ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Error analysis for deep neural network approximations of parametric hyperbolic conservation laws

T. De Ryck and S. Mishra

Research Report No. 2022-34
July 2022

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

ERROR ANALYSIS FOR DEEP NEURAL NETWORK APPROXIMATIONS OF PARAMETRIC HYPERBOLIC CONSERVATION LAWS

T. DE RYCK AND S. MISHRA

Abstract. We derive rigorous bounds on the error resulting from the approximation of the solution of parametric hyperbolic scalar conservation laws with ReLU neural networks. We show that the approximation error can be made as small as desired with ReLU neural networks that overcome the curse of dimensionality. In addition, we provide an explicit upper bound on the generalization error in terms of the training error, number of training samples and the neural network size. The theoretical results are illustrated by numerical experiments.

1. Introduction

A large number of interesting phenomena in science and engineering are modelled by a class of nonlinear PDEs called *hyperbolic systems of conservation laws* [18]. Examples include the compressible Euler equations of aerodynamics, the shallow-water equations of oceanography, the magnetohydrodynamics (MHD) equations of plasma physics and the equations of nonlinear elasticity. It is well-known that solutions of hyperbolic conservation laws develop discontinuities such as shock waves and contact discontinuities in finite time, even when the initial data are smooth. Thus, solutions are sought in the weak (distributional) sense. However, these weak solutions are not necessarily unique and additional admissible criteria or *entropy conditions* are imposed in order to recover uniqueness.

A variety of numerical methods have been proposed to approximate the entropy solutions of conservation laws efficiently. These include high-resolution finite volume methods, based on TVD, ENO and WENO reconstructions, discontinuous Galerkin and spectral viscosity methods [17]. Despite their well-documented successes, these standard numerical methods are characterized by their high computational cost, particularly in two and three space dimensions [28] and references therein. This cost is particularly evident when the problem at hand requires *multiple* calls to the PDE solver. Such *many-query* problems can appear in a variety of contexts such as Uncertainty quantification (UQ) [4], PDE constrained optimization [5], deterministic and Bayesian Inverse problems [35]. Using conventional numerical methods for these problems is unfeasible and one needs to develop *fast surrogates* which can accurately approximate the PDE solution at a very small fraction of the computational cost of a conventional numerical method for hyperbolic PDEs, particularly in several space dimensions.

Many different approaches have been tried in the context of designing fast and efficient surrogates for conservation laws. One class of methods can be classified as *reduced order models* or *reduced basis methods*, see [31] and references therein for further details of this approach. In these methods, the surrogate is formed from a linear combination of *principal modes*, which can be determined from different orthogonal decompositions of the data based *snapshot matrix*. However, these methods are known to be inefficient in transport-dominated problems such as hyperbolic conservation laws due to the *slow decay* of the underlying eigenvalues, characterizing the principal modes, [1, 7] and references therein.

Another class of surrogate models for PDEs is based on *deep neural networks* (DNNs). The literature on the use of DNNs as surrogates for PDEs has grown exponentially in the last couple of years and a very selected list includes [34, 22] for linear elliptic PDEs, [14] for linear and semi-linear parabolic PDEs, [32, 33] for physics informed neural networks for PDEs and [26, 25] for learning the underlying solution operator for PDEs.

DNNs have recently been explored in the context of designing surrogates for hyperbolic PDEs. References include [28] where the authors use DNNs to approximate observables, corresponding to systems of conservation laws, see also [27], and utilize this surrogate for efficient UQ. Moreover, in [29], DNNs have been investigated as backbones in optimization algorithms, constrained by hyperbolic systems of conservation laws.

(T. De Ryck and S. Mishra) Seminar for Applied Mathematics, ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland

E-mail addresses: tim.deryck@sam.math.ethz.ch, siddhartha.mishra@sam.math.ethz.ch.

The focus of most of the literature for DNN surrogates for hyperbolic conservation laws has been the design and empirical demonstration of these algorithms via numerical experiments. Consequently, there is a paucity of theoretical results in the form of error estimates for deep neural networks as surrogates for hyperbolic PDEs. Providing such rigorous estimates constitutes the main aim of this paper.

We seek to provide rigorous bounds on the error incurred by DNNs when approximating possibly discontinuous solutions of hyperbolic conservation laws. To this end, we will focus on the prototypical case of *scalar conservation laws*, where and in contrast to systems of conservation laws, a rigorous theory of well-posedness of entropy solutions as well convergence of numerical approximations is available. Moreover, we model *many-query* problems by considering scalar conservation laws with *parametric* initial conditions as well as *parametric* flux functions. Such problems arise in the context of UQ as well as design or optimization where the parameters represent random variables or design variables, respectively. A classic example of such parameterizations, particularly in the context of UQ [30], is provided by the so-called *truncated Karhunen-Loève expansions*.

With this setting, our main contributions in this paper are,

- We prove that neural networks can approximate solutions to scalar conservation laws with parametric initial condition (Section 3.1) and/or parametric flux function (Section 3.2) without incurring the curse of dimensionality with respect to the parameter space. We provide explicit error bounds in terms of the neural network size and prove that the network weights and biases remain bounded.
- An explicit, computable upper bound on the so-called *generalization error* is also provided. It shows that the curse of dimensionality is overcome in the number of training samples as well.
- We provide numerical experiments to validate our theoretical observations. In all examples, the training and generalization error only grow polynomially, at worst, in the input dimension and the generalization gap converges as the data set size grows, as predicted by the theory.

Thus, we provide rigorous analysis of the different sources of error in the approximation of parametric scalar conservation laws by DNNs and show that these errors can only grow polynomially in the parametric dimension.

The rest of this paper is organized as follows. In Section 2 we collect preliminary material such as definitions and basic neural network approximation results. The main results on the approximation error for parametric hyperbolic conservation laws by ReLU neural networks can be found in Section 3. A bound on the generalization error is prove in Section 4 and the numerical experiments are presented in Section 5.

2. Preliminaries on Neural Networks

Our fundamental aim in this paper is to study approximation of solutions of conservation laws by neural networks. To this end, we start by collecting some preliminary material and results on DNNs in this section.

2.1. Definitions.

In this paper, we will consider function approximation using feedforward artificial neural networks where only connections between neighbouring layers are allowed. For simplicity, we will just refer to them as neural networks. In the following, we formally introduce our definition of a neural network and the related terminology.

Definition 2.1. *We denote by $\sigma : \mathbb{R} \to \mathbb{R}$ the function that satisfies for all $x \in \mathbb{R}$ that $\sigma(x) = \max\{x, 0\}$ and we call σ the ReLU activation function or rectifier function. For $n \in \mathbb{N}$ and $x \in \mathbb{R}^n$, we define $\sigma(x) := (\max\{x_1, 0\}, \dots, \max\{x_n, 0\})$.*

Definition 2.2. *Let $L \in \mathbb{N}$ and $l_0, \dots, l_L \in \mathbb{N}$. Let σ denote the ReLU activation function and define the parameter space Θ as*

$$\Theta = \bigcup_{L\in\mathbb{N}} \bigcup_{l_0,\dots,l_L\in\mathbb{N}} \bigtimes_{k=1}^{L} \left(\mathbb{R}^{l_k \times l_{k-1}} \times \mathbb{R}^{l_k}\right). \tag{1}$$

For $\theta \in \Theta$, we define $(W_k, b_k) := \theta_k$ and $\mathcal{A}_k : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{l_k} : x \mapsto W_k x + b_k$ for $1 \le k \le L$ and we denote by $\Psi_\theta : \mathbb{R}^{l_0} \to \mathbb{R}^{l_L}$ the function that satisfies for all $x \in \mathbb{R}^{l_0}$ that

$$\Psi_\theta(x) = \begin{cases} \mathcal{A}_1(x) & L = 1 \\ (\mathcal{A}_L \circ \sigma \circ \mathcal{A}_{L-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{A}_1)(x) & L > 1. \end{cases} \tag{2}$$

We refer to Ψ_θ as (the realization of) the ReLU neural network associated to the parameter θ with L layers with widths $(l_1, \dots, l_L)$, of which the first $L-1$ layers are called hidden layers. For $1 \le k \le L$,

we say that layer k has width l_k and we refer to W_k and b_k as the weights and biases corresponding to layer k. If $L \geq 3$, we say that Ψ_θ is a deep ReLU neural network.

