover, it is stated for probability measures, and not for the Lebesgue measure as in the statement in this thesis above. However, Barron clarifies how to deal with general measures, including the case of the Lebesgue measure on bounded sets, see Section II.

To get the main ideas the statement from Theorem 5.1.4 is shown, as in [66]. In order to proof the Theorem two auxiliary results are given. The first one was shown in [3].

Lemma 5.1.5. *Let G be a subset of a Hilbert space such that the norm of each $g \in G$ is bounded by $B > 0$ and let f be in the closure of the convex hull of G, so $f \in \overline{co}(G)$. Then, for every $N \in \mathbb{N}$ and $c' > B^2$ there exist $g_1, \ldots, g_N \in G$ and $c_1, \ldots, c_N \in [0,1]$ with $\sum_{i=1}^N c_i = 1$ such that*

$$\left\| f - \sum_{i=1}^N c_i g_i \right\|^2 \leq \frac{c'}{N}.$$

Proof. Let $f \in \overline{\text{co}}(G)$. Then, for every $\delta > 0$ there exists $f^* \in \text{co}(G)$ such that $\|f - f^*\| \leq \delta$.
Since $f^* \in \text{co}(G)$, there exists $m \in \mathbb{N}$ such that

$$f^* = \sum_{i=1}^m c_i' g_i'$$

for $g'_1, \dots, g'_m \in G$ and $c'_1, \dots, c'_m \in [0,1]$ with $\sum_{i=1}^m c'_i = 1$. Considering a probability distribution σ on $\{1, \dots, m\}$ with $\mathbb{P}_\sigma(k) = c'_k$ for $1 \leq k \leq m$. Let $(g_j)_{j=1}^\infty$ be i.i.d. random variables with $g_j = g'_{i_j}$, where $i_j \sim \sigma$. Then it holds for each j that

$$\mathbb{E}(g_j) = \sum_{i=1}^m c'_i g'_i = f^* \tag{5.10}$$

and therefore $(X_j)_{j=1}^\infty := g_{i_j} - f^*$, where again $i_j \sim \sigma$, are i.i.d. random variables with $\mathbb{E}(X_j) = 0$. Moreover, it holds that

$$\begin{aligned}\mathbb{E}\left(\left\|\frac{1}{N}\sum_{j=1}^N X_j\right\|^2\right) &= \frac{1}{N^2}\sum_{j=1}^N \mathbb{E}\left(\|X_j\|^2\right) = \frac{1}{N^2}\sum_{j=1}^N \mathbb{E}_{i\sim\sigma}\left(\|g_i\|^2 - 2\langle g_i, f^*\rangle + \|f^*\|^2\right) \\ &= \frac{1}{N^2}\sum_{j=1}^N \mathbb{E}_{i\sim\sigma}\left(\|g_i\|^2\right) - \|f^*\|^2 \leq \frac{1}{N^2} N B^2 = \frac{B^2}{N}. \end{aligned} \tag{5.11}$$

The first identity follows from Bienaymé's identity (see e.g. p.106 in [41]), the second identity by the defintion of X_j, the third identity by linearity of $\mathbb{E}$ and (5.10) and the final estimation by the assumption that the norm of each element in G is bounded by B. By (5.11) there exists at least one event ω such that

$$\left\|\frac{1}{N}\sum_{j=1}^N X_j(\omega)\right\|^2 \leq \frac{B^2}{N},$$

therefore

$$\left\|\frac{1}{N}\sum_{j=1}^N \left(g_{i_j}(\omega) - f^*\right)\right\|^2 = \left\|\frac{1}{N}\sum_{j=1}^N \left(g_{i_j}(\omega)\right) - f^*\right\|^2 \leq \frac{B^2}{N}.$$

Hence, using the triangle inequality, yields

$$\left\|\frac{1}{N}\sum_{j=1}^N \left(g_{i_j}(\omega)\right) - f\right\|^2 \leq \left\|\frac{1}{N}\sum_{j=1}^N \left(g_{i_j}(\omega)\right) - f^*\right\|^2 + \|f^* - f\|^2 \leq \frac{B^2}{N} + \delta,$$

and since δ was arbitrary this yields the result. □

If one considers G to be an orthonormal basis, the elements $g_i \in G$ and the coefficients $c_i \in [0,1]$ can be explicitly given, see Remark 5.11 in [66].

Above, it was shown that in a Hilbert space any function f, which is in (the closure of) the convex hull of elements of a set G, can be approximated with an approximation rate of $\frac{1}{N}$ by a convex combination of only N elements in G.

Now transferring this argument to neural networks: Every function that can be represented by an arbitrary wide two-layer neural network (so with an arbitrary number of neurons in the first layer) with bounded activation function and where the weights in the last layer are positive and sum up to one can also be approximated with a neural networks with only N neurons in the first layer and an approximation rate of $\frac{1}{N}$.

Next, in order to use the Lemma above, it is shown that for $f \in \Gamma_C$ it holds that $f_{|B_1^d} - f(0)$ is in the closure of the convex hull of the set G_C, which contains functions mapping to a constant (bounded in absolute value by $2C$) on a half plane and mapping to 0 apart from the half plane.

Definition 5.1.6. For $d \in \mathbb{N}$ and $C > 0$ defining

$$G_C := \left\{ g : B_1^d \to \mathbb{R} : g(x) = \gamma \cdot \mathbf{1}_{\mathbb{R}^+}(\langle a, x\rangle + b) \text{ with } a \in \mathbb{R}^d, b, \gamma \in \mathbb{R}, |\gamma| \leq 2C \right\},$$

where $\mathbf{1}_{\mathbb{R}^+}$ denotes the indication function on $\mathbb{R}^+ := [0, \infty)$, so $\mathbf{1}_{\mathbb{R}^+}(x) := \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{else} \end{cases}$.

Lemma 5.1.7. *Let $d \in \mathbb{N}$, $C > 0$ and let $f \in \Gamma_C$. Then $f_{|B_1^d} - f(0) \in \overline{co}(G_C)$, where the closure is taken with respect to the norm $\|\cdot\|_{L^{2,\diamond}(B_1^d)}$ defined by*

$$\|g\|_{L^{2,\diamond}(B_1^d)} := \left(\frac{1}{|B_1^d|} \int_{B_1^d} |g(x)|^2 dx \right)^{1/2}.$$

The proof here is omitted, refer to Theorem 2 in [3] or Lemma 5.12 in [66].

Proof of Theorem 5.1.4. Let $f \in \Gamma_C$. By Lemma 5.1.7 it holds that

$$f_{|B_1^d} - f(0) \in \overline{\text{co}}(G_C).$$

Moreover, for $g \in G_C$ it holds that $\|g\|_{L^{2,\diamond}(B_1^d)} \leq 2C$ as

$$\left(\frac{1}{|B_1^d|} \int_{B_1^d} |g(x)|^2 dx \right)^{1/2} \leq \left(\frac{1}{|B_1^d|} \int_{B_1^d} (2C)^2 dx \right)^{1/2} = \left(\frac{1}{|B_1^d|} (2C)^2 |B_1^d| \right)^{1/2} = 2C.$$

Applying Lemma 5.1.5 to the Hilbert space $L^{2,\diamond}\left(B_1^d\right)$ yields, that for every $N \in \mathbb{N}$ there exist γ_i with $|\gamma_i| \leq 2, a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ for $1 \leq i \leq N$ such that

$$\frac{1}{|B_1^d|} \int_{B_1^d} \left| f(x) - f(0) - \sum_{i=1}^{N} \gamma_i \mathbf{1}_{\mathbb{R}^+}(\langle a_i, x\rangle + b_i) \right|^2 dx \leq \frac{(2C)^2}{N}.$$

Since $\rho(\lambda x) \to \mathbf{1}_{\mathbb{R}^+}(x)$ for $\lambda \to \infty$ a.e. by the dominated convergence theorem, for every $\delta > 0$ there exist $a_i' \in \mathbb{R}^d$ and $b_i' \in \mathbb{R}$ for $1 \leq i \leq N$ such that by the triangle inequality

$$\frac{1}{|B_1^d|} \int_{B_1^d} \left| f(x) - f(0) - \sum_{i=1}^{N} \gamma_i \rho(\langle a_i', x\rangle + b_i') \right|^2 dx \leq \frac{(2C)^2}{N} + \delta.$$

Defining now the neural network Φ as follows:

$$\Phi := \Big(\big((\gamma_1, \dots, \gamma_N), f(0)\big)\Big) \bullet P\Big(((a_1' b_1')), \dots, ((a_N', b_N'))\Big).$$

Then clearly $L(\Phi) = 2$, $M(\Phi) \leq N \cdot d + d + d \cdot 1 + 1 = N \cdot (d+2) + 1$, so (5.7) and (5.8) hold. Moreover,

$$R(\Phi)(x) = \sum_{i=1}^{N} \gamma_i \rho(\langle a_i', x \rangle + b_i') + f(0).$$

Hence, it holds that

$$\frac{1}{|B_1^d|} \int_{B_1^d} |f(x) - R(\Phi)(x)|^2 \, dx \leq \frac{c}{N},$$

which is (5.9), as $c > (2C)^2$ by assumption. □

The extension of Theorem 5.1.4 to $r > 0$ is quite straightforward with a small change in the definition of Γ_C, see Section V in [3].

Note that in the approximation statement (5.9) the exponential dependence is not somehow hidden or incorporated in the term $|B_1^d|$ (or in $|B_r^d|$ for $r > 0$) as (5.6) holds.

Of course, one could ask, if the approximation rate of $\frac{1}{N}$ in Theorem 5.1.4 can be improved and if yes, to which extent. From Lemma 4.1 and Proposition 4.6 in [65] it follows that it cannot be improved beyond $N^{-\frac{d+2}{d}}$, which is not significantly less than $\frac{1}{N}$.

It is a natural question to ask, which functions are contained in the Barron class Γ_C and which properties these functions fulfill. In [3], Section IX an extensive list of functions thereof and properties of functions in Γ_C are given. Only a few examples are given here.

The first two properties are based on properties of the Fourier transform. For $a > 0$ and $b \in \mathbb{R}^d$ it holds that (see Proposition 2.1 in [83])

$$\mathcal{F}(f(x+b)) = \hat{f}(\xi) e^{2\pi i \langle \xi, b \rangle} \quad \text{and} \tag{5.12}$$

$$\mathcal{F}(f(ax)) = \frac{1}{a^d} \hat{f}\left(\frac{\xi}{a}\right). \tag{5.13}$$

This yields the following properties for $C, C_i > 0, f \in \Gamma_C, f_i \in \Gamma_{C_i}$, $a > 0$, $a_i \in \mathbb{R}$ and $b \in \mathbb{R}^d$ for $1 \leq i \leq n$:

- If $f \in \Gamma_C$, then $f(\cdot + b) \in \Gamma_C$, as by the inverse Fourier theorem (5.3) it holds that

$$\begin{aligned} f(x+b) &= \int_{\mathbb{R}^d} \mathcal{F}(f(x+b)) \, e^{2\pi i \langle x, \xi \rangle} d\xi \\ &\overset{(5.12)}{=} \int_{\mathbb{R}^d} \hat{f}(\xi) e^{2\pi i \langle \xi, b \rangle} e^{2\pi i \langle x, \xi \rangle} d\xi, \end{aligned}$$

  and $|e^{2\pi i \langle \xi, b \rangle}| = |e^{2\pi i \langle x, \xi \rangle}| = 1$.

- If $f \in \Gamma_C$, then $f(a\cdot) \in \Gamma_{aC}$, by the same arguments as above using (5.13) instead of (5.12).
- If $f_i \in \Gamma_C$, then $\sum_{i=1}^n a_i f_i \in \Gamma_{\sum_{i=1}^n |a_i| C_i}$, by linearity of the integral.
- The Gaussian function on $\mathbb{R}^d$, so $x \mapsto e^{-\|x\|_2^2/2}$ is in Γ_C for $C = \mathcal{O}(\sqrt{d})$.
- If f can be written as $f(x) = g(\|x\|_2)$, then $f \in \Gamma_C$ for $C = \mathcal{O}(\sqrt{d})$.

For f in Γ_C the assumption $\int_{\mathbb{R}^d} \|2\pi\xi\|_2 \left|\hat{f}(\xi)\right| d\xi \leq C$ is a dimension dependent assumption, compare to d) in Remark 5.1.3). This assumption is not directly describable in classical smoothness order. However, Barron states two interesting properties related to the smoothness of a function f and being in Γ_C, at least for some $C > 0$:
For f to fulfill $\int_{\mathbb{R}^d} \|2\pi\xi\|_2 \left|\hat{f}(\xi)\right| d\xi < \infty$ it is ...

... necessary (and not sufficient) that all first-order partial derivatives are bounded, see [3], Section II.

... sufficient (but not necessary) that all partial derivatives of order up to $\lfloor d/2 \rfloor + 2$ are square-integrable, so f is in the Sobolev space $W^{\lfloor d/2 \rfloor + 2, 2}(\mathbb{R}^d)$. It can be shown that the constant C depends linearly on the $\|f\|_{W^{\lfloor d/2 \rfloor + 2, 2}(\mathbb{R}^d)}$, for details refer to example (15) in Section IX in [3].

The last two examples above and the statement about smoothness show the indirect dependence of C on the input dimension d.

5.2 Extensions

During the last 30 years, since Barron published his paper about overcoming the curse of dimensionality with shallow neural networks for functions in the Barron class Γ_C, numerous authors have published generalizations in some sense of the initial statement. Hence what is nowadays meant by "Barron class" or "Barron space" is not clear without rigorous specification, for an overview what can be meant see Section 7 in [12]. In this thesis two of the above mentioned generalizations are shown.

First, in Chapter 5.2.1 it is shown that the curse of dimensionality can be overcome for classification tasks, where for each area $\partial\Omega_i \subset \mathbb{R}^d$ can be described locally as a $d-1$ dimensional Barron function. Note, that in this case the accuracy of approximation does depend on the dimension d, but only as a multiplicative factor which is polynomial in d. The main source for this idea is [12].

Secondly, a set – the nowadays often called "Barron space" – which can be seen as a suitable continuum version of Realizations of two-layered neural networks, is defined. Moreover, a norm on this set, the "Barron norm" is defined and it is shown that a function f in the Barron class can be approximated by a shallow ReLU neural network with N

neurons in hidden layer with approximation rate $\frac{1}{N}$, so without exponential dependence on the input dimension with rate. Moreover properties of the Barron space and the relationship to the Barron class is studied. The basis for Subsection 5.2.2 is [21].

Content-wise not covered in this thesis, but worth to mention is [47], where the compositions of Barron class functions is studied.

5.2.1 Classification of sets with Barron class boundary

In general, approximation rates of classification functions of the form $\sum_{i=1}^{n} f_i \mathbf{1}_{\Omega_i}$, where each $\Omega_i \subset \mathbb{R}^d$ is an open set with piecewise smooth $\partial\Omega_i$ and f_i piecewise smooth (that includes also the case for constant f_i), are determined by the smoothness of $\partial\Omega_i$ and f_i, see e.g. [64].

Now a different approach, considering that the boundary $\partial\Omega_i$ can be parameterized by (d-1)-dimensional Barron functions for $1 \leq i \leq n$, as defined underneath, is shown to also overcome the curse of dimensionality. The definition given underneath can be seen as a generalization of Definition 5.1.2 to arbitrary bounded and nonempty sets.