We will be interested in proving the existence of neural networks of a certain size that approximate the solution of a PDE to a specified accuracy. Following [15], we will quantify the size of neural networks based on:

- the connectivity $\mathcal{M}(\Phi_\theta) := \|\theta\|_0$, i.e. the total sum of the number of non-zero entries in all weights and biases,
- the depth $\mathcal{L}(\Phi_\theta) := L$,
- the maximum width $\mathcal{W}(\Phi_\theta) := \max_{0\leq k\leq L} l_k$,
- the weight magnitude $\mathcal{B}(\Phi_\theta) := \|\theta\|_\infty$.

Note that this notation is in general not well-defined since the realization map $\theta \mapsto \Psi_\theta$ need not be injective. However, this will not pose a problem since in our proofs we will explicitly construct the parameter vector θ rather than proving the existence of a neural network Ψ_θ.

Next, we introduce the concept of a clipped neural network, following e.g. [3, 20].

Definition 2.3. *Let $a, b, \alpha, \beta \in \mathbb{R}$ with $a < b$ and $\alpha < \beta$, $L \in \mathbb{N}$ and $l_0, \dots, l_L \in \mathbb{N}$, $\theta \in \Theta$ and let Φ_θ be the associated ReLU neural network realization (cf. Definition 2.2). We call Φ_θ a (α, β)-clipped ReLU neural network if it holds that $\Phi_\theta(x) \in [\alpha, \beta]^{l_L}$ for all $x \in [a, b]^{l_0}$.*

Although the use of clipped neural networks will only become clear later on, one should note that any ReLU neural network Φ_θ can be transformed into a (α, β)-clipped ReLU neural network. To see this, consider the following clipping function

$$\mathcal{C} : \mathbb{R} \to \mathbb{R} : x \mapsto \max\{\alpha, \min\{\beta, x\}\}. \tag{3}$$

Clearly, $\mathcal{C}(x) \in [\alpha, \beta]$ for all $x \in \mathbb{R}$ and since $\mathcal{C}$ can also be expressed in terms of the rectifier function, it can be realized as a ReLU neural network. Given that the composition of two ReLU neural networks is again a ReLU neural network, it holds that $\mathcal{C} \circ \Phi_\theta$ is a (α, β)-clipped ReLU neural network for any $\theta \in \Theta$.

Per definition, ReLU neural networks are mappings of which both the input and the output is a vector. In our setting, however, we are interested in expressing or approximating the *parameter-to-solution* map $Y \to L^\infty(D) : y \mapsto u(T, \cdot, y)$ using ReLU neural networks. In order to do this, we must identify the output vector of a ReLU neural network with a function in $L^\infty(D)$. This leads to the following definition.

Definition 2.4. *Let $J, d \in \mathbb{N}$, let $a, b \in \mathbb{R}$ with $a < b$, $\theta \in \Theta$ and let Φ_θ be the ReLU neural network associated to θ with input dimension d and output dimension J. In addition, let $D = [a, b]$ and define $x_{j+1/2} = a + j(b-a)/J$ and $C_j := [x_{j-1/2}, x_{j+1/2})$ for $0 \leq j \leq J$. We define $u_\theta : D \times Y \to \mathbb{R}$ by setting*

$$u_\theta(x, y) = (\Phi_\theta(y))_j \quad \text{for } x \in C_j, y \in Y \text{ and } 1 \leq j \leq J. \tag{4}$$

We call u_θ the solution function associated to the ReLU neural network Φ_θ.

2.2. **Function approximation with ReLU neural networks.** One of the crucial elements in the explanation of the popularity of ReLU neural networks is their expressivity. Under very mild assumptions on the network architecture, ReLU neural networks can approximate many function classes to arbitrarily small accuracy. The classical universal approximation theorem [8, 19, 24] states for example that the class of ReLU neural networks with only one hidden layer is dense in the continuous functions on a compact interval. More recent work tries to provide insight in how the accuracy of the approximation depends on the regularity of the function class and the width and depth of the network.

In the following sections, we will require such accuracy results, but not with respect to the often used L^∞-norm, but to the Lipschitz seminorm. We define the Lipschitz seminorm for a function $f : [a, b] \to \mathbb{R}$ as

$$\|f\|_{\mathrm{Lip}([a,b];\mathbb{R})} = \sup_{x,y\in[a,b],x\neq y} \frac{|f(x) - f(y)|}{x - y}. \tag{5}$$

Next, we prove a result on the approximation of C^2 functions in Lipschitz seminorm by ReLU neural networks. The proof can be found in Appendix A.1.

Lemma 2.5. *Let $a, b \in \mathbb{R}$ with $a < b$ and let $f \in C^2([a, b]; \mathbb{R})$. For any $J \in \mathbb{N}$, there exists a realization of a ReLU neural network Φ^J such that*

$$\left\| f - \Phi^J \right\|_{\mathrm{Lip}([a,b];\mathbb{R})} \leq \frac{b-a}{J} \|f''\|_\infty. \tag{6}$$

In addition, Φ^J satisfies that $\mathcal{M}(\Phi^J) = O(J)$, $\mathcal{L}(\Phi^J) = 2$, $\mathcal{W}(\Phi^J) = J + 3$ and $\mathcal{B}(\Phi^J) \leq \max\{1, |a| + |f(a)|, |b| + |f(b)|, 2\|f'\|_\infty\}$.

Next, we adapt Yarotsky's result on the approximation of the multiplication operator by ReLU neural networks [36] from supremum norm to Lipschitz seminorm. The proof can be found in Appendix A.2.

Lemma 2.6. *Let $M, N > 0$ and denote by $\times : \mathbb{R}^2 \to \mathbb{R} : (x, y) \mapsto xy$ the multiplication operator. For any $m \in \mathbb{N}$, there exists a realization of a ReLU neural network $\widehat{\times}_m : [-M, M] \times [-N, N] \to \mathbb{R}$ such that $\widehat{\times}_m$ satisfies for all $y \in [-N, N]$ the error bound*

$$\left\| \times(\cdot, y) - \widehat{\times}_m(\cdot, y) \right\|_{\mathrm{Lip}([-M,M];\mathbb{R})} \leq \frac{M+N}{2^{m+1}}. \tag{7}$$

Furthermore, it holds that $\mathcal{M}(\widehat{\times}_m) = O(m)$, $\mathcal{L}(\widehat{\times}_m) = m + 1$, $\mathcal{W}(\widehat{\times}_m) = 8$ and $\mathcal{B}(\widehat{\times}_m) = O((M + N)^2)$.