Definition 5.2.1. Let $d \in \mathbb{N}$, $C > 0$ and $\Omega \subset \mathbb{R}^d$ be nonempty and bounded. A function $f : \Omega \to \mathbb{R}$ is said to be of Barron class with constant C if there are $x_0 \in \Omega$, $c \in [-C, C]$ and measurable $\hat{f} : \mathbb{R}^d \to \mathbb{C}$ satisfying for all $x \in \Omega$

$$\int_{\mathbb{R}^d} 2\pi |\xi|_{\Omega,x_0} \cdot |\hat{f}(\xi)| d\xi \leq C \text{ and } f(x) = c + \int_{\mathbb{R}^d} \left(e^{2\pi i \langle x,\xi\rangle} - e^{2\pi i \langle x_0,\xi\rangle}\right) \cdot \hat{f}(\xi) d\xi, \quad (5.14)$$

where $|\xi|_{\Omega,x_0} := \sup_{x\in\Omega} |\langle \xi, x - x_0\rangle|$. The class of all functions satisfying (5.14) is denoted by $\Gamma_C(\Omega, x_0)$.

For the case $\Omega = B_1^d$ with "base point" $x_0 = 0$, it holds that

$$|\xi|_{B_1^d,0} = \sup_{x\in B_1^d} |\langle \xi, x\rangle| = \|\xi\|_2 , \quad (5.15)$$

as in Definition 5.1.2. Equation (5.15) holds due to $|\langle \xi, x\rangle| \leq \|\xi\|_2 \cdot \|x\|_2 \leq \|\xi\|_2$ (Cauchy-Schwartz) and since $\left\langle \xi, \frac{\xi}{\|\xi\|_2} \right\rangle = \|\xi\|_2$, hence $\Gamma_C = \Gamma_C(B_1^d, 0)$. It can be shown that the choice of $x_0 \in \Omega$ is immaterial, as the respective norm changes at most by a factor of 2. Therefore writing $\Gamma_C(\Omega)$ instead of $\Gamma_C(\Omega, x_0)$, as the value of C is typically given and not to be chosen explicitely.

In order to proceed, a generalized version of Theorem 5.1.4 is stated underneath, see Proposition 2.2 in [12], based on the ideas given in [2].

Compared to Theorem 5.1.4, it is stated for the ReLU as activation function (instead of sigmoidal functions), bounded sets are considered (instead of the d-dimensional unit ball B_1^d) and the approximation error is measured in the ∞-norm (instead of the L^2-norm). It is worth mentioning that the statement underneath holds also true for sigmoidal functions in the L^2-norm (see Section V in [3]) and the $\infty-$norm (see [2]).

Theorem 5.2.2. *There exists $\kappa > 0$ such that for any $d \in \mathbb{N}$, any bounded set $\Omega \subset \mathbb{R}^d$ with nonempty interior, any $C > 0$, any $x_0 \in \Omega$, any $f \in \Gamma_C(\Omega, x_0)$ and any $N \in \mathbb{N}$ there exists a ReLU neural network Φ with*

$$\begin{aligned} L(\Phi_N) &= 2, \\ M(\Phi_N) &\leq 8N \cdot (d+2) + 1 \quad \textit{and} \\ \|f - R(\Phi_N)\|_\infty &\leq \frac{\kappa \cdot \sqrt{d} \cdot C}{\sqrt{N}}. \end{aligned}$$

The proof is omitted here and can be found in [12]. The constructed neural network Φ_N in the proof of the Theorem above has architecture $A(\Phi_N) = (d, 8N, 1)$, so $8N$ neurons in the hidden layer, therefore $M(\Phi_N) \leq d \cdot 8N + 8N + 8N \cdot 1 + 1 = 8N \cdot (d+2) + 1$.

Moreover, all weights can shown to be bounded by a constant $K = K(\Omega, x_0, C)$, so especially not dependent on N, and therefore independent of the approximation accuracy.

Note that if one would use the L^2-norm instead of the L^∞-norm in order to measure the approximation error the $\sqrt{d}$ on the right hand side could be omitted.

In order to proceed, the "Barron approximation set" $\Gamma A_C(\Omega)$ is defined as set of all functions which can be approximated with a shallow network with accuracy as in Theorem 5.2.2. This set can be seen as a scaled bigger version of $\Gamma_C(\Omega)$.

Definition 5.2.3. Let $d \in \mathbb{N}$ and $\Omega \subset \mathbb{R}^d$ bounded with nonempty interior. For $C > 0$ defining the Barron approximation set $\Gamma A_C(\Omega)$ as set of functions $f : \Omega \to \mathbb{R}$ such that for every $N \in \mathbb{N}$ there exists a shallow ReLU neural network Φ_N with N neurons in the hidden layer such that

$$\|f - R(\Phi_N)\|_\infty \leq \frac{C\sqrt{d}}{\sqrt{N}},$$

and where the weights of Φ_N can be bounded in absolute value by a constant adapted from to mentioned $K = K(\Omega, x_0, C)$ above.

For details about the constant K refer to [12], where it is also shown that there exists $\kappa_0 > 0$ such that for all $C > 0$ it holds that $\Gamma_C(\Omega, x_0) \subset \Gamma A_{\kappa_0 C}(\Omega)$. Therefore the definition and Theorem underneath, which are stated for $\Gamma A_C(\Omega)$ also hold for $\Gamma_{C \cdot \kappa_0^{-1}}(\Omega, x_0)$.

Now all ingredients to define sets with Barron class boundary are available.

Definition 5.2.4. Let $d \in \mathbb{N}, d \geq 2$, $B > 0$ and let $Q = [a, b] \subset \mathbb{R}^d$ be a rectangle, so $a, b \in \mathbb{R}^d$ with $a_i < b_i$ for $1 \leq i \leq d$. A function $f : Q \to \mathbb{R}$ is called a Barron Horizon function with constant B, if there are $1 \leq i \leq d$ and $f \in \Gamma A_B\big([a^{(i)}, b^{(i)}]\big)$ and $\theta \in \{\pm 1\}$ such that

$$F(x) = \mathbf{1}_{\theta x_i \leq f(x^{(i)})} \quad \forall\, x \in Q,$$

where $y^{(i)} = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_d)$ for $y \in \mathbb{R}^d$. The set of all Barron Horizon functions on a rectangle G is denoted by $\Gamma H_B(Q)$.
For given $M \in \mathbb{N}$ and $B > 0$, a compact set $\Omega \in \mathbb{R}^d$ is said to have Barron class boundary with constant B if there exists rectangles $Q_1, \ldots, Q_M \subset \mathbb{R}^d$ such that

- the rectangles have disjoint interiors, i.e. $Q_j^\circ \cap Q_k^\circ = \emptyset$ for $j \neq k$,
- the rectangles cover Ω, i.e. $\Omega \subset \bigcup_{j=1}^M Q_i$ and
- there exist Barron Horizon functions with constant B on each $Q_j \cap \Omega$, i.e. for $1 \leq j \leq M$ it holds that $\mathbf{1}_{Q_j \cap \Omega} \in \Gamma H_B(Q_j)$.

For given Ω a family of rectangles $(Q_j)_{j=1}^M$ is called an associated cover of Ω. Finally, the class of all sets which have Barron class boundary with constant B for M rectangles is denoted by $\Gamma_{B,M}(\mathbb{R}^d)$.

Next, the main result for this subsection, the approximation result for functions $\mathbf{1}_\Omega$ where Ω is a set with Barron class boundary, is shown. One notes that there is no exponential dependence on the input dimension d, therefore again the curse of dimensionality has been overcome. In [12] the statement is given for specific measures (α-tube compatible measures) and it is shown that for general probability measures the statement doesn't hold, see Section 6. However, in this thesis the statement is given for the Lebesgue measure, with is α-tube compatible with $\alpha = 1$, meaning that there exists a $C > 0$ such that for each measurable function $f : \mathbb{R}^{d-1} \to \mathbb{R}$ and each $1 \leq i \leq d$ and $\varepsilon \in (0, 1]$ it holds that

$$\mu\left(T_{f,\varepsilon}^{(i)}\right) \leq C \cdot \varepsilon \quad \text{where} \quad T_{f,\varepsilon}^{(i)} := \left\{x \in \mathbb{R}^d : \left|x_i - f(x^{(i)})\right| \leq \varepsilon\right\}. \tag{5.16}$$

Theorem 5.2.5. *Let $d, M, N \in \mathbb{N}, d \geq 2, B, C > 0$ and $\Omega \in \Gamma_{B,M}(\mathbb{R}^d)$. Then there exists a neural network Φ_N with*

$$\begin{aligned} L(\Phi_N) &= 4, \\ M(\Phi_N) &\leq 54MNd^2, \\ N(\Phi_N) &\leq 7M \cdot (N + d) \text{ and} \\ \mu\left(\{x \in \mathbb{R}^d : \mathbf{1}_\Omega(x) \neq R(\Phi_N)(x)\}\right) &\leq \frac{6MCB\sqrt{d^3}}{\sqrt{N}}, \end{aligned}$$

where μ denotes the Lebesgue measure. Moreover, $0 \leq R(\Phi_N)(x) \leq 1$ for all $x \in \mathbb{R}^d$, the architecture of Φ_N is given by $A(\Phi_N) = (d, M \cdot (N + 2d + 2), M \cdot (4d + 2), M, 1)$ and the absolute values of the weights can be bounded.

Proof (sketch). The neural network is constructed explicitly in 4 steps:
Step 1. Constructing neural networks approximating boundaries locally:
For an associated cover $(Q_j)_{j=1}^M$ fix $1 \leq m \leq M$ and let $Q_m := [a_{(m)}, b_{(m)}]$. By the assumption $\Omega \in \Gamma_{B,M}(\mathbb{R}^d)$ there exist $1 \leq i = i(m) \leq d$ and $\theta_m \in \{\pm 1\}$ and $f_m \in \Gamma A_B(Q_m^i)$ such that $\mathbf{1}_\Omega(x) = \mathbf{1}_{\theta_m x_i \leq f_m(x^{(i)})}$ for all $x \in Q_m$, where $Q_m^i := \prod_{j \neq i}[(a_j, (b_j]$. Moreover, it can be shown that one can assume $Q_m \subset [-R, R]^d$ for $1 \leq m \leq M$, where $R := \sup_{x \in \Omega} \|x\|_\infty$. Now, by Definition 5.2.3 there exist for $1 \leq m \leq M$ a shallow neural networks I_N^m with N neurons in the hidden layer such that

$$\|f_m - R(I_N^m)\|_\infty \leq \frac{\gamma}{\sqrt{N}},$$

where $\gamma := B\sqrt{d-1}$ and the absolute values of the weights can be bounded.
Step 2. Constructing neural networks approximating horizon functions:
Let

$$S_m := \{x \in Q_m : f_m(x^{(i)}) \geq \theta_m x_i\} \quad \text{and} \quad T_m := \{x \in Q_m : R(I_N^m)(x^{(i)}) \geq \theta_m x_i\},$$

then for $S_m \Delta T_m := (S_M \setminus T_m) \cup (T_M \setminus S_m)$ it holds that

$$S_m \Delta T_m \subset \left\{ x \in Q_m : \left| f_m(x^{(i)}) - \theta_m x_i \right| \leq \frac{\gamma}{\sqrt{N}} \right\}.$$

Since $\mathbf{1}_\Omega(x) = \mathbf{1}_{S_m}(x)$ for $x \in Q_m$ and (5.16) it follows that

$$\begin{aligned} \mu\Big(\{x \in Q_m : \mathbf{1}_\Omega(x) \neq \mathbf{1}_{T_m}(x)\}\Big) &= \mu\Big(\{x \in Q_m : \mathbf{1}_{S_m}(x) \neq \mathbf{1}_{T_m}(x)\}\Big) \\ &= \mu\big(S_m \Delta T_m\big) \leq \frac{C\gamma}{\sqrt{N}}. \end{aligned} \tag{5.17}$$

For $\delta > 0$ defining the the approximate Heaviside function $H_\delta : \mathbb{R} \to [0,1]$ by

$$H_\delta(x) := \begin{cases} 0 & \text{for } x \leq 0 \\ \frac{x}{\delta} & \text{for } 0 \leq x \leq \delta \\ 1 & \text{for } x \geq \delta \end{cases},$$

which is the realization of the ReLU neural network $\Phi_\delta(x) := \Big(\big((1,1)^T, (0,-\delta)^T\big), \big((\frac{1}{\delta}, -\frac{1}{\delta}), 0\big)\Big)$ as

$$R(\Phi_\delta) = \frac{1}{\delta}\rho(x) - \frac{1}{\delta}\rho(x-\delta) + 0 = H_\delta(x),$$

where ρ denotes the ReLU. Now an appropriate approximate Heaviside function applied to $R(I_N^m)(x^{(i)}) - \theta_m x_i$ is used to approximate $\mathbf{1}_{T_m}$. Therefore, note that for $\delta > 0$ and measurable $\phi : \mathbb{R}^{d-1} \to \mathbb{R}$ it holds that

$$\Big\{(t,u) \in \mathbb{R}^{d-1} \times \mathbb{R} : \mathbf{1}_{u \leq \phi(t)} \neq H_\delta(\phi(t) - u)\Big\} \subset \Big\{(t,u) \in \mathbb{R}^{d-1} \times \mathbb{R} : |\phi(t) - u| \leq \delta\Big\},$$

therefore picking $\delta = \frac{\gamma}{\sqrt{N}}$ and using (5.16) yields

$$\mu\Big(\{x \in Q_m : \mathbf{1}_{T_m}(x) \neq R(J_N^m)(x)\}\Big) \leq \frac{C\gamma}{\sqrt{N}}, \tag{5.18}$$

where J_N^m is chosen such that

$$R(J_N^m) = H_{\gamma/\sqrt{N}}\big(R(I_N^m)(x^{(i)}) - \theta_m x_i\big).$$

Moreover, clearly $0 \leq R(J_N^m) \leq 1$.
Step 3. Truncation and distinction between small and large rectangles:
Next, the realizations $R(J_N^m)$ are truncated in the sense that they are only supported on

Q_m via a ReLU neural network, respectively. This is done via an argument based on Lemma A.6 in [64].
Let $[a,b] = \prod_{i=1}^n [a_i, b_i]$ be a rectangle in $\mathbb{R}^d$. For $\varepsilon \in \big(0, \frac{1}{2} \cdot \min_{1\leq i\leq d}(b_i - a_i)\big)$ defining $[a+\varepsilon, b-\varepsilon] := \prod_{i=1}^d [a_i+\varepsilon, b_i-\varepsilon]$. Moreover, for $1 \leq i \leq d$ defining $t_i : \mathbb{R} \to \mathbb{R}$ by

$$t_i(u) := \begin{cases} 0 & \text{for } u \in \mathbb{R} \setminus [a_i, b_i] \\ 1 & \text{for } u \in [a_i+\varepsilon, b_i-\varepsilon] \\ \frac{u-a_i}{\varepsilon} & \text{for } u \in [a_i, a_i+\varepsilon] \\ \frac{b_i-u}{\varepsilon} & \text{for } u \in [b_i-\varepsilon, b_i] \end{cases}$$

and $\eta_\varepsilon : \mathbb{R}^d \times \mathbb{R}$ by

$$\eta_\varepsilon(x,y) := \rho\Big(\sum_{i=1}^d t_i(x_i) + \rho(y) - d\Big).$$

Then, for $y \in [0,1]$ and $x \in [a+\varepsilon, b-\varepsilon]$ it holds that

$$\eta_\varepsilon(x,y) = \rho(\rho(y)) = y$$

and if $x \in \mathbb{R}^d \setminus [a,b]$ (so $t_i(x_i) = 0$ for at least one coordinate direction i) it holds that $\eta_\varepsilon(x,y) = 0$ since

$$0 \leq \eta_\varepsilon(x,y) \leq \rho(d - 1 + \rho(y) - d) = \rho(y-1) = 0.$$

Therefore any function $g : \mathbb{R}^d \to [0,1]$ fulfills

$$\Big\{x \in \mathbb{R}^d : \eta_\varepsilon(x, g(x)) \neq \mathbf{1}_{[a,b]}(x) \cdot g(x)\Big\} \subset [a,b] \setminus [a+\varepsilon, b-\varepsilon].$$