We note that, in terms of complexity, sharper results than Lemma 2.5 are available in the literature, especially for functions of higher regularity. As this exact complexity will not be very important later on, we refrain from discussing these results. In addition, Lemma 2.5 has two advantages. Suppose one is interested in approximating some function to accuracy $\epsilon > 0$ using neural network Φ^ϵ. First, the exact network size and structure of the approximation is stated, whereas in the literature, the exact network size is often dependent on unknown constants that might be very large. Therefore, networks that are asymptotically smaller in the limit $\epsilon \to 0$, when compared to those considered in Lemma 2.5, can be actually larger for realistic values of ϵ. Second, a sharper complexity comes often at the cost of $\mathcal{B}(\Phi^\epsilon) \to \infty$ when $\epsilon \to 0$, e.g. [10]. In practice however, the weights and biases of the network are often regularized to avoid that $\mathcal{B}(\Phi^\epsilon) \to \infty$. Lemma 2.5 and Lemma 2.6 are therefore highly relevant as the magnitudes of the weights and biases of the constructed networks are bounded uniformly in ϵ.

3. ReLU DNN approximation of the solution of parametric scalar hyperbolic conservation laws

In this section we prove our main results on the approximation of the solution of scalar conservation laws with parametric initial data and flux by ReLU neural networks. We show in particular that the constructed network overcomes the curse of dimensionality as its size depends only polynomially on the input dimension of the network. When the initial data and the flux follow a Karhunen-Loève expansion, the size can be seen to only depend *linearly* on the input dimension of the network. We first consider scalar conservation laws with parametric initial data and a fixed flux and generalize to parametric flux functions thereafter.

3.1. Parametric initial data, fixed flux.

We consider the one-dimensional scalar conservation law with parametric initial condition,

$$\begin{cases} \partial_t u(t, x, y) + \partial_x f(u(t, x, y)) = 0, \\ u(0, x, y) = u_0(x, y), \end{cases} \tag{8}$$

for all $x \in D = [a, b]$, $y \in Y = [0, 1]^d$ and $t \in [0, T]$. We make the following assumptions on the flux and the initial data:

- *(A1)* We assume that $f \in C^2(\mathbb{R}; \mathbb{R})$.
- *(A2)* We assume that that $u_0 \in L^\infty(D \times Y)$ and that $\sup_{y \in Y} \mathrm{TV}(u_0(\cdot, y)) < \infty$.
- *(A3)* We assume that the map $u_0(x, \cdot)$ can be accurately approximated by a ReLU neural network for every $x \in D$. More precisely, we assume that there exists a constant $C_{\mathcal{B}}(u_0) > 0$ such that for every $x \in D$ and $\epsilon > 0$ there exists a ReLU neural network $\widehat{u}_0^\epsilon(x, \cdot)$ with $\mathcal{B}(\widehat{u}_0^\epsilon(x, \cdot)) \leq C_{\mathcal{B}}$ that satisfies the error bound
$$\left\| u_0(x, \cdot) - \widehat{u}_0^\epsilon(x, \cdot) \right\|_{L^1(Y;\mathbb{R})} \leq \epsilon.$$
In addition, the connectivity, depth and maximal width of the networks should only grow polynomially in d, i.e. for $\mathcal{Q} \in \{\mathcal{M}, \mathcal{L}, \mathcal{W}\}$ there should exist $C_{\mathcal{Q}}(u_0), \sigma_{\mathcal{Q}}(u_0), \eta_{\mathcal{Q}}(u_0) > 0$ such that it holds for all $x \in D$ that
$$\mathcal{Q}(\widehat{u}_0^\epsilon(x, \cdot)) \leq C_{\mathcal{Q}} d^{\sigma_{\mathcal{Q}}} \epsilon^{-\eta_{\mathcal{Q}}}.$$
Finally, we require that
$$C_0 = \sup_{\epsilon > 0} \max\{\|u_0\|_{L^\infty(D \times Y)}, \|\widehat{u}_0^\epsilon\|_{L^\infty(D \times Y)}\} < \infty,$$
$$C_{TV} = \sup_{y \in Y, \epsilon > 0} \max\{\mathrm{TV}(u_0(\cdot, y)), \mathrm{TV}(\widehat{u}_0^\epsilon(\cdot, y))\} < \infty.$$

The third assumption guarantees that u_0 can be approximated by ReLU neural networks without the curse of dimensionality. Examples of functions for which (A3) is satisfied include functions with a compositional structure or functions for which the relevant input data lies on a lower-dimensional manifold. Another class of functions that is especially relevant for conservation laws is highlighted in the following example.

Example 3.1. *Assumption (A3) is satisfied by (random) initial data that admits a Karhunen-Loève expansion* [30]*, as one can then truncate this expansion up to d terms to retrieve initial data of the form,*

$$u_0(x,y) = \overline{u}(x) + \sum_{i=1}^{d} \sqrt{\lambda_i} y_i \varphi_i(x), \tag{9}$$

where $\overline{u} \in L^\infty(D) \cap BV(D)$, $y_i \in [0,1]$ and where φ_i are some basis functions for all i. Note that the constant d now reflects the dimensionality of the initial data. For every $x \in D$, $u_0(x,\cdot)$ can be trivially realized by a ReLU neural network with 0 hidden layers and $d+1$ non-zero weights and biases. In this case, it holds for all $x \in D$ that $\mathcal{M}(\widehat{u}_0^\epsilon(x,\cdot)) = d+1$, $\mathcal{L}(\widehat{u}_0^\epsilon(x,\cdot)) = 1$ and $\mathcal{W}(\widehat{u}_0^\epsilon(x,\cdot)) = d$.

Under the aforementioned three assumptions, we prove that the solution of (8) at time $t = T$ can be approximated using a ReLU neural network. Moreover, the size of this network and its number of parameters only grow polynomially in the dimension d of the parameter space of the initial data.

Theorem 3.2. *Let $T > 0$, $d \in \mathbb{N}$, $a, b \in \mathbb{R}$ with $a < b$ and $D = [a,b]$. For $y \in Y = [0,1]^d$, denote by $u(T,\cdot,y)$ the solution at time $t = T$ of (8) with a flux $f : [-C_0, C_0] \to \mathbb{R}$ that satisfies assumption (A1) and initial condition $u_0(\cdot,y)$ that satisfies assumptions (A2) and (A3). Then for every $N \in \mathbb{N}$ there exists a ReLU neural network such that the associated solution map $\widehat{\mathcal{U}}^N : Y \to \mathbb{R}$ (cf. Definition 2.4) satisfies the error bound*

$$\sup_{y \in Y} \left\| u(T,\cdot,y) - \widehat{\mathcal{U}}^N(\cdot,y) \right\|_{L^1(D)} \leq \frac{2C_{TV}T\left(C_0 \|f''\|_\infty + 18\left(1 + \|f'\|_\infty\right)^2\right) + 1}{\sqrt{N}}. \tag{10}$$

In addition, it holds that that for $N \in \mathbb{N}$,

$$\begin{aligned}
\mathcal{M}(\widehat{\mathcal{U}}^N) &= O\left(d^{\sigma_{\mathcal{M}}} N^{1+\eta_{\mathcal{M}}/2} + N^{5/2}\right), \\
\mathcal{L}(\widehat{\mathcal{U}}^N) &\leq C_{\mathcal{L}} d^{\sigma_{\mathcal{L}}} N^{\eta_{\mathcal{L}}/2} + N, \\
\mathcal{W}(\widehat{\mathcal{U}}^N), &\leq \frac{2(b-a)}{T\|f'\|_\infty} \max\left\{1 + C_{\mathcal{W}} d^{\sigma_{\mathcal{W}}} N^{1+\eta_{\mathcal{W}}/2}, N^{3/2}\right\}, \\
\mathcal{B}(\widehat{\mathcal{U}}^N) &\leq \max\{C_{\mathcal{B}}, C_0 + \left|f(-C_0)\right|, C_0 + \left|f(C_0)\right|, 2\|f'\|_\infty, (2\|f'\|_\infty)^{-1}\}.
\end{aligned} \tag{11}$$

Proof. The proof will proceed in the following way. We will consider a modification of scalar conservation law (8) where both flux function and initial data of (8) have been replaced by ReLU neural networks. As a first step, we will show that the solution $\widehat{u}$ of the modified scalar conservation law is a good approximation of the solution u of (8). Next, we will prove that it is possible to approximate $\widehat{u}$ with a ReLU neural network. Combining these two results then proves the theorem.