Note that $t_i(u)$ can be implemented by the ReLU neural network with 2 layers given as

$$\Phi_{t_i} := \Big(\big((1,1,1,1)^T, (-a_i, -a_i-\varepsilon, -b_i+\varepsilon, -b_i)^T\big), \big(\frac{1}{\varepsilon}(1,-1,-1,1), 0\big)\Big),$$

as

$$R\left(\Phi_{t_i}\right)(u) = \frac{\rho(u-a_i) - \rho(u-a_i-\varepsilon) - \rho(u-b_i+\varepsilon) + \rho(u-b_i)}{\varepsilon} = t_i(u).$$

Therefore also $\eta_\varepsilon(x,y)$ can be implemented with the 3-layered ReLU neural network

$$\Phi_{\eta_\varepsilon} := \Phi_1^{id} \bullet \big(1,0\big) \bullet \big((\underbrace{1,\ldots,1}_{d-\text{times}}, -1), (-d)\big) \bullet FP\Big(\Phi_{t_1}, \ldots, \Phi_{t_d}, \Phi_1^{id} \bullet \big(1,0\big)\Big).$$

Moreover, note $0 \leq t_i \leq 1$ and $0 \leq \eta_\varepsilon(x,y) \leq \rho(\rho(y)) \leq 1$ for all $y \in [0,1]$.
Now distinguishing two cases:

If the rectangle Q_m has width along some coordinate direction i less than $\frac{2\gamma}{\sqrt{N}}$ ("small rectangle case"), then for a suitable (constant) function $g_m : \mathbb{R}^{d-1} \to \mathbb{R}$ it holds that

$Q_m \subset T^{(i)}_{g_m, 2\gamma/\sqrt{N}}$, therefore $\mu(Q_m) \leq \frac{2C\gamma}{\sqrt{N}} \leq \frac{2dC\gamma}{\sqrt{N}}$, where the last estimation is done to get the same bound as for the other case. Thus choosing L_N^m to be network with $R(L_N^m)(x,y) = 0$ for all $x \in \mathbb{R}^d, y \in \mathbb{R}$. Then

$$\mu\Big(\Big\{ x \in \mathbb{R}^d : \mathbf{1}_{Q_m}(x) \cdot R\big(J_N^m(x)\big) \neq R(L_N^m)\big(x, R\big(J_N^m(x)\big)\big) \Big\} \Big) \leq \mu(Q_m) \leq \frac{2dC\gamma}{\sqrt{N}}. \tag{5.19}$$

Otherwise ("big rectangle case"), for a rectangle Q_m with $Q_m = [a,b]$ it holds that $\frac{\gamma}{\sqrt{N}} \leq \frac{1}{2} \cdot \min_{1 \leq i \leq d}(b_i - a_i)$ and therefore $[a,b] \setminus [a + \frac{\gamma}{\sqrt{N}}, b - \frac{\gamma}{\sqrt{N}}]$ is contained in the union of $2d$ tubes with width $\frac{\gamma}{\sqrt{N}}$. Hence, choosing L_m^N to be the network with

$$R(L_N^m) := \eta_{\gamma/\sqrt{N}}$$

and it follows that

$$\mu\Big(\Big\{ x \in \mathbb{R}^d : \mathbf{1}_{Q_m}(x) \cdot R\big(J_N^m(x)\big) \neq R(L_N^m)\big(x, R\big(J_N^m(x)\big)\big) \Big\} \Big) \leq \frac{2dC\gamma}{\sqrt{N}}. \tag{5.20}$$

Note that the same bound is estimated for both cases. Moreover, in both cases the function $x \mapsto R(L_N^m)\big(x, R(J_N^m(x))\big)$ is supported on Q_m and vanishes on the boundary of Q_m.

Step 4. Final construction and error estimate:
It holds that

$$\begin{aligned}
&\mu\Big(\Big\{ x \in \mathbb{R}^d : \mathbf{1}_{\Omega \cap Q_m}(x) \neq R(L_N^m)\big(x, R\big(J_N^m(x)\big)\big) \Big\} \Big) \\
&\leq \mu\Big(\Big\{ x \in \mathbb{R}^d : R(L_N^m)\big(x, R\big(J_N^m(x)\big)\big) \neq \mathbf{1}_{Q_m}(x) \cdot R\big(J_N^m(x)\big) \Big\} \Big) \\
&\quad + \mu\Big(\{ x \in Q_m : R(J_N^m)(x) \neq \mathbf{1}_{T_m}(x) \} \Big) + \mu\Big(\{ x \in Q_m : \mathbf{1}_{T_m}(x) \neq \mathbf{1}_{\Omega}(x) \} \Big) \\
&\leq \frac{2dC\gamma}{\sqrt{N}} + \frac{C\gamma}{\sqrt{N}} + \frac{C\gamma}{\sqrt{N}} = \frac{2(d+1)C\gamma}{\sqrt{N}},
\end{aligned}$$

using (5.19) or (5.20) respectively, (5.18) and (5.17). Finally, defining the desired neural network Φ_N such that

$$R(\Phi_N)(x) = \sum_{m=1}^{m} R(L_N^m)\big(x, R\big(J_N^m(x)\big)\big).$$

Then, due to $\mathbf{1}_\Omega = \sum_{m=1}^M \mathbf{1}_{\Omega \cap Q_m}$ a.e., obtaining

$$\mu\Big(\Big\{ x \in \mathbb{R}^d : \mathbf{1}_\Omega(x) \neq R(\Phi_N)(x) \Big\} \Big) \leq \frac{2M(d+1)C\gamma}{\sqrt{N}} = \frac{2M(d+1)CB\sqrt{d-1}}{\sqrt{N}},$$

which can be simplified to $\frac{6MCB\sqrt{d^3}}{\sqrt{N}}$ as $(d+1)\sqrt{d-1} \leq (d+1)^{3/2} \leq (2d)^{3/2}$ and $2^{1+3/2} = 2^{5/2} < 6$.

Additionally, as for $1 \leq m \leq M$ it holds that $0 \leq R(J_N^m) \leq 1$ (see end of Step 2), also $\zeta_m(x) := R(L_N^m)\big(x, R\big(J_N^m(x)\big)\big)$ satisfies $0 \leq \zeta_m \leq 1$. Finally, as each ζ_m is supported on Q_m and vanishes on the boundary of Q_m, and since the rectangles have disjoint interior by Definition 5.2.4, also $0 \leq R(\Phi_N) \leq 1$ holds.

The exact examination of the architecture of Φ_N and the estimation of the bounds on the weights is omitted here, refer to [12], p.19-22. □

5.2.2 The Barron space

In this subsection, a different set compared to the Barron class defined in Definition 5.2.1, the so-called "Barron space" and a norm on this set defined. It is shown, that for functions in the Barron space, it is possible to overcome the curse of dimensionality for approximation of functions in the Barron space with ReLU neural networks. The main reference therefore is [21]. Moreover, the relationship between the Barron class and the Barron space is given, compare to Chapter 7 in [12].

Definition 5.2.6. Let $d \in \mathbb{N}$ and $\Omega \subset \mathbb{R}^d$ be nonempty and bounded. Denoting with $\mathcal{P}_d$ the set of all Borel probability measures on $\Lambda := \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$ and, as always, the ReLU with ρ, so $\rho(x) = \max\{0, x\}$. For $\mu \in \mathcal{P}_d$ and $x \in \Omega$ defining

$$\mu_\rho(x) := \int_\Lambda a \cdot \rho(\langle w, x\rangle + c) d\mu(a, w, c). \tag{5.21}$$

The Barron space $BS_\rho(\Omega)$ is defined as

$$BS_\rho(\Omega) := \left\{ f : \Omega \to \mathbb{R} : \exists\, \mu \in \mathcal{P}_d : \|\mu\|_\rho < \infty \text{ and } f(x) = \mu_\rho \,\forall x \in \Omega \right\},$$

where

$$\|\mu\|_\rho := \int_\Lambda |a| \cdot (\|w\|_1 + |c|) d\mu(a, w, c).$$

The Barron norm $\|\cdot\|_{BS_\rho(\Omega)}$ for $f \in BS_\rho(\Omega)$ is defined as

$$\|f\|_{BS_\rho(\Omega)} := \inf_{\mu \in \mathcal{P}_d} \left\{ \|\mu\|_\rho : f = \mu_\rho|_\Omega \right\}.$$

If Ω is fixed in the following writing BS_ρ instead of $BS_\rho(\Omega)$ and $\|\cdot\|_{BS_\rho}$ instead of $\|\cdot\|_{BS_\rho(\Omega)}$.

Remark 5.2.7. Remarks on the Definition above.

- Equation (5.21) can be seen as a continuum analog of the realization of a shallow ReLu neural network Φ, which is given by

$$R(\Phi)(x) = \frac{1}{N} \sum_{i=1}^{N} a_i \rho(\langle x, w_i\rangle + c_i),$$

for some $N \in \mathbb{N}$, where $a_i, c_i \in \mathbb{R}$ and $w_i \in \mathbb{R}^d$ for $1 \leq i \leq N$, see [21]. Therefore, (5.21) is the extension of the empirical parameter distribution of a network to any parameter distribution, see [22].

- In [21] the Barron norm on BS_ρ is defined for each $1 \leq p \leq \infty$ by

$$\|f\|_{BS_{\rho,p}} := \inf_{\mu \in \mathcal{P}_d} \left(\int_\Lambda |a|^p \cdot (\|w\|_1^p + |c|^p) d\mu(a,w,c) \right)^{1/p} \quad \text{for } 1 \leq p < \infty \text{ and}$$
$$\|f\|_{BS_{\rho,\infty}} := \inf_{\mu \in \mathcal{P}_d} \max_{(a,w,c) \in \operatorname{supp}(\mu)} |a| \cdot (\|w\|_1 + |c|).$$

  Denoting the Barron space with norm depending on p as above by $BS_{\rho,p}$. Then it holds that (Remark 1 in [21])

$$BS_{\rho,\infty} \subset \cdots \subset BS_{\rho,2} \subset BS_{\rho,1}.$$

  Moreover for $f \in BS_{\rho,1}$ it holds that $f \in BS_{\rho,\infty}$ with $\|f\|_{BS_{\rho,\infty}} = \|f\|_{BS_{\rho,1}}$, hence $BS_{\rho,1} = BS_{\rho,\infty}$. One follows for $1 \leq p < \infty$ (see Proposition 1 in [21])

$$BS_{\rho,p} = BS_{\rho,\infty} \quad \text{with} \quad \|f\|_{BS_{\rho,p}} = \|f\|_{BS_{\rho,\infty}}. \tag{5.22}$$

  Thereforem there is no need to distinguish between different spaces $BS_{\rho,p}$, they are all the same as $BS_{\rho,1}$, which is denoted by BS_ρ in Definition 5.2.6.

- It can be shown, that the Barron space BS_ρ with Barron norm $\|\cdot\|_{BS_\rho}$ is indeed a Banach space, so the term "space" is justified and that the Barron norm fulfills the norm axioms, see Theorem 2.7 in [23].

- Each $f \in BS_\rho$ is Lipschitz continuous, see Theorem 3.3 in [22].

- The ρ is denoted in BS_ρ as one could also define a Barron space similarly than above, but using another function than the ReLU in (5.21). See underneath for the Barron space related to the Heaviside function.

Now showing that for functions in the Barron space it is possible to overcome the curse of dimensionality.

Theorem 5.2.8. *Let $d \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$ be nonempty and bounded and μ be a probability measure on Ω. Then, for any $f \in BS_\rho(\Omega)$ and $N \in \mathbb{N}$ there exists a shallow ReLU neural network $\Phi^N = \Big(\big((w_1, \ldots, w_N)^T, c \big), \big(a^T, 0 \big) \Big)$ - where $w_1, \ldots, w_N \in \mathbb{R}^d, c, a \in \mathbb{R}^N$ - with N neurons in the hidden layer, so*

$$L(\Phi^N) = 2, \tag{5.23}$$
$$M(\Phi^N) \leq N \cdot (d+2) \quad \textit{such that} \tag{5.24}$$
$$\int_\Omega \left| f(x) - R(\Phi^N)(x) \right|^2 d\mu(x) \leq \frac{3 \|f\|_{BS_\rho}^2}{N}. \tag{5.25}$$

Moreover, the path norm $\|\cdot\|_{path}$ *of the shallow neural network* Φ^N *fulfills*

$$\left\|\Phi^N\right\|_{path} := \frac{1}{N}\sum_{i=1}^{N} |a_i| \cdot (\|w_i\|_1 + |c_i|) \leq 2\,\|f\|_{BS_\rho}\,. \tag{5.26}$$

Proof (sketch). The proof idea resembles the idea of Barron, cited in this thesis in the proof of Theorem 5.1.4, and is also based on sampling. A detailed proof can be found in [21]. Let $\varepsilon \in (0, \frac{1}{5})$ and $f \in BS_\rho$. Considering μ to be a probability measure such that

$$f(x) = \int_\Lambda a \cdot \rho(\langle w, x\rangle + c) d\mu(a, w, c) \quad \text{and}$$
$$\left(\int_\Lambda |a|^2 \cdot (\|w\|_1^2 + |c|^2) d\mu(a, w, c)\right) \leq (1+\varepsilon) \cdot \|f\|_{BS_\rho}^2\,,$$

which exists due to $f \in BS_\rho = BS_{\rho,2}$, see (5.22). Moreover, let $\phi(x, \theta) := a\rho(\langle x, w\rangle + c)$ with $\theta = (a, w, c) \sim \mu$. Then $\mathbb{E}_{\theta\sim\mu}\big(\phi(x,\theta)\big) = f(x)$. Let $\Theta := (\theta_i)_{i=1}^N$ be i.i.d. random variables drawn from $\mu(\cdot)$. Consider the following empirical average (which can be seen as realization of a shallow neural network)

$$\tilde{f}_N(x, \Theta) := \frac{1}{N}\sum_{i=1}^{N} \phi(x, \theta_i)$$

and let

$$\mathcal{E}(\Theta) := \mathbb{E}_x\big(|\tilde{f}_N(x,\Theta) - f(x)|^2\big) = \int_\Omega \left|f(x) - R(\Phi^N)(x)\right|^2 d\mu(x)$$

be the approximation error. One can show that Θ fulfills

$$\mathbb{E}_\Theta\big(\mathcal{E}(\Theta)\big) \leq \frac{(1+\varepsilon)\cdot\|f\|_{BS_\rho}^2}{N} \quad \text{and} \quad \mathbb{E}_\Theta\big(\,\|\Theta\|_{path}\,\big) \leq (1+\varepsilon)\cdot\|f\|_{BS_\rho}$$