Step 1. Let $u(T,\cdot,y)$ be the solution to (8) for some $y \in Y$ at time $t = T$. By the maximum principle [18, Thm. 2.14], it holds that

$$\left\|u(T,\cdot,y)\right\|_{L^\infty(D)} \leq \left\|u_0(\cdot,y)\right\|_{L^\infty(D)} \tag{12}$$

since $u_0(\cdot,y) \in L^\infty(D) \cap BV(D)$. As $u_0 \in L^\infty(D \times Y)$ (A2) we find that also $u(T) \in L^\infty(D \times Y)$. This allows us to restrict the domain of the flux function $f : \mathbb{R} \to \mathbb{R}$ to the compact interval $[-C_0, C_0]$, where C_0 is defined in (A3). By assumption (A1) and Lemma 2.5, for every $K \in \mathbb{N}$ there exists a ReLU neural network $\widehat{f} := \widehat{f}_K : [-C_0, C_0] \to \mathbb{R}$ that satisfies the error bound

$$\left\|f - \widehat{f}\right\|_{\mathrm{Lip}([-C_0,C_0];\mathbb{R})} \leq \frac{2C_0\left\|f''\right\|_{L^\infty([-C_0,C_0])}}{K}. \tag{13}$$

Furthermore, we can approximate $u_0(x,\cdot)$ for every $x \in D$ by a ReLU neural network $\widehat{u}_0^\epsilon(x,\cdot)$. Based on $\widehat{u}_0^\epsilon$, we define a function $\tilde{u}_0^\epsilon$ such that $\tilde{u}_0^\epsilon(\cdot,y)$ is piecewise constant for all $y \in Y$ and $\tilde{u}_0^\epsilon(x,\cdot)$ can be realized by a ReLU neural network for all $x \in D$. For this reason we take $J \in \mathbb{N}$ and set $\Delta x = (b-a)/J$ and $x_{j-1/2} = a + j\Delta x$, $1 \leq j \leq J+1$. We then set

$$\tilde{u}_0^\epsilon(x,\cdot) = \widehat{u}_0^\epsilon(x_j,\cdot), \qquad \text{for } x \in C_j := [x_{j-1/2}, x_{j+1/2}], \quad 1 \leq j \leq J, \tag{14}$$

$$\overline{u_0}(x,\cdot) = u_0(x_j,\cdot), \qquad \text{for } x \in C_j, \quad 1 \leq j \leq J. \tag{15}$$

We will prove an upper bound for $\|u_0 - \tilde{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))}$. Using the triangle inequality, we obtain

$$\|u_0 - \tilde{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))} \leq \|u_0 - \overline{u_0}\|_{L^\infty(Y;L^1(D))} + \|\overline{u_0} - \tilde{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))}. \tag{16}$$

We first prove that $\|u_0 - \overline{u_0}\|_{L^\infty(Y;L^1(D))}$ is small for large J. Rewriting this norm and performing a change of variables gives us

$$\begin{aligned}
&\|u_0 - \overline{u_0}\|_{L^\infty(Y;L^1(D))} \\
&\quad = \sup_{y\in Y} \sum_j \int_{C_j} |u_0(x,y) - u_0(x_j,y)|dx \\
&\quad \leq \sup_{y\in Y} \int_0^{\frac{b-a}{2J}} \sum_j \left(\left|u_0(x_j+t,y) - u_0(x_j,y)\right| + \left|u_0(x_j,y) - u_0(x_{j-1/2}+t,y)\right| \right) dt \\
&\quad \leq \sup_{y\in Y} \int_0^{\frac{b-a}{2J}} \mathrm{TV}(u_0(\cdot,y))dt = \frac{b-a}{2J} \sup_{y\in Y} \mathrm{TV}(u_0(\cdot,y)).
\end{aligned} \tag{17}$$

Now we still must find a bound for the second term in (16). Note that

$$\left\|\overline{u_0}(\cdot,y) - \tilde{u}_0^\epsilon(\cdot,y)\right\|_{L^1(D)} = \frac{b-a}{J} \sum_j |u_0(x_j,y) - \widehat{u}_0^\epsilon(x_j,y)| \tag{18}$$

is nothing more than a numerical quadrature approximation of $\left\|u_0(\cdot,y) - \widehat{u}_0^\epsilon(\cdot,y)\right\|_{L^1(D)}$. Since $\sup_{y\in Y} \mathrm{TV}(u_0(\cdot,y)) < \infty$ (A2) and $\sup_{y\in Y,\epsilon>0} \mathrm{TV}(\widehat{u}_0^\epsilon(\cdot,y)) < \infty$ (A3) it follows from the Koksma-Hlawka inequality [6] that

$$\left| \|u_0 - \widehat{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))} - \|\overline{u_0} - \tilde{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))} \right| \leq \frac{4(b-a)C_{TV}}{J}, \tag{19}$$

where $C_{TV} = \sup_{y\in Y,\epsilon>0} \max\{\mathrm{TV}(u_0(\cdot,y), \mathrm{TV}(\widehat{u}_0^\epsilon(\cdot,y)\}$. From assumptions (A2) and (A3) it follows that $C_{TV} < \infty$. Since we know by (A3) that for any $\epsilon > 0$ we can choose $\widehat{u}_0^\epsilon$ such that $\|u_0 - \widehat{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))} \leq \epsilon$, we have now successfully bounded the second term in (16). In summary, we have that

$$\|u_0 - \tilde{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))} \leq \frac{9}{2} C_{TV} \frac{b-a}{J} + \epsilon. \tag{20}$$

Now denote by $\widehat{u}^\epsilon$ the solution to the following modification of scalar conservation law (8):

$$\begin{cases} \partial_t \widehat{u}^\epsilon(t,x,y) + \partial_x \widehat{f}(\widehat{u}^\epsilon(t,x,y)) = 0, \\ \widehat{u}^\epsilon(0,x,y) = \tilde{u}_0^\epsilon(x,y), \end{cases} \tag{21}$$

for all $x \in D = [a,b]$, $y \in Y = [0,1]^d$ and $t \in [0,T]$. Note that (21) is well-defined because the maximum principle ensures again that

$$\sup_{x\in D, y\in Y, \epsilon>0} |\widehat{u}^\epsilon(T,x,y)| \leq C_0, \tag{22}$$

where C_0 is as in (A3). By [18, Thm. 4.3] and (13), it then holds that

$$\begin{aligned}
\|u - \widehat{u}^\epsilon\|_{L^\infty([0,T]\times Y;L^1(D))} &\leq \|u_0 - \tilde{u}_0^\epsilon\|_{L^\infty(Y;L^1(D))} + C_{TV} T \left\| f - \widehat{f} \right\|_{\mathrm{Lip}([-C_0,C_0];\mathbb{R})} \\
&\leq \frac{9C_{TV}(b-a)}{2J} + \epsilon + \frac{C_{TV} T C_0}{K} \left\| f'' \right\|_{L^\infty([-C_0,C_0])}.
\end{aligned} \tag{23}$$