Defining the events

$$E_1 := \left\{\mathcal{E}(\Theta) < \frac{3\,\|f\|_{BS_\rho}^2}{N}\right\} \quad \text{and} \quad E_2 := \left\{\|\Theta\|_{path} < 2\,\|f\|_{BS_\rho}\right\}$$

yields (using the Markov inequality, see e.g. 6.2.2 in [67])

$$\mathbb{P}\big(E_1\big) = 1 - \mathbb{P}\big(E_1^c\big) \geq 1 - \frac{\mathbb{E}_\Theta\big(\mathcal{E}(\Theta)\big)N}{3\,\|f\|_{BS_\rho}^2} \geq 1 - \frac{1+\varepsilon}{3} = \frac{2-\varepsilon}{3} \quad \text{and} \tag{5.27}$$

$$\mathbb{P}\big(E_2\big) = 1 - \mathbb{P}\big(E_2^c\big) \geq 1 - \frac{\mathbb{E}_\Theta\big(\,\|\Theta\|_{path}\,\big)}{2\,\|f\|_{BS_\rho}} \geq 1 - \frac{1+\varepsilon}{2} = \frac{1-\varepsilon}{2}. \tag{5.28}$$

Therefore

$$\begin{aligned}\mathbb{P}\big(E_1 \cap E_2\big) &= \mathbb{P}\big(E_1\big) + \mathbb{P}\big(E_2\big) - \mathbb{P}\big(E_1 \cup E_2\big) \\ &\geq \mathbb{P}\big(E_1\big) + \mathbb{P}\big(E_2\big) - 1 \geq \frac{2-\varepsilon}{3} + \frac{1-\varepsilon}{2} - 1 = \frac{1-5\varepsilon}{6} > 0, \end{aligned} \tag{5.29}$$

where (5.27) and (5.28) were used. Hence for any $\hat{\Theta} = (\hat{\theta}_i)_{i=1}^N = (\hat{a}_i, \hat{w}_i, \hat{c}_i)_{i=1}^N$ in $E_1 \cap E_2$, which exists due to (5.29), the neural network

$$\Phi^N := \Big(\big((\hat{w_1}, \dots, \hat{w_N})^T, \hat{c}\big), \big(\frac{1}{N} \cdot \hat{a}^T, 0\big)\Big)$$

with $R(\Phi^N) = \frac{1}{N}\sum_{i=1}^N \phi(x, \hat{\theta}_i) = \tilde{f}_N(x, \Theta)$ fulfills (5.23)-(5.26). □

As the last point in this subsection, for $\Omega \subset \mathbb{R}^d$ bounded with nonempty interior the relation between the Barron class $\Gamma_C(\Omega)$ for some $C > 0$ (see Definition 5.2.1) and the Barron space $BS_\rho(\Omega)$ (see Definition 5.2.6) and is stated. In fact, an even stronger result is shown, from which the relation between the Barron class and the Barron space can be deducted easily, see [12].

Definition 5.2.9. Let $d \in N$ and $\Omega \subset \mathbb{R}^d$ bounded with nonempty interior. For $s \geq 0$ defining the generalized s-Barron class as

$$G\Gamma^s(\Omega) := \left\{ f : \Omega \to \mathbb{R} : \exists F : \mathbb{R}^d \to \mathbb{C} : \|F\|_{\Gamma^s} < \infty, f(x) = \int_{\mathbb{R}^d} e^{2\pi i \langle x, \xi \rangle} F(\xi) d\xi \, \forall x \in \Omega \right\}$$

where

$$\|F\|_{\Gamma^s} := \int_{\mathbb{R}^d} (1 + 2\pi \|\xi\|_2)^s \, |F(\xi)| d\xi$$

and

$$\|f\|_{G\Gamma^s} := \inf \left\{ \|F\|_{\Gamma^s} : F : \mathbb{R}^d \to \mathbb{C} \text{ measurable}, f(x) = \int_{\mathbb{R}^d} e^{2\pi i \langle x, \xi \rangle} F(\xi) d\xi \, \forall x \in \Omega \right\}.$$

Note that if $f \in \Gamma_C(\Omega, x_0)$ for some $C > 0$ and $x_0 \in \Omega$, then $f \in G\Gamma^1(\Omega)$ and vice versa.

Theorem 5.2.10. *Let $d \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$ bounded with nonempty interior and $s \geq 0$.*

If $G\Gamma^s(\Omega) \subset BS_\rho(\Omega)$, then $s \geq 2$.
In particular, $G\Gamma^1(\Omega) \not\subset BS_\rho(\Omega)$.

The proof is omitted here, see Proposition 7.4 in [12]. With the note above Theorem 5.2.10 it follows that functions in the Barron class $\Gamma_C(\Omega)$ for some $C > 0$ do not necessary lie in the Barron space $BS_\rho(\Omega)$.

One notes that the Barron space BS_ρ depends on the function ρ appearing in (5.21). If one chooses another function, one gets a different (also called) Barron space with probably different properties. For the Heaviside function $H(x) = \mathbf{1}_{\mathbb{R}^+}(x)$ defining (compare to Definition 5.2.6)

$$\mu_H(x) := \int_\Lambda a \cdot H(\langle w, x\rangle + c) d\mu(a, w, c),$$

then the according Barron space $BS_H(\Omega)$ is defined as

$$BS_H(\Omega) := \left\{ f : \Omega \to \mathbb{R} : \exists\, \mu \in \mathcal{P}_d : \|\mu\|_\rho < \infty \text{ and } f(x) = \mu_\rho \,\forall x \in \Omega \right\},$$

where $\Omega \subset \mathbb{R}^d$ nonempty and bounded, $\Lambda := \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$, $\mu \in \mathcal{P}_d$ (the set of all Borel probability measures on Λ), $x \in \Omega$ and

$$\|\mu\|_H := \int_\Lambda |a| d\mu(a, w, c).$$

The Barron norm $\|\cdot\|_{BS_H(\Omega)}$ for $f \in BS_H(\Omega)$ is defined as

$$\|f\|_{BS_H(\Omega)} := \inf_{\mu \in \mathcal{P}_d} \{\|\mu\|_H : f = \mu_H|_\Omega\}.$$

It can be shown (Lemma 7.1 in [12]) that Barron spaces according to different functions have different properties and relations to each other.

Lemma 5.2.11. *Let $d \in \mathbb{N}$ and $\Omega \subset \mathbb{R}^d$ nonempty and bounded. The the following holds:*

1) $BS_\rho(\Omega) \hookrightarrow BS_H(\Omega)$, *where for U with nonempty interior the inclusion is strict.*
2) $G\Gamma^1(\Omega) \hookrightarrow BS_H(\Omega)$.
3) $G\Gamma^2(\Omega) \hookrightarrow BS_\rho(\Omega)$.

For details refer to [22, 24]. Also Barron spaces which are defined neither for the ReLU nor for the Heaviside functions and their properties are studied, see e.g. [21, 50].

6 Approximation of solutions of Partial Differential Equations (PDEs)

Nowadays, there are several results available which show that solutions of various types of PDEs can be approximated by (ReLU) neural networks without the curse of dimensionality, see e.g. the citations in [30].

Most of these works are based on emulating an existing method that does not suffer from the curse of dimensionality. This holds true for the case of first-order transport equations, which are hyperbolic and studied in Section 6.1, where the method of characteristics is the underlying method. In fact, it is shown that ReLU neural networks are capable of emulating the method of characteristics.

In Section 6.2 the parabolic linear Kolmogorov PDE is studied. The construction of the ReLU neural network overcoming the curse of dimensionality in this section is based on the the Feynman-Kac formula, which establishes a link between parabolic PDEs and stochastic processes, and random sampling like in the proofs of Chapter 5.

6.1 The parametric linear transport equation

In Subsection 6.1.1 the Cauchy problem for the linear transport equation is formulated and the unique existence of solutions for the homogeneous and non-homogeneous equation considered is given. The main reference for this part is [28].
Subsection 6.1.2 is based on [45] and shows that ReLU neural networks can approximate the solutions of the homogeneous and non-homogeneous linear transport equation. For the homogeneous problem the desired ReLU neural network is constructed explicitly. Due to the compositionality of the solution the curse of dimensionality can be overcome to some extent. Lastly, it is also shown for which setting the biggest advantage in comparison to direct approximation occurs.

6.1.1 Formulation and solution of the parametric linear transport equation

The definitions and results in this subsection are mainly as in Chapter 2 in [28]. Starting with the definition of the homogeneous linear transport equation.

Definition 6.1.1. Let $d, d', s, k \in \mathbb{N}$ and $T > 0$. For given vector field $V \in C^k([0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \mathbb{R}^d)$ and initial condition $u_0 \in C^s(\mathbb{R}^d)$ the homogeneous parametric linear transport equation is given by

$$\begin{cases} \partial_t u(t,x,\mu) + V(t,x,\mu) \cdot \nabla_x u(t,x,\mu) = 0, \\ u(0,x,\mu) = u_0(x), \end{cases} \tag{6.1}$$

where $t \in [0,T]$, $x \in \mathbb{R}^d$, $\mu \in [0,1]^{d'}$ and ∇_x denotes the gradient with respect to x.

Like every hyperbolic partial differential equation (see [75]) also (6.1) can be solved via the method of characteristics. The main idea is to consider curves (so-called "characteristic curves") that are defined such that the solution u (of in this case (6.1)) is constant along these curves. Henceforth, the solution at a point $(t,x,\mu) \in [0,T] \times \mathbb{R}^d \times [0,1]^{d'}$ equals the number evaluated at the origin of this curve.

Definition 6.1.2. The characteristic curve of the transport operator $\partial_t + V(t,x,\mu) \cdot \nabla_x$ passing through x at time $s = t$ is given by the set $\{(s, \gamma(s)) : s \in [0,T]\}$, where γ is the solution of the characteristic systems of ordinary differential equations

$$\begin{cases} \dot{\gamma}(s) = V(s, \gamma(s), \mu), \\ \gamma(t) = x. \end{cases} \tag{6.2}$$

In order to ensure that the method of characteristics works, the following assumptions on the vector field V are made for $d, d' \in \mathbb{N}$ and $T > 0$:

$$\text{For some } k \in \mathbb{N} \text{ it holds that } V \in C^k([0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \mathbb{R}^d) \text{ and} \tag{6.3}$$

$$\exists\, C > 0 \text{ s.t. } \|V(t,x,\mu)\|_{C^k} \leq C(1 + \|x\|_2) \,\forall\, (t,x,\mu) \in [0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \tag{6.4}$$

where $\|V(t,x,\mu)\|_{C^k}$ stands for $\|V(t,x,\mu)\|_{C^k([0,T]\times\mathbb{R}^d\times[0,1]^{d'},\mathbb{R}^d)}$.

The assumptions (6.3) and (6.4) on the vector field V yield global existence of the characteristic curves and characterises their regularity, compare to Theorem 3.3 in [45] and Theorem 2.2.2 in [28].

Theorem 6.1.3. *For $d, d' \in \mathbb{N}$ and $T > 0$ let the vector field V fulfill (6.3) and (6.4) for some $k \in \mathbb{N}$. Then, for all $(t,x,\mu) \in [0,T] \times \mathbb{R}^d \times [0,1]^{d'}$ the system (6.2) has a unique solution $\gamma \in C^k([0,T])$. Furthermore, the map*

$$X(s,t,x,\mu) := \gamma(s)$$

fulfills $X \in C^k([0,T] \times [0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \mathbb{R}^d)$.

The proof is omitted here, refer to [45, 28].

It is not surprising that the map X will be approximated by a ReLU neural network later on. In order to use statements from Chapter 2 about approximation with ReLU neural networks the following statement is given, compare to Proposition 3.4 in [45]:

Proposition 6.1.4. *For $d, d' \in \mathbb{N}$ and $T > 0$ let the vector field V fulfill (6.3) and (6.4) for some $k \in \mathbb{N}$. Then, for every $K \subset \mathbb{R}^d$ compact and nonempty there exists a constant $G = G(k, d, T, K, \|V\|_{C^k([0,T]\times K\times[0,1]^{d'},\mathbb{R}^d)})$ such that*

$$\|X\|_{C^k([0,T]\times[0,T]\times K\times[0,1]^{d'})} \leq G.$$

For the proof referring to Appendix A in [45].

The next theorem states the unique existence of a solution for the homogeneous parametric linear transport equation (6.1). Moreover, the theorem shows the compositional structure of the solution, namely as composition of the initial condition u_0 with the solution of the characterstic system of ODEs (6.2) evaluated at $s = 0$, compare to Theorem 2.2.4 in [28], where also the proof is stated.

Theorem 6.1.5. *Let $d, d', k, s \in \mathbb{N}$ and $T > 0$ and let the vector field V fulfill the assumptions* (6.3) *and* (6.4). *Furthermore, let $u_0 \in C^s(\mathbb{R}^d)$. Then the homogeneous parametric linear transport equation* (6.1) *has a unique solution $u \in C^{min\{s,k\}}([0,T] \times \mathbb{R}^d \times [0,1]^{d'})$ which is given by*

$$u(t, x, \mu) = u_0(X(0, t, x, \mu)). \tag{6.5}$$

If the initial condition u_0 would not be (once) differentiable, one could considerate a weak formulation of (6.1) and get for $u_0 \in L^\infty(\mathbb{R}^d)$ a global weak solution $u \in L^\infty([0,T] \times \mathbb{R}^d \times [0,1]^{d'})$. In this thesis the idea of a weak notion is not carried on, refer to [28].

The following statement is the analogon to Theorem 6.1.5 for the non-homogeneous parametric linear transport equation, compare to Theorem 3.9 in [60].

Theorem 6.1.6. *Let $d, d', k, s, s' \in \mathbb{N}$ and $T > 0$ and let the vector field V fulfill the assumptions* (6.3) *and* (6.4). *Further, let $u_0 \in C^s(\mathbb{R}^d)$ and $f \in C^{s'}([0,T] \times \mathbb{R}^d \times [0,1]^{d'})$. Then the non-homogeneous parametric linear transport equation, for $t \in [0,T]$, $x \in \mathbb{R}^d$ and $\mu \in [0,1]^{d'}$,*

$$\begin{cases} \partial_t u(t, x, \mu) + V(t, x, \mu) \cdot \nabla_x u(t, x, \mu) = f(t, x, \mu), \\ u(0, x, \mu) = u_0(x), \end{cases} \tag{6.6}$$

has a unique solution $u \in C^{min\{s,s',k\}}([0,T] \times \mathbb{R}^d \times [0,1]^{d'})$ which is given by

$$u(t, x, \mu) = u_0(X(0, t, x, \mu)) + \int_0^t f(s, X(s, t, x, \mu), \mu) ds. \tag{6.7}$$

6.1.2 Approximation of the solution of the linear parametric transport equation

In the last subsection the theoretical framework, meaning well-posedness and unique solvability of the (non-)homogeneous linear transport equation (6.1) and (6.6) for a vector field V fulfilling assumptions (6.3) and well as (6.4) and s-times differentiable initial condition u_0 was established.

Now coming to the approximation of the solution of (6.1) and (6.6) with ReLU neural networks, where the compositional structure (initial condition u_0 composed with a flow along characteristic curves) of the solutions (6.5) and (6.7) of Theorem 6.1.5 and 6.1.6,

respectively. In order to approximate the solution with ReLU neural network reasonably well, assuming that the initial condition u_0 can be approximated by ReLU neural networks reasonably well, see Definition 6.1.7

This subsection is content-wise based on Chapter 4 in [45], whereas different to the theory established/cited there statements from Section 2.3 about approximation of C^s-functions with ReLU neural networks are used.

Definition 6.1.7. For $d \in \mathbb{N}$ and $r > 0$ a function $f \in L^\infty(\mathbb{R}^d)$ is called r-approximable by neural networks if, for every nonempty and compact set $K \in \mathbb{R}^d$ there exists a constant $c = c(K, r, f) > 0$ such that for every $\varepsilon \in (0, 1/2)$ there exists a neural network Φ_f^ε such that

$$\begin{aligned} L(\Phi_f^\varepsilon) &\leq c \log_2\left(\frac{1}{\varepsilon}\right), \\ M(\Phi_f^\varepsilon) &\leq c\varepsilon^{-\frac{1}{r}} \log_2\left(\frac{1}{\varepsilon}\right) \quad \text{and} \\ \left\| f - R(\Phi_f^\varepsilon) \right\|_{L^\infty(K)} &\leq \varepsilon. \end{aligned}$$

Remark 6.1.8. From Theorem 2.3.11 and Remark 2.3.12 is follows that $f \in C^s(K)$ is r-approximable for $r = s/d$ and nonempty, bounded $K \subset \mathbb{R}^d$.

Now it is possible to state the main theorems of this section for the approximation of the solution of the linear transport equation, starting with the homogeneous case, compare to Theorem 4.4 in [45].

Theorem 6.1.9. *Let $d, d', k \in \mathbb{N}$, $T, r > 0$ and let the vector field V fulfill the assumptions* (6.3) *and* (6.4)*. Further, let the initial condition $u_0 \in C^1(\mathbb{R}^d)$ be r-approximable by neural networks. Let $u \in C^1([0,T] \times \mathbb{R}^d \times [0,1]^{d'})$ denote the unique solution* (6.5) *of the homogeneous parametric linear transport equation* (6.1)*, so of*

$$\begin{cases} \partial_t u(t,x,\mu) + V(t,x,\mu) \cdot \nabla_x u(t,x,\mu) = 0, \\ u(0,x,\mu) = u_0(x), \end{cases}$$

for $t \in [0,T]$, $x \in \mathbb{R}^d$ and $\mu \in [0,1]^{d'}$. Then, for every $\varepsilon \in (0, 1/2)$ and every compact and nonempty $K \subset \mathbb{R}^d$, there exists a ReLU neural network $\Phi^{u,\varepsilon}$ with input dimension $D := 1 + d + d'$, such that for the restriction $\overline{u} := u|_{[0,T] \times K \times [0,1]^{d'}}$ it holds that, for $c = c(d, r, k, K, T, \|V\|_{C^k([0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \mathbb{R}^d)}, u_0) > 0$,

$$L(\Phi^{u,\varepsilon}) \leq c \log_2\left(\frac{1}{\varepsilon}\right), \tag{6.8}$$

$$M(\Phi^{u,\varepsilon}) \leq c\left(\varepsilon^{-\frac{1}{r}} + \varepsilon^{-\frac{D}{k}}\right) \log_2\left(\frac{1}{\varepsilon}\right) \quad \text{and} \tag{6.9}$$

$$\|\overline{u} - R(\Phi^{u,\varepsilon})\|_{L^\infty([0,T] \times K \times [0,1]^{d'})} \leq \varepsilon. \tag{6.10}$$

Proof. The unique solution $u \in C^1([0,T] \times \mathbb{R}^d \times [0,1]^{d'})$ of (6.1) is (see Theorem 6.1.5) given by

$$u(t,x,\mu) = u_0(X(0,t,x,\mu)).$$

Moreover, $X \in C^k([0,T] \times [0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \mathbb{R}^d)$ (see Theorem 6.1.3), hence

$$\tilde{X} := X(0,\cdot,\cdot,\cdot) \in C^k([0,T] \times \mathbb{R}^d \times [0,1]^{d'}, \mathbb{R}^d)$$

The idea of the proof is to first approximate u_0 and $\tilde{X}$ with ReLU neural networks separately using Theorem 2.3.11 and then sparsely concatenate these two neural networks.

Step 1. Approximating u_0 and $\tilde{X}$ with ReLU neural networks:
Let $\varepsilon \in (0, 1/2)$. The mentioned Theorem 2.3.11 holds for C^k-functions defined on compact sets (see also Remark 2.3.12), henceforth the function $\hat{X}$ is restricted to $U := [0,T] \times K \times [0,1]^{d'}$ and the function u_0 is restricted to $B_G^d(0)$ (the closed ball with radius G around 0 in $\mathbb{R}^d$), where G is an upper bound for $\|X\|_{C^0([0,T]\times K\times[0,1]^{d'})}$, which exists due to Proposition 6.1.4.

Since u_0 is r-approximable by Definition 6.1.7 there exists $c_1 = c_1(d,r,T,K,u_0) > 0$ and a neural network $\Phi_{u_0}^{\delta_1}$ for $\delta_1 := \varepsilon/2$ with with d-dimensional input, such that

$$\begin{aligned} L(\Phi_{u_0}^{\delta_1}) &\leq c_1 \log_2\left(\frac{1}{\delta_1}\right), \\ M(\Phi_{u_0}^{\delta_1}) &\leq c_1 \delta_1^{-\frac{1}{r}} \log_2\left(\frac{1}{\delta_1}\right) \quad \text{and} \\ \left\| u_0 - R(\Phi_{u_0}^{\delta_1}) \right\|_{L^\infty(B_G(0))} &\leq \delta_1. \end{aligned}$$

By the mentioned statements Theorem 2.3.11 and Remark 2.3.12 from Chapter 2 there exists a constant $c_2 = c_2(d,T,k,K,\|V\|_{C^k([0,T]\times\mathbb{R}^d\times[0,1]^{d'},\mathbb{R}^d)}) > 0$, and a neural network $\Phi_{\tilde{X}}^{\delta_2}$ for $\delta_2 := \varepsilon/(2L_{u_0})$, where $L_{u_0} := \sup_{x\neq y\in U} \|u_0(x) - u_0(y)\|_2 / \|x-y\|_2 < \infty$, which exists due to u_0 is differentiable, with $D := 1 + d + d'$-dimensional input such that

$$\begin{aligned} L(\Phi_{\tilde{X}}^{\delta_2}) &\leq c_2 \log_2\left(\frac{1}{\delta_2}\right), \\ M(\Phi_{\tilde{X}}^{\delta_2}) &\leq c_2 \delta_2^{-\frac{D}{k}} \log_2\left(\frac{1}{\delta_2}\right) \quad \text{and} \\ \left\| \tilde{X} - R(\Phi_{\tilde{X}}^{\delta_2}) \right\|_{L^\infty(U)} &\leq \delta_2. \end{aligned}$$

Step 2. Defining $\Phi^{u,\varepsilon}$ and showing the desired properties (6.8)-(6.10):
As indicated before setting now $\Phi^{u,\varepsilon} := \Phi_{u_0}^{\delta_1} \odot \Phi_{\tilde{X}}^{\delta_2}$. This yields, using triangle inequality

and the approximation property of $\Phi_{u_0}^{\delta_1}$ and $\Phi_{\tilde{X}}^{\delta_2}$, that

$$\begin{aligned}
&\|u - R\left(\Phi^{u,\varepsilon}\right)\|_{L^\infty(U)} \\
=&\left\|u_0 \circ \tilde{X} - R\left(\Phi_{u_0}^{\delta_1} \odot \Phi_{\tilde{X}}^{\delta_2}\right)\right\|_{L^\infty(U)} \\
=&\left\|u_0 \circ \tilde{X} - R\left(\Phi_{u_0}^{\delta_1}\right) \circ R\left(\Phi_{\tilde{X}}^{\delta_2}\right)\right\|_{L^\infty(U)} \\
\leq&\left\|u_0 \circ \tilde{X} - u_0 \circ R\left(\Phi_{\tilde{X}}^{\delta_2}\right)\right\|_{L^\infty(U)} + \left\|u_0 \circ R\left(\Phi_{\tilde{X}}^{\delta_2}\right) - R\left(\Phi_{u_0}^{\delta_1}\right) \circ R\left(\Phi_{\tilde{X}}^{\delta_2}\right)\right\|_{L^\infty(U)} \\
\leq&L_{u_0}\left\|\tilde{X} - R\left(\Phi_{\tilde{X}}^{\delta_2}\right)\right\|_{L^\infty(U)} + \left\|u_0 - R\left(\Phi_{u_0}^{\delta_1}\right)\right\|_{L^\infty(B_G(0))} \leq L_{u_0}\delta_2 + \delta_1 = \varepsilon,
\end{aligned}$$

so (6.10) holds. Moreover, due to (2.16) and the bounded number of weights of $\Phi_{u_0}^{\delta_1}$ and $\Phi_{\tilde{X}}^{\delta_2}$ it holds that

$$\begin{aligned}
M(\Phi^{u,\varepsilon}) = M(\Phi_{u_0}^{\delta_1} \odot \Phi_{\tilde{X}}^{\delta_2}) &\leq 2 \cdot \left(M(\Phi_{u_0}^{\delta_1}) + M(\Phi_{\tilde{X}}^{\delta_2})\right) \\
&\leq 2 \cdot \left(c_1 \delta_1^{-\frac{1}{r}} \log_2\left(\frac{1}{\delta_1}\right) + c_2 \delta_2^{-\frac{D}{k}} \log_2\left(\frac{1}{\delta_2}\right)\right) \\
&\leq c_3 \left(\varepsilon^{-\frac{1}{r}} + \varepsilon^{-\frac{D}{k}}\right) \log_2\left(\frac{1}{\varepsilon}\right),
\end{aligned}$$

for an appropriate choice of $c_3 > 0$. For the layers, using (2.15) and that number of layers of $\Phi_{u_0}^{\delta_1}$ and $\Phi_{\tilde{X}}^{\delta_2}$ is bounded, yields

$$\begin{aligned}
L(\Phi^{u,\varepsilon}) = L(\Phi_{u_0}^{\delta_1} \odot \Phi_{\tilde{X}}^{\delta_2}) &= L(\Phi_{u_0}^{\delta_1}) + L(\Phi_{\tilde{X}}^{\delta_2}) \\
&\leq c_1 \log_2\left(\frac{1}{\delta_1}\right) + c_2 \log_2\left(\frac{1}{\delta_2}\right) \\
&\leq c_4 \log_2\left(\frac{1}{\varepsilon}\right),
\end{aligned}$$

for an appropriate choice of $c_4 > 0$. With

$$c = c(d, r, k, K, T, \|V\|_{C^k([0,T]\times\mathbb{R}^d\times[0,1]^{d'},\mathbb{R}^d)}, u_0) := \max\{c_3, c_4\} > 0$$

equations (6.8) and (6.9) hold. □

Remark 6.1.10. Showing in which case the Theorem above is substantially better than direct approximation without using the compositionality of the solution (6.5).

- The power of Theorem 6.1.9 can be seen if V is substantially smoother than the initial condition. So, if one considers $d \leq s \ll D \leq k$, where d denotes the space dimension, s the smoothness of the initial condition u_0, $D = 1 + d + d'$ the total dimension (time, space and additional parameter) and k the smoothness of V, then a direct approximation via Theorem 2.3.11 of the solution $u \in C^s([0,T] \times$

$K \times [0,1]^{d'}$) on bounded and nonempty $K \in \mathbb{R}^d$ with a ReLU neural network Φ_1 would require

$$M(\Phi_1) \leq c\varepsilon^{-D/s} \log_2(1/\varepsilon) \tag{6.11}$$

many weights. In contrast, using Theorem 6.1.9 yields a neural network Φ_2 with only

$$M(\Phi_2) \leq c\left(\varepsilon^{-1/r} + \varepsilon^{-D/k}\right) \log_2(1/\varepsilon) \leq 2c(1/\varepsilon)\log_2(1/\varepsilon) \tag{6.12}$$

needed weights, due to $d \leq s$ (and therefore $r = s/d \geq 1$) and $D \leq k$ (and therefore $D/k \leq 1$). Note that for $s \ll D$ the difference between (6.11) and (6.12) is substantially.

- As mentioned, the compositionality yields the required better approximation result in Theorem 6.1.9 compared to the statements in Chapter 2. In fact, the solution $u(t,x,\mu) = u_0(X(0,t,x,\mu))$ of the homogeneous linear transport equation (6.1) can be seen as a compositional function (as explained in Chapter 3) with $f_1 = \tilde{X}$ and $f_2 = u_0$ in a more generalized setting, where the constituent functions may have different orders of smoothness and the functions involved are not defined on $[-1,1]^{d_i}$ for some $d_i \in \mathbb{N}$ but on a suitable compact subset $K \in \mathbb{R}^d$.

A statement similar to Theorem 6.1.9 also holds for the non-homogeneous parametric linear transport equation (6.6), see Theorem 4.7 in [45]. It is shown for two different types of source terms. Henceforth let, for $d, d' \in \mathbb{N}$ and $T > 0$, the vector field V fulfill (6.3) and (6.4) for some $k \in \mathbb{N}$ and moreover assuming that one of the following properties hold:

$$(i) \quad f(t,x,\mu) = f_1(t,x) \in C^{s'}([0,T] \times \mathbb{R}^d) \quad \text{for } s' := \lceil (d+1)k/D \rceil, \tag{6.13}$$

$$(ii) \quad f(t,x,\mu) = f_2(t,x,\mu) \in C^{s'}([0,T] \times \mathbb{R}^d \times [0,1]^{d'}), \quad \text{for } s' := k, \tag{6.14}$$

so assuming that f is very regular if it depends on μ and much less regularity is sufficient if f doesn't depend on μ.

Theorem 6.1.11. *Let $d, d', k \in \mathbb{N}$, $T, r > 0$ and let the vector field V fulfill the assumptions* (6.3) *and* (6.4). *Further, let the initial condition $u_0 \in C^1(\mathbb{R}^d)$ be r-approximable by neural networks and let f and $s' \in \mathbb{N}$ as in* (6.13) *or* (6.14). *Let $u \in C^1([0,T] \times \mathbb{R}^d \times [0,1]^{d'})$ denote the unique solution* (6.7) *of the non-homogeneous parametric linear transport equation* (6.6), *so of*

$$\begin{cases} \partial_t u(t,x,\mu) + V(t,x,\mu) \cdot \nabla_x u(t,x,\mu) = f(t,x,\mu), \\ u(0,x,\mu) = u_0(x), \end{cases}$$

for $t \in [0,T]$, $x \in \mathbb{R}^d$ and $\mu \in [0,1]^{d'}$. Then, for every $\varepsilon \in (0,1/2)$ and every compact and nonempty $K \subset \mathbb{R}^d$, there exists a ReLU neural network $\Phi^{u,\varepsilon}$ with input dimension

$D := 1 + d + d'$, such that for the restriction $\overline{u} := u|_{[0,T]\times K\times[0,1]^{d'}}$ it holds that, for $c = c(d, r, D, k, K, T, \|V\|_{C^k([0,T]\times\mathbb{R}^d\times[0,1]^{d'},\mathbb{R}^d)}, u_0, \|f\|_{C^{s'}}) > 0$,

$$L(\Phi^{u,\varepsilon}) \leq c \log_2\left(\frac{1}{\varepsilon}\right), \tag{6.15}$$

$$M(\Phi^{u,\varepsilon}) \leq c\left(\varepsilon^{-\frac{1}{r}} + \varepsilon^{-\frac{D+1}{k}-1}\right)\log_2\left(\frac{1}{\varepsilon}\right) \quad \textit{and} \tag{6.16}$$

$$\|\overline{u} - R(\Phi^{u,\varepsilon})\|_{L^\infty([0,T]\times K\times[0,1]^{d'})} \leq \varepsilon. \tag{6.17}$$

Proof (sketch). The unique solution of (6.6) is (see Theorem 6.1.6) given by

$$u(t, x, \mu) = u_0(X(0, t, x, \mu)) + \int_0^t f(s, X(s, t, x, \mu), \mu)ds. \tag{6.18}$$

A ReLU neural network approximating $(u_0 \circ X)(0, \cdot, \cdot, \cdot)$ was constructed in the proof of Theorem 6.1.5. By Theorem 2.3.11 there exists a ReLU neural network which approximates f. For a ReLU neural network which approximates the antiderivative (to be more precise, the left Riemann sum) of f with respect to the first coordinate, the final construction of the desired network and the properties (6.15)-(6.17) refer to the proof of Theorem 4.7 in [45]. □

Note that if $u_0 \in C^s(\mathbb{R}^d)$, then by Remark 6.1.7 the r in Theorem 6.1.11 can be replaced by s/d.