Step 2. We now proceed to show that $\widehat{u}^\epsilon$ can be approximated using a ReLU neural network. We do this by emulating the Lax-Friedrichs scheme. For this purpose, we discretize $[0,T] \times D$ in time and space: let $N \in \mathbb{N}$ and define $\Delta t = T/N$ and $t^n = n\Delta t$, $0 \leq n \leq N$. We then set $\Delta x = F\Delta t = (b-a)/J$ for $F = \left\|f'\right\|_{L^\infty([-C_0,C_0])}$ and $J \in \mathbb{N}$ such that the CFL condition is satisfied. Note that this is the same J as in step 1. Furthermore we define $x_{j-1/2} = a + j\Delta x$, $1 \leq j \leq J+1$. For every y we can then define the cell averages

$$\widehat{U}_j^0(y) = \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \tilde{u}_0^\epsilon(x,y)dx = \tilde{u}_0^\epsilon(x_j,y) = \widehat{u}_0^\epsilon(x_j,y), \qquad 1 \leq j \leq J. \tag{24}$$

Thanks to our choice of initial data $\tilde{u}_0^\epsilon$, we can explicitly calculate the cell averages, instead of approximating them. Next, we will approximate $\widehat{u}^\epsilon$ at time $t = T$ using a finite volume scheme. We opt for the

Lax-Friedrichs scheme because of its simple form and suitable theoretical properties. The Lax-Friedrichs approximation of $\frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \widehat{u}^\epsilon(t^{n+1},x,y)dx$ is recursively defined as

$$\widehat{U}_j^{n+1}(y) = \frac{\widehat{U}_{j+1}^n(y)+\widehat{U}_{j-1}^n(y)}{2} - \frac{\Delta t}{2\Delta x}(\widehat{f}(\widehat{U}_{j+1}^n(y)) - \widehat{f}(\widehat{U}_{j-1}^n(y))). \tag{25}$$

Now recall that $\widehat{u}_0^\epsilon(x_j,y)$, and therefore also $\widehat{U}_j^0(y)$, can be written as a ReLU neural network. Furthermore, $\widehat{U}_j^{n+1}(y)$ can be written as a ReLU DNN, as $\widehat{f}$ itself is a ReLU DNN. It is therefore possible to obtain a ReLU DNN that maps y to $(\widehat{U}_1^N(y),\dots,\widehat{U}_J^N(y))$. This network is visualized in Figure 1.

Let $\widehat{\mathcal{U}}^N(\cdot,y)$ be the piecewise constant function associated to to the constructed DNN, i.e. $\widehat{\mathcal{U}}^N(x,y) = \widehat{U}_j^N(y)$ for $x\in(x_{j-1/2},x_{j+1/2})$ (cf. Definition 2.4). The accuracy of this approximation is quantified by an error estimate due to Kuznetsov (Lemma A.1). For every y, it holds that

$$\sup_{y\in Y}\left\|\widehat{u}^\epsilon(T,\cdot,y) - \widehat{\mathcal{U}}^N(\cdot,y)\right\|_{L^1(D)} \le \frac{31C_{TV}T\left(1+F\right)^2}{\sqrt{N}}, \tag{26}$$

where C_{TV} was defined in Step 1 and $F = \left\|f'\right\|_{L^\infty([-C_0,C_0])}$. Note that in practice the observed convergence rate is generally higher than 1/2.

Step 3. We now combine the results from the previous steps. Set $\epsilon = 1/\sqrt{N}$ and $K = \left\lceil\sqrt{N}\right\rceil$ such that $K\le 2\sqrt{N}$. Using (23) and (26) and by recalling that $(b-a)N = FTJ$ we obtain

$$\begin{aligned}
&\sup_{y\in Y}\left\|u(T,\cdot,y) - \widehat{\mathcal{U}}^N(\cdot,y)\right\|_{L^1(D)}\\
&\quad\le \sup_{y\in Y}\left\|u(T,\cdot,y) - \widehat{u}^\epsilon(T,\cdot,y)\right\|_{L^1(D)} + \sup_{y\in Y}\left\|\widehat{u}^\epsilon(T,\cdot,y) - \widehat{\mathcal{U}}^N(\cdot,y)\right\|_{L^1(D)}\\
&\quad\le \frac{9C_{TV}FT}{2N} + \epsilon + \frac{2C_{TV}TC_0}{K}\left\|f''\right\|_{L^\infty([-C_0,C_0])} + \frac{31C_{TV}T\left(1+F\right)^2}{\sqrt{N}}\\
&\quad\le \frac{2C_{TV}T\left(C_0\left\|f''\right\|_{L^\infty([-C_0,C_0])} + 18\left(1+F\right)^2\right)+1}{\sqrt{N}}.
\end{aligned} \tag{27}$$

This proves (10).

Step 4. Lastly, we calculate the depth and the number of non-zero weights and biases of the ReLU neural network $\widehat{\mathcal{U}} := \widehat{\mathcal{U}}^N$, where we identify the constructed ReLU network with vectorial output $(\widehat{U}_1^N(y),\dots,\widehat{U}_J^N(y))$ with the associated piecewise constant function (cf. Definition 2.4). Given that there are N time steps, and that the network $\widehat{f}$ has only one hidden layer, the total network can be seen to have $\mathcal{L}(\widehat{u}_0^\epsilon)+N$ hidden layers. Similarly, the maximum width of the network can be seen to be $J\cdot\max\{2+K,\mathcal{W}(\widehat{u}_0^\epsilon)\}\le 2J\cdot\max\{1+\sqrt{N},\mathcal{W}(\widehat{u}_0^\epsilon)\}$. Finally, the network essentially consists of $J\cdot N$ identical subnetworks, which we denote by $\widehat{\mathcal{L}}$. Each of these subnetworks consists of two copies of $\widehat{f}$ and two copies of a ReLU neural network (of which the depth matches that of $\widehat{f}$) that realizes the identity function. In both cases, two copies correspond to the evaluations in $\widehat{U}_{j-1}^n(y)$ and $\widehat{U}_{j+1}^n(y)$, cf. (25). It is also clear that the calculation of $(\widehat{U}_j^0(y))_j$ gives rise to J networks with each $O(d^{\sigma_{\mathcal{M}}}\epsilon^{-\eta_{\mathcal{M}}})$ non-zero weights and biases. Given that $\widehat{f}$ consists of $O(K)$ non-zero weights and biases, the total network has $O\left(J\mathcal{M}(\widehat{u}_0^\epsilon)+JNK\right)$ non-zero weights and biases. Finally, using that $K\le 2\sqrt{N}$, $(b-a)N = FTJ$ and assumption (A3) concludes the proof.

□

Remark 3.3. *The proof of Theorem 3.2 does not critically depend on the use of the Lax-Friedrichs scheme and the ReLU activation function. In fact, the argument also works with many other activation functions and finite volume schemes for which the convergence rate can explicitly proven. The advantage of the Lax-Friedrichs scheme is that it easily can be written as a ReLU neural network and that the proof of its convergence rate is straightforward.*

The bounds of Theorem 3.2 are particularly simple in the setting of Example 3.1. The following corollary shows that the connectivity and network width only depend linearly on the parameter space dimension.