Several extensions (e.g. the non-linear setting) of the homegeneous 6.1 and non-homogeneous 6.6 linear transport equation are shown in [45].

In general, the results shown in this chapter for the linear transport equation are based on the method of characteristics and can be adjusted to hold for the class of hyperbolic PDEs, where the solution can be written as a composition including the initial condition u_0 and can be found using this method, see [75].

6.2 The linear Kolmogorov PDE

In Subsection 6.2.1 the linear Kolmogorov PDE is defined and the existence and uniqueness of a (viscosity) solution to it for the case of affine diffusion and drift is discussed. The main references for this subsection are [30, 31].
Subsection 6.2.2 is based on [8, 30] (see also Chapter 4 in [10]) and shows that ReLU neural networks can approximate the unique viscosity solution of the linear Kolmogorov PDE with affine diffusion and drift under some, not to restrictive, assumptions on the setting without the curse of dimensionality. A bit simplified one could say that if the initial condition can be approximated without the curse of dimensionality also the viscosity solution of the linear Kolmogorov PDE with affine diffusion and drift has this property.

6.2.1 Formulation and solutions of the linear Kolmogorov PDE

Starting with the definiton of the linear Kolmogorov PDE.

Definition 6.2.1. Let $d \in \mathbb{N}$ and $T > 0$. For given initial value $u_0 \in C(\mathbb{R}^d, \mathbb{R})$, $\mu \in C(\mathbb{R}^d, \mathbb{R}^d)$ (drift coefficient) and $\sigma \in C(\mathbb{R}^d, \mathbb{R}^{d\times d})$ (diffusion coeffcient) the linear Kolmogorov equation is given by

$$\begin{cases} \partial_t u(t,x) = \frac{1}{2} tr\left(\sigma(x)\sigma(x)^T \nabla_x^2 u(t,x)\right) + \langle \mu(x), \nabla_x u(t,x)\rangle, \\ u(0,x,) = u_0(x), \end{cases} \tag{6.19}$$

for $t \in [0,T]$ and $x \in \mathbb{R}^d$, where ∇_x and ∇_x^2 denote the gradient and Hessian with respect to x, respectively and $tr(A)$ denotes the trace, so the sum of the diagonal elements of a quadratic matrix, so for $A = (a_{ij})_{i,j=1}^d$: $tr(A) = \sum_{i=1}^d a_{ii}$. The term with the trace in (6.19) can be seen as Frobenius inner product of the matrices $\sigma(x)\sigma(x)^T$ and $\nabla_x^2 u(t,x)$.

Remark 6.2.2. Several remarks to the linear Kolmogorov equation:

a) Equation (6.19) is named after the author of [42], where it first appeared in a slightly different way.

b) The heat equation

$$\partial_t u(t,x) = \Delta_x u(t,x),$$

where Δ_x denotes the Laplace operator, so $\Delta_x u = \sum_{i=1}^d \frac{\partial^2 u}{\partial x_i^2}$, is also covered by (6.19) choosing $\sigma(x)$ to be the identity matrix and $\mu(x) = 0$.

c) Also the Black-Scholes equation in option pricing is covered by (6.19) for the case the of affine σ and μ. The initial condition u_0 can often (e.g. for the European put option pricing problem) be represented as a composition of minima, maxima, and linear combinations such as

$$u_0(x) = \min\{\max\{D - y^T x, 0\}, D\}, \tag{6.20}$$

with suitable $D > 0$ and $y \in \mathbb{R}^d$. The initial condition u_0 as in (6.20) can be exactly rebuild by the ReLU neural network

$$\Phi_{u_0}^{\mathrm{BS}} = \big((-y^T, D), (-1, D), (-1, D)\big), \tag{6.21}$$

which fulfills

$$R\left(\Phi_{u_0}^{\mathrm{BS}}\right) = D - \rho(D - \rho(D - y^T x)) = \min\{\max\{D - y^T x, 0\}, D\} = u_0(x),$$

since $\min\{z, D\} = D - \rho(D - z) =: \psi(z, D)$ (see next line) for given $D > 0, z \in \mathbb{R}$ and where the ReLU is denoted by ρ. For $z \geq D$ it holds that $\psi(z, D) = D - 0 =$

$D = \min\{z, D\}$, for $z < D$ it holds that $\psi(z, D) = D - (D - z) = z = \min\{z, D\}$. By Definition of $\Phi_{u_0}^{\mathrm{BS}}$ it holds that

$$L\left(\Phi_{u_0}^{\mathrm{BS}}\right) = 3 \quad \text{and} \quad M\left(\Phi_{u_0}^{\mathrm{BS}}\right) = d + 5 \leq 6d. \tag{6.22}$$

Moreover, a large number of relevant initial conditions can be shown to be approximable by ReLU neural networks without the curse of dimensionality, compare to [8, 30].

d) If a classical solution to (6.19) exists depends heavily on σ, μ and u_0. Considering now smooth μ and σ satisfying growth conditions. Then, if $\sigma = 0$ the equation is a first order PDE and if u_0 is smooth, then the solution is smooth, so (6.19) is regularity-preserving. For $\sigma \neq 0$ in some (to be more precise: hypoelliptic) cases (6.19) has a infinitely often differentiable solution, even if u_0 is only continuous, so the equation has a smoothing effect. However, for non-hypoelliptic PDEs of form (6.19) it can be shown that even if u_0 is C^∞ on a compact domain it can happen that the solution is not once differentiable for any time $t > 0$ – the equation is said to have a roughening effect. For details refer to [31].

Typically one wants to know the value of a solution of (6.19) at time $T > 0$, which is in a lot of cases not a classical solution (see point d) above). This yields to the concept of viscosity solutions, a type of weak solutions, compare to Section 4.1 in [31]. Proposition 3.4 in [30] shows that under some assumptions on the setting there exists a unique viscosity solution of (6.19).

6.2.2 Approximation of the solution of the linear Kolmogorov PDE

The goal for the remaining chapter here is to show that under specific given assumptions on the setting the unique solution of (6.19) can be approximated by a ReLU neural network overcoming the curse of dimensionality for affine σ and μ whenever u_0 can be approximated without the curse of dimensionality. Note that this is the case for the setting described in c) in Remark 6.2.2.

Starting with two properties about affine functions, compare to Corollary 2.10 and 2.11 in [30], which are stated there in a generalized way.

Lemma 6.2.3. *For $d \in \mathbb{N}$ let $\mu : \mathbb{R}^d \to \mathbb{R}^d$ be an affine function, so $\mu(x) = Ax + b$ for $A \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$. Then exists $c > 0$ such that for all $x \in R^d$ it holds that*

$$\|\mu(x)\|_2 \leq c(1 + \|x\|_2),$$

where $\|\cdot\|_2$ denotes the Euclidean norm in $\mathbb{R}^d$.

Lemma 6.2.4. *For $d \in \mathbb{N}$ let $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ be an affine function. Then exists $c > 0$ such that for all $x \in R^d$ it holds that*

$$\|\sigma(x)\|_2 \leq c(1 + \|x\|_2),$$

where $\|\cdot\|_2$ denotes the Frobenius norm on $\mathbb{R}^{d\times d}$ for $d \times d$-dimensional input and the Euclidean norm in $\mathbb{R}^d$ for d-dimensional input.

Moreover, for a given neural network Φ with architecture $A(\Phi) = (N_0, N_1, \dots, N_L)$ denote by

$$W(\Phi) = \sum_{\ell=1}^{L} (N_\ell \cdot N_{\ell-1} + N_\ell),$$

the maximum number of parameters of networks with this architecture. Clearly $M(\Phi) \leq W(\Phi)$.

Now defining the setting which is considered to be fulfilled for the remaining part of this chapter.

Setting 6.2.5. For $d \in \mathbb{N}$, let $\mu \in C(\mathbb{R}^d, \mathbb{R}^d)$ and $\sigma \in C(\mathbb{R}^d, \mathbb{R}^{d\times d})$ be affine functions satisfying

$$\|\sigma(x)\|_2 + \|\mu(x)\|_2 \leq K(1 + \|x\|_2)$$

for $K > 0$, which exists due to the two Lemmata 6.2.3 and 6.2.3 above. Furthermore for $D > 0$ consider $u_0 \in C(\mathbb{R}^d, [-D, D])$ can be approximated by neural networks in the following sense:
Let $\zeta \geq 1$, $, \gamma, \lambda, \nu \geq 0$ and let $\Phi_{u_0}^\varepsilon$ be a ReLU neural network such that for $\varepsilon \in (0, 1)$ and $x \in \mathbb{R}^d$ it holds that

$$\begin{aligned}
&(i) \quad \left|u_0(x) - R\left(\Phi_{u_0}^\varepsilon\right)(x)\right| \leq \varepsilon(1 + \|x\|_2^\nu), \\
&(ii) \quad \left|R\left(\Phi_{u_0}^\varepsilon\right)\right| \leq D \quad \text{and} \\
&(iii) \quad W\left(\Phi_{u_0}^\varepsilon\right) \leq \zeta d^\gamma \varepsilon^{-\lambda}.
\end{aligned}$$

For $T > 0$ let $u \in C([0, T], \times\mathbb{R}^d, \mathbb{R})$ be the unique function (existence and uniqueness is guaranteed by Proposition 3.4 in [30]) satisfying for all $x \in \mathbb{R}^d$

$$\begin{aligned}
&(iv) \quad u(x,t) \text{ is a viscosity solution of (6.19) for } t \in (0, T), \\
&(v) \quad u(0, x) = u_0(x) \quad \text{and} \\
&(vi) \quad \text{there exists } \theta \geq 0 \text{ such that it holds that} \\
&\qquad \max_{t\in[0,T]} u(t, x) \leq \theta\left(1 + \|x\|_2^\theta\right).
\end{aligned}$$

Additionally, consider a probability space $(\Omega, \mathcal{G}, \mathbb{P})$ and let $(\mathcal{G}_t)_{t\in[0,T]}$ be a filtration which satisfies the usual conditions (as defined in Definition 21.22 in [40], existence is guaranteed by Lemma 9.5 in [39]). Let $(B_t)_{t\in[0,T]} : [0, T] \times \Omega \to \mathbb{R}^d$ be a d-dimensional $(\mathcal{G}_t)$-Brownian Motion (see Defintion 21.8 and Theorem 21.9 in [40]) and for every $\mathcal{G}_0$-measurable random variable $\chi : \Omega \to \mathbb{R}^d$ denote by

$$(S_t^\chi)_{t\in[0,T]} : [0, T] \times \Omega \to \mathbb{R}^d$$

the (up to indistinguishability) unique $(\mathcal{G}_t)$-adapted stochastic process with continuous sample paths satisfying the stochastic differential equation (SDE)

$$dS_t^\chi = \sigma(S_t^\chi)dB_t + \mu(S_t^\chi)dt \quad \text{and} \quad S_0^\chi = \chi$$

$\mathbb{P}$-almost sure for $t \in [0,T]$. Lastly, for $u < v$ let the input data $X : \Omega \to [u,v]^d$ be $\mathcal{G}_0$-measurable and uniformly distributed on $[u,v]^d$.

Remark 6.2.6. Two remarks to Setting 6.2.5.

- Note that in order to fulfill (i) and (ii) in Setting 6.2.5 the realisation of the neural network $\Phi_{u_0}^\varepsilon$ needs to be in $[-D,-D]$ for given $D > 0$. This is ensured via the clip function, which acts as a limitation function and maps for given $D > 0$ function values which are higher than D to D and function values which are smaller than $-D$ to $-D$, so

  $$\text{clip}_D(x) := \min\{|x|, D\} \operatorname{sgn}(x).$$

  The clip function can be shown to be the realization of the neural network

  $$\Phi^{\text{clip}} := \left(\left(\begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right), \left(\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} D \\ D \end{pmatrix} \right), ((-1,1),0) \right),$$

  with $L(\Phi^{\text{clip}}) = 3$ and $M(\Phi^{\text{clip}}) = 8$, compare to Lemma A.2 in [8]. So the neural network $\Phi_{u_0}^\varepsilon$ can be seen as the neural network Φ^{clip} concatenated with the neural network approximating u_0 (whose realization probably yields a value outside of $[-D,D]$).

- At first sight, the assumptions on u_0 in Setting 6.2.5 seem quite restricting. However, one verifies that it holds for u_0 as in (6.20) for $\zeta = 6$, $\gamma = 1$ and $\lambda = \nu = 0$, as $R\left(\Phi_{u_0}^{\text{BS}}\right) = u_0$ (compare to (6.21)) yields (i) and (ii) and (6.22) yields (iii).

Now it is possible to state the main theorem of this section (based on Theorem 3.3 in [8]), which shows that under the assumption of the setting above the solution of (6.19) – so for affine drift μ and diffusion σ, where u_0 can be approximated well – can be approximated by a ReLU neural network without the curse of dimensionality.

Theorem 6.2.7. *Assume that the Setting 6.2.5 is fulfilled for $d \in \mathbb{N}$ and $T > 0$. Let $u < v$, so $[u,v]^d$ is a cube in $\mathbb{R}^d$. Then there exist $C, c \geq 0$ such that the following holds: For every $\varepsilon \in (0,1)$ there exists a ReLU neural network Φ^{Kol} such that*

$$\begin{aligned} L(\Phi^{Kol}) &= L(\Phi_{u_0}^{cd^{-\nu/2}\varepsilon^{1/2}}), \\ M(\Phi^{Kol}) &\leq Cd^{\nu\lambda/2+\gamma}\varepsilon^{-\lambda/2-2} \quad \text{and} \\ \frac{1}{(v-u)^d} \left\| R(\Phi^{Kol}) - u(T,\cdot) \right\|_{L^2([u,v]^d)}^2 &\leq \varepsilon, \end{aligned}$$

where $\Phi_{u_0}^{cd^{-\nu/2}\varepsilon^{1/2}}$ is the neural network approximating u_0 as in Setting 6.2.5.

In order to sketch the proof the Theorem 6.2.7, two auxiliary Lemmata (Lemma A.4 and A.5 in [8]) are given. Subsequently for $p, z > 0$ let $s := \max\{2, p\}$ and define

$$\mathfrak{c}_p(z) := 2^{p/2}(z + KT + sK\sqrt{T})^p \exp(pK^2T[\sqrt{T} + s]^2). \tag{6.23}$$

The first Lemma states that there exists random variables $\mathfrak{M}$ and $\mathfrak{N}$ such that $S_T^x = \mathfrak{M}x + \mathfrak{N}$ $\mathbb{P}$-a.s. for all $x \in \mathbb{R}^d$.

Lemma 6.2.8. *Assume that the Setting 6.2.5 is fulfilled for $d \in \mathbb{N}$ and $T > 0$. Let $u < v$, so $[u, v]^d$ is a cube in $\mathbb{R}^d$. Let $e_i \in \mathbb{R}^d$ for $1 \leq i \leq d$ be the standard basis in $\mathbb{R}^d$. Then the random variables $\mathfrak{M} : \Omega \to \mathbb{R}^{d\times d}$ and $\mathfrak{N} : \Omega \to \mathbb{R}^d$ defined by*

$$\mathfrak{M} := (S_T^{e_1} - S_T^0, S_T^{e_2} - S_T^0, \ldots, S_T^{e_d} - S_T^0) \quad \text{and} \quad \mathfrak{N} := (S_T^0)$$

fulfill for every $x \in \mathbb{R}^d$ $\mathbb{P}$-a.s. that

$$S_T^x = \mathfrak{M}x + \mathfrak{N} =: \mathcal{A}_{\mathfrak{M},\mathfrak{N}}(x) \tag{6.24}$$

and it holds that

$$\mathbb{E}\left(\|\mathfrak{M}\|_2 + \|\mathfrak{N}\|_2\right) \leq 3\mathfrak{c}_1(1)d \quad \text{and} \tag{6.25}$$

$$\left\|\mathbb{E}\left(\|\mathcal{A}_{\mathfrak{M},\mathfrak{N}}\|_2\right)\right\|_{L^2(\mathbb{P}_X)} \leq c_\nu(\max\{1, |u|, |v|\})d^{\nu/2}, \tag{6.26}$$

where $\|f\|_{L^2(\mathbb{P}_X)} := (\int_{[u,v]^d} f^2 d\mathbb{P}_X)^{1/2}$.

The proof of Lemma 6.2.8 is omitted here, refer to Lemmata 2.7 and 2.15 in [30]. The second Lemma shows that realization of a neural network composed with an affine function can be written as realization of another neural network.

Lemma 6.2.9. *Let $d, n \in \mathbb{N}$ and let Φ be a neural network with input dimension d. For $1 \leq j \leq n$ let $M_j \in \mathbb{R}^{d\times d}$ and $N_j \in \mathbb{R}^d$. Then there exists a neural network $\tilde{\Phi}$ such that*

$$\begin{aligned} L(\tilde{\Phi}) &= L(\Phi), \\ M(\tilde{\Phi}) &\leq n^2 W(\Phi) \quad \text{and} \\ R(\tilde{\Phi})(x) &= \frac{1}{n}\sum_{j=1}^{n} R(\Phi)(M_j x + N_j). \end{aligned}$$

Proof. Let Φ have length $L(\Phi) = L \in N$ and let the architecture of Φ be given as $A(\Phi) = (N_0, N_1, \ldots, N_L)$ with $N_0 = d$. Denote by $\mathbb{I}_{N_L}$ the N_L-dimensional identity matrix and define for $1 \leq j \leq n$ the one-layered neural networks

$$\Phi_j := \big((M_j, N_j)\big). \tag{6.27}$$

Then the neural network

$$\tilde{\Phi} := \Big(\frac{1}{n}\big(\underbrace{\mathbb{I}_{N_L}, \ldots, \mathbb{I}_{N_L}}_{n\text{-times}}\big), 0\Big) \bullet FP(\underbrace{\Phi, \ldots, \Phi}_{n\text{-times}}) \bullet P(\Phi_1, \Phi_2, \ldots, \Phi_n) \tag{6.28}$$

fulfills by construction $R(\tilde{\Phi})(x) = \frac{1}{n}\sum_{j=1}^{n} R(\Phi)(M_j x + N_j)$. By Definition 2.1.4, it holds that $L(\tilde{\Phi}) = (1 + L - 1) + 1 - 1 = L$. From (6.28) it follows that

$$A(\tilde{\Phi}) = (N_0, n \cdot N_1, \dots, n \cdot N_{L-1}, N_L).$$

Hence,

$$\begin{aligned} M(\tilde{\Phi}) &\leq nN_1(N_0 + 1) + \left(\sum_{\ell=2}^{L-1} nN_\ell(nN_{\ell-1} + 1)\right) + N_L(nN_{L-1} + 1) \\ &\leq n^2 \left(\sum_{\ell=1}^{L} N_\ell(N_{\ell-1} + 1)\right) = n^2 W(\Phi). \end{aligned} \tag{6.29}$$

□

Remark 6.2.10. With notation as in Lemma 6.2.9 and $\Phi := \big((A_1, b_1), (A_2, b_2), \dots, (A_L, b_L)\big)$ with $A(\Phi) = (N_0, N_1, \dots, N_L)$, where $N_0 = d$, the neural network

$$\tilde{\Phi} = \big((C_1, d_1), (C_2, d_2), \dots, (C_L, d_L)\big)$$

in (6.28) is given as

$$\begin{aligned} C_1 &= \begin{pmatrix} A_1 M_1 \\ \vdots \\ A_1 M_n \end{pmatrix}, \quad d_1 = \begin{pmatrix} A_1 N_1 + b_1 \\ \vdots \\ A_1 N_n + b_1 \end{pmatrix}, \\ C_\ell &= \begin{pmatrix} A_\ell & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_\ell \end{pmatrix}, \quad d_\ell = \begin{pmatrix} b_\ell \\ \vdots \\ b_\ell \end{pmatrix} \quad \text{for } 2 \leq \ell \leq L-1 \quad \text{and} \\ C_L &= \left(\frac{1}{n} A_L, \dots, \frac{1}{n} A_L\right), \quad d_L = b_L. \end{aligned}$$

Hence, it holds that

$$M(\tilde{\Phi}) = nN_1(N_0 + 1) + \sum_{\ell=2}^{L-1} \Big(nN_\ell(\cancel{n}N_{\ell-1} + 1\Big) + N_L(nN_{L-1} + 1) \leq nW(\Phi).$$

Note the not appearing factor n (marked in red and strike-through) in the sum above compared to (6.29) due to the many zeroes in the matrices C_ℓ for $2 \leq \ell \leq L-1$. Therefore $M(\tilde{\Phi})$ is typically way smaller than $n^2 W(\Phi)$.

Proof of Theorem 6.2.7 (sketch). Let $\varepsilon \in (0, 1)$ and define $\mathfrak{m} := \max\{1, |u|, |v|\}$. Consider $\mathfrak{c}_p(z)$ as in (6.23) and $\mathfrak{M}, \mathfrak{N}$ as in Lemma 6.2.8. Let $c := (8\mathfrak{c}_\nu(\mathfrak{m})^{-1})$ and

$$n \in [16D^2\varepsilon^{-1}, 32D^2\varepsilon^{-1}) \cap \mathbb{N}, \quad \delta := cd^{-\nu/2}\varepsilon^{1/2}. \tag{6.30}$$

Moreover, let $((\mathfrak{M}_j, \mathfrak{N}_j))_{j=1}^n$ be i.i.d. random variables with $((\mathfrak{M}_1, \mathfrak{N}_1)) \sim ((\mathfrak{M}, \mathfrak{N}))$. From Setting 6.2.5 and with Lemma 6.2.8 and the Feynman-Kac formula (which establishes a link between parabolic PDEs and stochastic processes, for the PDE (6.19) considered here compare to Corollary 2.23 in [30]) it follows for $x \in \mathbb{R}^d$ that

$$\begin{aligned} \left|u_0(x) - R\left(\Phi_{u_0}^\delta\right)(x)\right| &\leq \delta(1 + \|x\|_2^\nu), \\ \left|R\left(\Phi_{u_0}^\delta\right)\right| &\leq D \quad \text{and (most important)} \\ u(T,x) &= \mathbb{E}\left(u_0 \circ \mathcal{A}_{\mathfrak{M},\mathfrak{N}}(x)\right). \end{aligned}$$

One can show that the random variable $G : \Omega \to [0, \infty)$ given by

$$G := \varepsilon^{-1/2} \underbrace{\left\| u(T,x) - \frac{1}{n}\sum_{j=1}^n R(\Phi_{u_0}^\delta)(\mathfrak{M}_j x + \mathfrak{N}_j) \right\|_{L^2(\mathbb{P}_X)}}_{:=G_1} + (6\mathfrak{c}_1(1)dn)^{-1} \underbrace{\max_{1\leq j\leq n} \left(\|\mathfrak{M}_j\|_2 + \|\mathfrak{N}_j\|_2\right)}_{:=G_2}$$

fulfills $\mathbb{E}(G) \leq 1$ since $\mathbb{E}(G_1) \leq \frac{1}{2}\varepsilon^{1/2}$ (which is not shown here in detail) and $\mathbb{E}(G_2) \leq 3\mathfrak{c}_1(1)dn$ (see (6.25)).

Hence, there exists $\omega \in \Omega$ such that $G(\omega) \leq 1$ and $G_1(\omega) \leq \frac{1}{2}\varepsilon^{1/2}$. With $M_j := \mathfrak{M}_j$ and $N_j := \mathfrak{N}_j$ for $1 \leq j \leq n$ it holds that

$$\frac{1}{(v-u)^d} \left\| \frac{1}{n}\sum_{j=1}^n R(\Phi_{u_0}^\delta)(M_j \cdot + N_j) - u(T,\cdot) \right\|_{L^2([u,v]^d)}^2 \leq G_1^2(\omega) \leq \varepsilon G^2(\omega) \leq \varepsilon,$$

since $G_1(\omega) \leq \varepsilon^{1/2} G(\omega)$. Now defining the desired network Φ^{Kol} using Lemma 6.2.9. As in (6.27) let $\Phi_j := \big((M_j, N_j)\big)$ and set

$$\Phi^{\mathrm{Kol}} := \Big(\frac{1}{n}\big(\underbrace{\mathbb{I}_{N_L}, \ldots, \mathbb{I}_{N_L}}_{n\text{-times}}\big), 0\Big) \bullet FP\left(\underbrace{\Phi_{u_0}^\delta, \ldots, \Phi_{u_0}^\delta}_{n\text{-times}}\right) \bullet P(\Phi_1, \Phi_2, \ldots, \Phi_n).$$

Then, by definition it holds that

$$R(\Phi^{\mathrm{Kol}})(x) = \left(\frac{1}{n}\sum_{j=1}^n R(\Phi_{u_0}^\delta)(M_j x + N_j)\right)$$

with $\left|R(\Phi^{\mathrm{Kol}})\right| \leq D$ by Setting 6.2.5 and Remark 6.2.6.

Moreover, Lemma 6.2.9, equation (6.30) and Setting 6.2.5 yield

$$L\left(\Phi^{\mathrm{Kol}}\right) = L\left(\Phi_{u_0}^{\delta}\right) \quad \text{and}$$
$$M\left(\Phi^{\mathrm{Kol}}\right) \leq n^2 W\left(\Phi_{u_0}^{\delta}\right) \leq (32D^2\varepsilon^{-1})^2 \zeta d^{\gamma}\delta^{-\lambda} \leq C d^{\nu\lambda/2+\gamma}\varepsilon^{-\lambda/2-2}.$$

□

Remark 6.2.11. Remarks to (the proof of) Theorem 6.2.7.

- Once more it was shown that the curse of dimensionality has been overcome, as in Theorem 6.2.7 no exponential dependence on the approximation error ε for the number of needed layers and weights accors when approximation with ReLU neural networks for the linear Kolmogorov PDE (6.19) with affine drift and diffusion, under the assumption that the initial condition can be approximated well (as defined in Setting 6.2.5.

- The G_2-part of the random variable G in the proof of Theorem 6.2.7 is not needed for further arguments in this thesis. However, in [8] the absolute values of the weights in the constructed network in Theorem 6.2.7 are shown to be bounded. Hence also in the analogoues of Setting 6.2.5 and Lemma 6.2.9 statements about the boundedness of the weights are given.

- For the special case of the Black-Scholes equation (compare to c) in Remark 6.2.2) with affine drift and diffusion and with u_0 as in (6.20), so

  $$u_0(x) = \min\{\max\{D - y^T x, 0\}, D\},$$

  which can be rebuilt by the ReLU neural network $\Phi_{u_0}^{\mathrm{BS}}$ given in (6.21), Theorem 6.2.7 yields that for $u < v$ there exists $C > 0$ such that for every $\varepsilon \in (0,1)$ there exists a ReLU neural network Φ^{BS} with

  $$L\left(\Phi^{\mathrm{BS}}\right) = L\left(\Phi_{u_0}^{\mathrm{BS}}\right) = 3,$$
  $$M\left(\Phi^{\mathrm{BS}}\right) \leq C d \varepsilon^{-2} \quad \text{and}$$
  $$\frac{1}{(v-u)^d} \left\| R(\Phi^{\mathrm{BS}}) - u(T,\cdot) \right\|^2_{L^2([u,v]^d)} \leq \varepsilon.$$

Bibliography

[1] Anthony, M. & Bartlett, P. L. (1999). *Neural network learning: theoretical foundations.* Cambridge University Press. http://doi.org/10.1017/CBO9780511624216

[2] Barron, A.R. (1992). Neural net approximation. *Proc. 7th Yale Workshop on Adaptive and Learning Systems, 1,* 69-72.

[3] Barron, A.R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory, 39(3),* 930-945. http://doi.org/10.1109/18.256500

[4] Beck, C., Hornung, F., Hutzenthaler, M., Jentzen, A. and Kruse, T. (2020). Overcoming the curse of dimensionality in the numerical approximation of Allen-Cahn partial differential equations via truncated full-history recursive multilevel Picard approximations. *Journal of Numerical Mathematics 28(4),* 197-222. https://doi.org/10.1515/jnma-2019-0074

[5] Bellman, R. (1957). *Dynamic Programming.* Princeton University Press.

[6] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Jozefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H.P.d.O., Raiman, J., Salimans, T., Schlatter, J., ..., Zhang, S. (2019). *Dota 2 with Large Scale Deep Reinforcement Learning.* http://doi.org/10.48550/arXiv.1912.06680

[7] Berner, J., Elbrächter, D. & Grohs, P. (2019). How degenerate is the parametrization of neural networks with the ReLU activation function? *Advances in Neural Information Processing Systems, 32,* 7790–7801. http://doi.org/10.48550/arXiv.1905.09803

[8] Berner, J., Grohs, P. & Jentzen, A. (2020). Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black-Scholes Partial Differential Equations. *SIAM Journal on Mathematics of Data Science, 2(3),* 631-657. http://doi.org/10.1137/19M125649X

[9] Berner, J., Grohs, P., Kutyniok, G. & Petersen, P. (2022). The Modern Mathematics of Deep Learning. In P. Grohs & G. Kutyniok (Eds.), *Mathematical Aspects of Deep Learning* (pp. 1-111). Cambridge University Press. http://doi.org/10.1017/9781009025096.002

[10] Blechschmidt, J. & Ernst, O.G. (2021). Three ways to solve partial differential equations with neural networks - A review. *GAMM-Mitteilungen, 44(2),* 1-29. http://doi.org/10.1002/gamm.202100006

[11] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., ..., Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems, 34,* 1877–1901. http://doi.org/10.48550/arXiv.2005.14165

[12] Caragea, A., Petersen, P.C. & Voigtlaender, F. (2022). *Neural network approximation and estimation of classifiers with classification boundary in a Barron class.* http://doi.org/10.48550/arXiv.2011.09363

[13] Chen, M., Jiang, H., Liao, W. & Zhao, T. (2019). Efficient approximation of deep ReLU networks for functions on low dimensional manifolds. *Advances in Neural Information Processing Systems, 32*, 8174-8184.