Corollary 3.4 (Theorem 3.2 for Karhunen-Loève expansion)**.** *Let $T>0$, $d\in\mathbb{N}$, $a,b\in\mathbb{R}$ with $a<b$ and $D=[a,b]$. Let $C_0 = \|u_0\|_{L^\infty(D\times Y)}$ and $C_{TV} = \sup_{y\in Y} TV(u_0(\cdot,y))$. For $y\in Y=[0,1]^d$, denote*

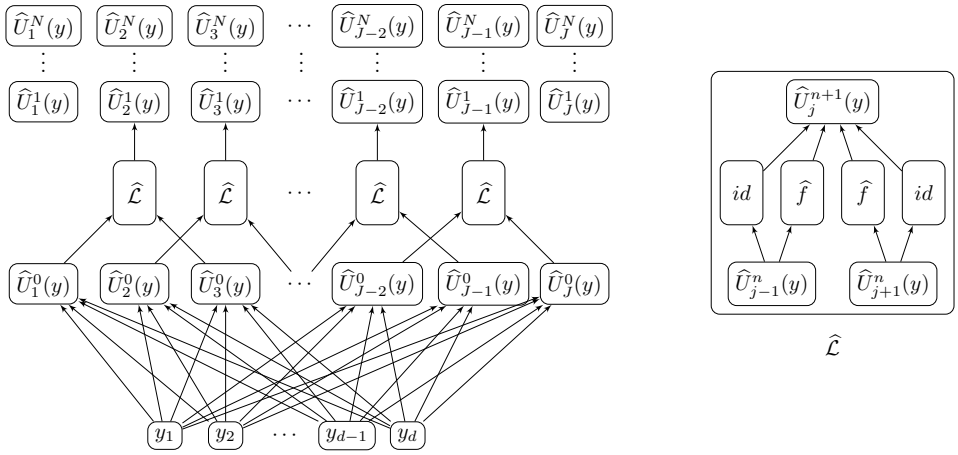


FIGURE 1. Flowchart of the ReLU neural networks $\widehat{\mathcal{U}}^N$ and $\widehat{\mathcal{L}}$ for a fixed flux.

by $u(T,\cdot,y)$ the solution at time $t=T$ of (8) with a flux $f:[-C_0,C_0]\to\mathbb{R}$ that satisfies assumption (A1) and initial condition $u_0(\cdot,y)$ given by a Karhunen-Loève expansion (cf. Example 3.1). Then for every $N\in\mathbb{N}$ there exists a ReLU neural network such that the associated solution map $\widehat{\mathcal{U}}^N: Y\to\mathbb{R}$ (cf. Defintion 2.4) satisfies the error bound

$$\sup_{y\in Y}\left\|u(T,\cdot,y)-\widehat{\mathcal{U}}^N(\cdot,y)\right\|_{L^1(D)}\leq\frac{2C_{TV}T\left(C_0\|f''\|_\infty+18\left(1+\|f'\|_\infty\right)^2\right)}{\sqrt{N}}. \tag{28}$$

In addition, it holds that for $N\in\mathbb{N}$,

$$\begin{aligned}
\mathcal{M}(\widehat{\mathcal{U}}^N)&=O\left(dN+N^{5/2}\right),\\
\mathcal{L}(\widehat{\mathcal{U}}^N)&\leq N+1,\\
\mathcal{W}(\widehat{\mathcal{U}}^N)&\leq\max\left\{1+d,\frac{2(b-a)N^{3/2}}{T\|f'\|_\infty}\right\},\\
\mathcal{B}(\widehat{\mathcal{U}}^N)&\leq\max\{C_{\mathcal{B}},C_0+|f(-C_0)|,C_0+|f(C_0)|,2\|f'\|_\infty,(2\|f'\|_\infty)^{-1}\}.
\end{aligned} \tag{29}$$

3.2. **Parametric flux and initial data.** We expand the results from the previous section to one-dimensional scalar conservation laws with parametric initial condition and a parametric flux,

$$\begin{cases}\partial_t u(t,x,y,z)+\partial_x f(u(t,x,y),z)=0,\\ u(0,x,y,z)=u_0(x,y),\end{cases} \tag{30}$$

for all $x\in D=[a,b]$, $y\in Y=[0,1]^d$, $z\in Z=[0,1]^s$ and $t\in[0,T]$. In this notation, the initial condition is parametrized by the vector y and the flux function by the vector z. To prove our approximation result we need to slightly adapt assumptions (A1-3) we made in the previous section. In the current setting we assume:

(B1) We replace (A1) by the assumption that the flux is in the form of a truncated Karhunen-Loève expansion (cf. Example 3.1),

$$f(u,z)\approx\overline{f}(u)+\sum_{i=1}^{s}\sqrt{\lambda_i}z_i\varphi_i(u),$$

where $\overline{f}\in C^2(\mathbb{R};\mathbb{R})$ and φ_i are some smooth basis functions.

(B2-3) We take over assumptions (A2) and (A3), including the definition of C_0 and C_{TV}.

(B4) We make the assumption that there exist constants $C_f>1$ and $\sigma_f>0$ such that it holds that

$$\|\partial_u f(u,z)\|_{L^\infty([-C_0,C_0])}\leq C_f s^{\sigma_f}$$

and that

$$\max_i\sqrt{\lambda_i}\left(C_0\|\varphi_i''\|_\infty+\frac{\|\varphi_i\|_\infty+1}{4}\|\varphi_i'\|_\infty\right)\leq C_f s^{2\sigma_f},$$

where $\|\cdot\|_\infty := \|\cdot\|_{L^\infty([-C_0,C_0])}$.

Assumption (B4) is related to the decay of the Karhunen-Loève eigenvalues. We will see in the numerical experiments that these conditions are generally easily satisfied. We can then obtain the following result.

Theorem 3.5 (Extension of Theorem 3.2 to uncertain flux)**.** *Let $T > 0$, $d, s \in \mathbb{N}$, $a, b \in \mathbb{R}$ with $a < b$ and $D = [a, b]$. For $y \in Y = [0, 1]^d$ and $z \in Z = [0, 1]^s$, denote by $u(T, \cdot, y, z)$ the solution at time $t = T$ of (8) with flux $f(\cdot, z)$ as in (B1) and initial condition $u_0(\cdot, y)$ that satisfies assumptions (A2) and (A3). Then for every $N \in \mathbb{N}$ there exists a ReLU neural network $\widehat{\mathcal{U}}^N$ that satisfies the error bound*

$$\sup_{y\in Y} \left\| u(T,\cdot,y) - \widehat{\mathcal{U}}^N(\cdot,y) \right\|_{L^1(D)} \leq \frac{2C_{TV}T\left(C_0\left\|\overline{f}''\right\|_\infty + 19\left(1 + C_f s^{\sigma_f}\right)^2\right) + 1}{\sqrt{N}}. \tag{31}$$

Let In addition, it holds that that there exist a constant $\gamma(\sigma_f, \eta_{\mathcal{L}}, \eta_{\mathcal{W}}, \eta_{\mathcal{M}}) \geq 0$ with the property that $\gamma = 0$ if $\sigma_f = 0$ such that for $N \in \mathbb{N}$ it holds that

$$\begin{aligned}
\mathcal{M}(\widehat{\mathcal{U}}^N) &= O\left(d^{\sigma_{\mathcal{M}}} s^\gamma N^{\eta_{\mathcal{M}}/2} + s^{2+\gamma} N^{5/2}\right),\\
\mathcal{L}(\widehat{\mathcal{U}}^N) &= O\left(d^{\sigma_{\mathcal{L}}} s^\gamma N^{\eta_{\mathcal{L}}/2} + s^\gamma N \ln(sN)\right),\\
\mathcal{W}(\widehat{\mathcal{U}}^N) &= O\left(d^{\sigma_{\mathcal{W}}} s^\gamma N^{\eta_{\mathcal{W}}/2} + s^{2+\gamma} N^{3/2}\right),\\
\mathcal{B}(\widehat{\mathcal{U}}^N) &= O(1).
\end{aligned} \tag{32}$$

Proof. The proof is essentially the same as that of Theorem 3.2, only the approximation of f requires more attention, as well as determining the size of the network.