[14] Chui, C.K. & Mhaskar, H. N. (2018). Deep nets for local manifold learning. *Frontiers in Applied Mathematics and Statistics, 4(12),* 1-12. http://doi.org/10.48550/arXiv.1607.07110

[15] Cloninger, A. & Klock, T. (2020). *ReLU nets adapt to intrinsic dimensionality beyond the target domain.* http://doi.org/10.48550/arXiv.2008.02545

[16] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematical Control Signal Systems, 2,* 303–314. http://doi.org/10.1007/BF02551274

[17] Dalhoum, A.L.A. & Al-Rawi, M. (2019) High-Order Neural Networks are Equivalent to Ordinary Neural Networks. *The Canadian Center of Science and Education (CCSE) Modern Applied Science, 13(2),* 228-239. http://doi.org/10.5539/mas.v13n2p228

[18] Daubechies, I., DeVore, R., Foucart, S., Hanin, B. & Petrova, G. (2019). *Nonlinear Approximation and (Deep) ReLU Networks.* http://doi.org/10.48550/arXiv.1905.02199

[19] DeVore, R., Howard, R. & Micchelli, C. (1989). Optimal nonlinear approximation. *Manuscripta mathematica, 63(4),* 469–478. http://doi.org/10.1007/BF01171759

[20] DeVore, R.A. (1998). Nonlinear approximation. *Acta numerica, 7,* 51–150. http://doi.org/10.1017/S0962492900002816

[21] E, W., Ma, C. & Wu, L. (2019). *The Barron Space and the Flow-induced Function Spaces for Neural Network Models.* http://doi.org/10.48550/arXiv.1906.08039

[22] E, W. & Wojtowytsch, S. (2020). *Representation formulas and pointwise properties for Barron functions.* http://doi.org/10.48550/arXiv.2006.05982

[23] E, W. & Wojtowytsch, S. (2020). *On the Banach spaces associated with multi-layer ReLU networks: Function representation, approximation theory and gradient descent dynamics.* http://doi.org/10.48550/arXiv.2007.15623

[24] E, W., Ma, C., Wojtowytsch, S. & Wu, L. (2020). *Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't.* http://doi.org/10.48550/arXiv.2009.10713

[25] Elbraechter, D., Grohs, P., Jentzen, A. & Schwab, C. (2018). DNN Expression Rate Analysis of High-Dimensional PDEs: Application to Option Pricing. *Constructive Approximation, 55,* 3–71. http://doi.org/10.1007/s00365-021-09541-6

[26] Gachagan, A., Ijomah, W., Marshall, S. & Nwankpa, C. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning.* http://doi.org/10.48550/arXiv.1811.03378

[27] Girosi, F. & Poggio, T. (1989). Representation Properties of Networks: Kolmogorov's Theorem Is Irrelevant. *Neural Computation, 1(4),* 465–469. https://doi.org/10.1162/neco.1989.1.4.465

[28] Golse, F. (2013). *Mean field kinetic equations.* Lecture notes, École polytechnique. http://www.cmls.polytechnique.fr/perso/golse/M2/PolyKinetic.pdf

[29] Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning.* MIT Press. www.deeplearningbook.org

[30] Grohs, P., Hornung, F., Jentzen, A. & Von Wurstemberger, P. (2020). *A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations.* http://doi.org/10.48550/arXiv.1809.02362

[31] Hairer, M., Hutzenthaler, M. & Jentzen, A. (2015). Loss of regularity for Kolmogorov equations. *Annals of Probability, 43(2),* 468-527. http://doi.org/10.1214/13-AOP838

[32] Hambuckers, J., Kneib, T. & Wiemann, P.F.V. (2021). *Using the Softplus Function to Construct Alternative Link Functions in Generalized Linear Models and Beyond.* http://doi.org/10.48550/arXiv.2111.14207

[33] He, K., Zhang, X., Ren, S. & Sun, J. (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.* http://doi.org/10.48550/arXiv.1502.01852

[34] He, J., Li, L., Xu, J. & Zheng, C. (2019). ReLU deep neural networks and linear finite elements. *Journal of Computational Mathematics, 38(3),* 502-527. http://doi.org/10.4208/jcm.1901-m2018-0160

[35] Homolak, J. (2023). *Opportunities and risks of ChatGPT in medicine, science, and academic publishing: a modern Promethean dilemma.* http://doi.org/10.3325/cmj.2023.64.1

[36] Imaizumi, M. & Fukumizu, K. (2018). *Deep Neural Networks Learn Non-Smooth Functions Effectively.* http://doi.org/10.48550/arXiv.1802.04474

[37] Imaizumi, M. & Fukumizu, K. (2020). *Advantage of Deep Neural Networks for Estimating Functions with Singularity on Hypersurfaces.* http://doi.org/10.48550/arXiv.2011.02256

[38] Izenman, A.J. (2012). Introduction to manifold learning. *WIREs Computational Statistics, 4,* 439–446. http://doi.org/10.1002/wics.1222

[39] Kallenberg, O. (2017). *Random Measures, Theory and applications.* Springer Link. http://doi.org/10.1007/978-3-319-41598-7

[40] Klenke, A. (2008). *Probability Theory: A comprehensive course* (2nd ed.). Springer Link. http://doi.org/10.1007/978-1-4471-5361-0

[41] Klenke, A. (2013). *Wahrscheinlichkeitstheorie.* (3rd ed.). Springer Link. http://doi.org/10.1007/978-3-662-62089-2

[42] Kolmogoroff, A. (1931). Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung. *Mathematische Annalen, 104,* 415–458. http://doi.org/10.1007/BF01457949

[43] Kolmogorov, A. N. (1956). The representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Doklady Akademii Nauk SSSR, 108(2),* 179–182.

[44] Krizhevsky, A., Sutskever, I. & Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou & K.Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc. http://doi.org/10.1145/3065386

[45] Laakmann, F. & Petersen, P.C. (2020). *Efficient Approximation of Solutions of Parametric Linear Transport Equations by ReLU DNNs.* http://doi.org/10.48550/arXiv.2001.11441

[46] LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature 521,* 436–444. http://doi.org/10.1038/nature14539

[47] Lee, H., Ge, R. Ma, T., Risteski, A. & Arora, S. (2017). *On the ability of neural nets to express distributions.* http://doi.org/10.48550/arXiv.1702.07028

[48] Lee, J.M. (2013). *Introduction to smooth manifolds.* (2nd ed.). Springer Link. http://doi.org/10.5860/choice.40-4655

[49] Leshno, M., Lin, V.Y., Pinkus, A. & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks, 6(6),* 861–867. http://doi.org/10.1016/S0893-6080(05)80131-5

[50] Li, Z., Ma, C. & Wu, L. (2020). *Complexity Measures for Neural Networks with General Activation Functions Using Path-based Norms.* http://doi.org/10.48550/arXiv.2009.06132

[51] Mallat, S. (2012). Group invariant scattering. *Communications on Pure and Applied Mathematics, 65(10),* 1331–1398. http://doi.org/10.48550/arXiv.1101.2286

[52] McCulloch, W.S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*, 115–133. http://doi.org/10.1007/BF02478259

[53] Mhaskar, H. (1993). Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics, 1,* 61–80. http://doi.org/10.1007/BF02070821

[54] Mhaskar, H. (1996). Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation, 8(1),* 164–177. http://doi.org/10.1162/neco.1996.8.1.164

[55] Mhaskar, H., Liao, Q., & Poggio, T. (2016). *Learning functions: When is deep better than shallow?.* http://doi.org/10.48550/arXiv.1603.00988

[56] Mhaskar, H.N. & Poggio, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications, 14(6),* 829–848. http://doi.org/10.48550/arXiv.1608.03287

[57] Mhaskar, H.N. & O'Dowd, R. (2023). *Local transfer learning from one data space to another.* http://doi.org/10.48550/arXiv.2302.00160

[58] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning.* http://doi.org/10.48550/arXiv.1312.5602

[59] Montanelli, H. & Yang, H. (2019). *Error bounds for deep ReLU networks using the Kolmogorov-Arnold superposition theorem.* https://doi.org/10.48550/arXiv.1906.11945

[60] Mouhot, C. (2013). *Hyperbolicity: scalar transport equations, wave equations.* Lecture Notes, University of Cambridge. http://cmouhot.files.wordpress.com/1900/10/chapter41.pdf

[61] Novak, E. & Wozniakowski, H. (2009). Approximation of infinitely differentiable multivariate functions is intractable. *Journal of Complexity, 25(4),* 398–404. http://doi.org/10.1016/j.jco.2008.11.002

[62] Opschoor, A.A., Petersen, P.C. & Schwab, C. (2019). Deep ReLU Networks and High-Order Finite Element Methods. *Analysis and Applications, 18(5),* 715–770. http://doi.org/10.1142/S0219530519410136

[63] Oswald, P. (1990). On the degree of nonlinear spline approximation in Besov-Sobolev spaces. *Journal of Approximation Theory, 61(2),* 131–157. http://doi.org/10.1016/0021-9045(90)90001-7

[64] Petersen, P.C. & Voigtlaender, F. (2018). Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks, 108,* 296–330. http://doi.org/10.48550/arXiv.1709.05289

[65] Petersen, P.C. & Voigtlaender, F. (2021). *Optimal learning of high-dimensional classification problems using deep neural networks.* http://doi.org/10.48550/arXiv.2112.12555

[66] Petersen, P.C. (2022). *Neural Network Theory.* Lecture notes, University of Vienna. http://www.pc-petersen.eu/Neural_Network_Theory.pdf

[67] Pishro-Nik, H. (2014). *Introduction to Probability, Statistics and Random Processes.* Kappa Research, LLC. http://www.probabilitycourse.com

[68] Poggio, T., Anselmi, F. & Rosasco, L. (2015). I-theory on depth vs width: hierarchical function composition. *Center for Brains, Minds and Machines (CBMM), 41,* 1-29. http://cbmm.mit.edu/sites/default/files/publications/cbmm_memo_041.pdf

[69] Poggio, T., Rosasco, L., Shashua, A., Cohen, N. & Anselmi, F. (2015). Notes on Hierarchical Splines, DCLNs and i-theory. *Center for Brains, Minds and Machines (CBMM), 37,* 1-23. http://cbmm.mit.edu/sites/default/files/publications/cbmm-memo_37.pdf

[70] Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B. & Liao, Q. (2017). *Why and When Can Deep - but Not Shallow - Networks Avoid the Curse of Dimensionality: a Review.* http://doi.org/10.48550/arXiv.1611.00740

[71] Raslan, M.K. (2021). *Solving Parametric PDEs with Neural Networks: Unfavorable Structure vs. Expressive Power.* Doctoral thesis, Technical University of Berlin. http://depositonce.tu-berlin.de/handle/11303/12513

[72] Rosenblatt, F. (1958) The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review, 65,* 386-408. http://doi.org/10.1037/h0042519

[73] Roweis, S.T. & Saul, L.K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290,* 2323–2326. http://doi.org/10.1126/science.290.5500.2323

[74] Rudin, W. (1976). *Principles of Mathematical Analysis* (3rd ed.). McGraw-Hill, Inc.

[75] Salih, A. (2016). *Method of Characteristics.* Lecture Notes, Indian Institute of Space Science and Technology. http://www.iist.ac.in/sites/default/files/people/IN08026/MoC_0.pdf

[76] Schmidt-Hieber, J. (2019). *Deep ReLU network approximation of functions on a manifold.* http://doi.org/10.48550/arXiv.1908.00695

[77] Schmidt-Hieber, J. (2021). The Kolmogorov-Arnold representation theorem revisited. *Neural Networks, 137,* 119-126. https://doi.org/10.1016/j.neunet.2021.01.020

[78] Schoenberg, I.J. (1969). Cardinal interpolation and spline functions. *Journal of Approximation theory, 2(2),* 167–206. http://doi.org/10.1016/0021-9045(69)90040-9

[79] Schwab, C. & Zech, J. (2019). Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Analysis and Applications, 17(1),* 19-55. http://doi.org/10.1142/S0219530518500203

[80] Shaham, U., Cloninger, A. & Coifman, R.R. (2015). Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis, 44(3),* 537–557. http://doi.org/10.1016/j.acha.2016.04.003

[81] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T. & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature 550,* 354–359. http://doi.org/10.1038/nature24270

[82] Smith, D. J. & Vamanamurthy, M.K. (1989). How Small Is a Unit Ball?. *Mathematics Magazine, 62(2),* 101-107 http://doi.org/10.1080/0025570X.1989.11977419

[83] Stein, E.M. & Shakarchi, R. (2003). *Fourier Analysis: An Introduction.* Princeton University Press. http://doi.org/10.1515/9781400883899-002

[84] Suzuki, T. (2018). *Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality.* http://doi.org/10.48550/arXiv.1810.08033

[85] Telgarsky, M. (2015). *Representation Benefits of Deep Feedforward Networks.* http://doi.org/10.48550/arXiv.1509.08101

[86] Tenenbaum, J.B., De Silva, V. & Langford, J.C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science, 290,* 2319–2323. http://doi.org/10.1126/science.290.5500.2319

[87] Tu, L.W. (2010). *An Introduction to Manifolds* (2nd ed.). Springer Link. doi.org/10.1007/978-1-4419-7400-6

[88] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems, 30*, 5998-6008. http://doi.org/10.48550/arXiv.1706.03762

[89] Vitushkin, A.G. & Henkin, G.M. (1967). Linear superpositions of functions. *Russian Mathematical Surveys, 22(1),* 77-125. https://doi.org/10.1070/RM1967v022n01ABEH001204

[90] Werner, D. (2018). *Funktionalanalysis*. Springer Link. http://doi.org/10.1007/978-3-662-55407-4

[91] Whitney, H. (1934). Analytic extensions of differentiable functions defined in closed sets. *Transactions of the American Mathematical Society, 36(1),* 63-89. http://doi.org/10.1090/S0002-9947-1934-1501735-3

[92] Wiatowski, T. & Boelcskei, H. (2018). A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction. *IEEE Transactions on Information Theory, 64(3)*, 1845–1866. http://doi.org/10.1109/TIT.2017.2776228

[93] Yarotsky, D. (2017). Error bounds for approximations with deep ReLU networks. *Neural Networks, 94,* 103–114. http://doi.org/10.48550/arXiv.1610.01145

[94] Young, T., Hazarika, D., Poria, S. & Cambria, E. (2018). *IEEE Computational Intelligence Magazine 13(3),* 55-75. http://doi.org/10.1109/MCI.2018.2840738

Last access to all links above on July 16, 2023.

List of Figures