Step 1. We start by proving the equivalent of (13) in the proof Theorem 3.2. Using the triangle inequality, it suffices to treat each term in the flux of (B1) separately. We approximate $\overline{f}$ by the network from Lemma 2.5 using parameter $J \leftarrow K_1 = \lceil\sqrt{N^*}\rceil$, where we denote by N^* the number of time steps. For every i, we approximate φ_i by the network from Lemma 2.5 using parameter $J \leftarrow K_2 = sK_1$. Finally, we approximate the multiplication operators using Lemma 2.6 with parameters $m \leftarrow \log_2(s\sqrt{N^*})$, $M \leftarrow 1$ and $N \leftarrow \|\varphi_i\|_\infty$. This then gives us the existence of a ReLU neural network $\widehat{f}$ that satisfies the error bound

$$\begin{aligned}
&\sup_{z\in Z} \left\| f(\cdot,z) - \widehat{f}(\cdot,z) \right\|_{\mathrm{Lip}([-C_0,C_0];\mathbb{R})}\\
&\quad\leq \left\| \overline{f} - \widehat{\overline{f}} \right\|_{\mathrm{Lip}([-C_0,C_0];\mathbb{R})} + \sum_{i=1}^s \sqrt{\lambda_i} \|\varphi_i - \widehat{\varphi}_i\|_{\mathrm{Lip}([-C_0,C_0];\mathbb{R})}\\
&\qquad + \sum_{i=1}^s \sqrt{\lambda_i} \sup_{z\in Z} \left\| \times(z_i,\cdot) - \widehat{\times}_m(z_i,\cdot) \right\|_{\mathrm{Lip}([-\|\varphi_i\|_\infty, \|\varphi_i\|_\infty];\mathbb{R})} \|\varphi_i\|_{\mathrm{Lip}([-C_0,C_0];\mathbb{R})}\\
&\quad\leq \frac{2C_0\left\|\overline{f}''\right\|_\infty}{K_1} + s\max_i \sqrt{\lambda_i}\left(\frac{2C_0\|\varphi_i''\|_\infty}{K_2} + \frac{(\|\varphi_i\|_\infty + 1)\|\varphi_i'\|_\infty}{2^{m+1}}\right)\\
&\quad\leq \frac{4C_0\left\|\overline{f}''\right\|_\infty + \max_i \sqrt{\lambda_i}\left(4C_0\|\varphi_i''\|_\infty + (\|\varphi_i\|_\infty + 1)\|\varphi_i'\|_\infty\right)}{2\sqrt{N^*}}\\
&\quad\leq \frac{2C_0\left\|\overline{f}''\right\|_\infty + 2C_f s^{2\sigma_f}}{\sqrt{N^*}} \leq \frac{2C_0\left\|\overline{f}''\right\|_\infty + 2(1 + C_f)^2}{\sqrt{N}},
\end{aligned} \tag{33}$$

where we used assumption (B4) and where we defined $N = s^{-4\sigma_f} N^*$ in order to make the error bound independent of s. The existence of the network $\widehat{\mathcal{U}}^{N^*}$ and error bound (31) then follow mutatis mutandis from the steps described in the proof of Theorem 3.2.

Step 2. We now determine the size of $\widehat{\mathcal{U}}^{N^*}$, cf. step 3 of the proof of Theorem 3.2. A visualization can be found in Figure 2. Using the notation from the proof of Theorem 3.2, the calculation of $(\widehat{U}_j^0(y))_j$ gives rise to J^* parallel subnetworks of depth $C_{\mathcal{L}} d^{\sigma_{\mathcal{L}}} (N^*)^{\eta_{\mathcal{L}}/2}$, width $C_{\mathcal{W}} d^{\sigma_{\mathcal{W}}} (N^*)^{\eta_{\mathcal{W}}/2}$ and $C_{\mathcal{M}} d^{\sigma_{\mathcal{M}}} (N^*)^{\eta_{\mathcal{M}}/2}$ non-zero weights and biases. The second part of the network consists of $J^* \cdot N^*$ copies of the network $\widehat{f}$.

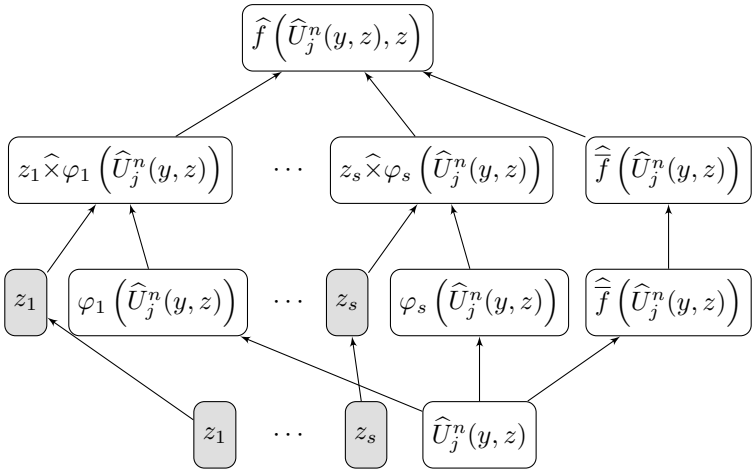


FIGURE 2. Flowchart of the ReLU neural network $\widehat{f}$ for uncertain flux functions. Gray neurons are shared over all j.

The following holds:

$$\begin{aligned}
\mathcal{M}(\widehat{f}) &= \mathcal{M}\left(\widehat{\widehat{f}}\right) + s(\mathcal{M}\left(\widehat{\varphi}\right) + \mathcal{M}\left(\widehat{\times}\right)) = O\left(\sqrt{N^*} + s^2\sqrt{N^*} + s\log_2(s\sqrt{N^*})\right) = O(s^2\sqrt{N^*}),\\
\mathcal{L}(\widehat{f}) &= \max\left\{\mathcal{L}\left(\widehat{\widehat{f}}\right), \mathcal{L}\left(\widehat{\varphi}\right) + \mathcal{L}\left(\widehat{\times}\right)\right\} = O(\log_2(s\sqrt{N^*})),\\
\mathcal{W}(\widehat{f}) &= \mathcal{W}\left(\widehat{\widehat{f}}\right) + s(\mathcal{W}\left(\widehat{\varphi}\right) + \mathcal{W}\left(\widehat{\times}\right)) = O(s^2\sqrt{N^*}).
\end{aligned} \tag{34}$$

Concatenating the two aforementioned parts of the network gives therefore rise to $L := N^* \cdot \mathcal{L}(\widehat{f})$ layers. Finally, a third part of the network makes the value of the s-dimensional vector z accessible to all layers of the other two parts of the network. This can be done using a subnetwork of width s, depth L and connectivity $O(sL)$. Adding the contributions from all three subnetworks, one obtains the claimed complexities of the theorem. To make the results presentable, we only report the complexities in the theorem. Note however that exact bounds can very easily be obtained from our calculations. □

3.3. **Extensions.** In what follows, we present a number of ways in which Theorem 3.2 and Theorem 3.5 can be extended. For simplicity, we will only prove these results for a fixed flux function. The extension to a parametric flux function is completely analogous to the proof of Theorem 3.5. Moreover, all results can also be generalized to systems of conservation laws, as long as it can be proven that some finite volume scheme, such as the Lax-Friedrichs scheme, converges to the true solution at a known convergence rate. We consider extensions to different neural network architectures (Section 3.3.1), different kinds of approximation (observables in Section 3.3.2 and space-time in Section 3.3.3) as well as multi-dimensional scalar conservation laws (Section 3.3.4).

3.3.1. *Residual and recurrent neural networks.* The network that has been constructed in the proof of Theorem 3.2 is very sparse and consists of subnetworks that are many times repeated. One possibility to reduce the size of the network is the introduction of skip connections, i.e. connections between non-consecutive hidden layers. These kind of networks are referred to as residual neural networks. Skip connections can most notably be used to simplify the neural network representing (25) (see also $\mathcal{L}$ in Figure 1). Note that there is also a strong connection between the structure of the constructed network and that of a recurrent neural network: the network to calculate $(\widehat{U}_j^{n+1}(y))_j$ from $(\widehat{U}_j^n(y))_j$ is independent from n. Casting a part of the constructed network as a recurrent network is thus another way to drastically reduce the size of the network.

3.3.2. *Observables.* In some applications one is not necessarily interested in the full solution of the conservation law, but rather in the image of the solution under some functional or so-called *observable* or *quantity of interest*. These can be expressed in the generic form

$$L_T(y,u) = \int_D \phi(x) g(u(T,x,y))dx, \tag{35}$$

where $\phi : D \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ are suitable test functions. This allows us to define the *parameters-to-observable* map

$$\mathcal{L}_T(y) : Y \to \mathbb{R} : y \mapsto L_T(y, u). \tag{36}$$

Provided that ϕ and g are sufficiently regular, it follows from Theorem 3.2 that $\mathcal{L}_T$ can also be approximated using DNNs of which the size only depends linearly on the input dimension d.

3.3.3. *Space-time approximations.* Theorem 3.2 approximates the solution of (8) on a grid at a fixed time $T > 0$. It is natural to ask the question whether this result can be extended to space-time $D \times [0, T]$. The following corollary shows that this is indeed possible, if one allows the network weights to grow with increasing N.

Corollary 3.6 (Space-time extension of Theorem 3.2)**.** *Let $T > 0$, $d \in \mathbb{N}$, $a, b \in \mathbb{R}$ with $a < b$ and $D = [a, b]$. For $y \in Y = [0, 1]^d$, denote by $u(t, \cdot, y)$ the solution at time t of (8) with a flux $f : [-C_0, C_0] \to \mathbb{R}$ that satisfies assumption (A1) and initial condition $u_0(\cdot, y)$ that satisfies assumptions (A2) and (A3). Then for every $N \in \mathbb{N}$ there exists a ReLU neural network $\widehat{U}^N : [0, T] \times D \times Y \to \mathbb{R}$ that satisfies the error bound*

$$\sup_{y \in Y} \left\| u(t, \cdot, y) - \widehat{U}^N(t, \cdot, y) \right\|_{L^1(D)} \leq \frac{2C_{TV}T \left(C_0 \left\| f'' \right\|_\infty + 21 \left(1 + \left\| f' \right\|_\infty \right)^2 \right) + 1}{\sqrt{N}} \tag{37}$$

for all $t \in [0, T]$. In addition, it holds that that for $N \in \mathbb{N}$,

$$\begin{aligned} \mathcal{M}(\widehat{U}^N) &= O\left(d^{\sigma_{\mathcal{M}}} N^{1+\eta_{\mathcal{M}}/2} + N^{5/2} \right), \\ \mathcal{L}(\widehat{U}^N) &= O(d^{\sigma_{\mathcal{L}}} N^{\eta_{\mathcal{L}}/2} + N), \\ \mathcal{W}(\widehat{U}^N), &= O(d^{\sigma_{\mathcal{W}}} N^{1+\eta_{\mathcal{W}}/2} + N^{3/2}), \\ \mathcal{B}(\widehat{U}^N) &= O(N). \end{aligned} \tag{38}$$

Proof. In the notation of the proof of Theorem 3.2, recall that $\widehat{U}_j^n(y)$ was the approximation we obtained for $u(t^n, x_j, y)$. We thus have access to an approximation to $u(\cdot, \cdot, y)$ on a grid in space-time. Our goal is to approximate $u(t, x, y)$ by some $\widehat{U}(t, x, y)$ for all (t, x) based on $\widehat{U}_j^n(y)$. First, we define

$$\alpha^n(t) = \max\left\{ \min\left\{ \frac{t^n - t}{\Delta t}, 1 \right\}, 0 \right\} \quad \text{and} \quad \beta_j(x) = \max\left\{ \min\left\{ \frac{x_j - x}{\Delta x}, 1 \right\}, 0 \right\}. \tag{39}$$

It holds that $\alpha^n(t) = 1$ if $t \leq t^{n-1}$ and $\alpha^n(t) = 0$ if $t \geq t^n$ and similarly for $\beta^j(x)$. In addition, both quantities can be written as a ReLU network, e.g. $\alpha^n(t) = \sigma(1 - \sigma(1 - (t^n - t)/\Delta t))$, where σ is the ReLU activation function. Then note that a good approximation of $u(t, x_{j+1/2}, y)$ would be given by

$$\widehat{U}_j^0 + \sum_{n=1}^{N} \alpha^n(t) \cdot (\widehat{U}_j^n(y) - \widehat{U}_j^{n-1}(y)). \tag{40}$$

As the multiplication operator can not be exactly written as a ReLU neural network, an approximation is necessary. Appendix A.4 introduces a ReLU neural network, which we denote by $\star$, consisting of one hidden layer with two neurons that mimics the multiplication operator adequately in our setting. We proceed by replacing the multiplication $\cdot$ by $\star$ in (40) and define

$$\widehat{U}_j(t, y) = \widehat{U}_j^0 + \sum_{n=1}^{N} \alpha^n(t) \star (\widehat{U}_j^n(y) - \widehat{U}_j^{n-1}(y)) \tag{41}$$

and similarly

$$\widehat{U}(t, x, y) = \widehat{U}(t, x_1, y) + \sum_{j=1}^{J} \beta_j(x) \star (\widehat{U}_j(t, y) - \widehat{U}_{j-1}(t, y)) \tag{42}$$

Now we calculate that for $t^{n-1} \leq t < t^n$,

$$\Delta x \sum_j \left| \widehat{U}_j(t, y) - \widehat{U}_j^{n-1}(y) \right| \leq \Delta x \sum_j \left| \widehat{U}_j^n(y) - \widehat{U}_j^{n-1}(y) \right| \leq 2F\mathrm{TV}(\widehat{u}_0^\epsilon)\Delta t \tag{43}$$

by (93) in Appendix A.3, where $F = \left\|f'\right\|_{L^\infty([-C_0,C_0])}$. In addition, it holds for $x_{j-1} \leq x < x_j$ and $t^{n-1} \leq t < t^n$ that

$$\begin{aligned}\left|\widehat{U}(t,x,y) - \widehat{U}_j(t,y)\right| &\leq \left|\widehat{U}_{j-1}(t,y) - \widehat{U}_j(t,y)\right| \\ &\leq \left|\widehat{U}_{j-1}(t,y) - \widehat{U}_{j-1}^{n-1}(y)\right| + \left|\widehat{U}_{j-1}^{n-1}(y) - \widehat{U}_j^{n-1}(y)\right| + \left|\widehat{U}_j^{n-1}(y) - \widehat{U}_j(t,y)\right|.\end{aligned} \tag{44}$$

Since the Lax-Friedrichs method is a monotone, consistent and conservative scheme it satisfies Harten's lemma and is therefore total variation diminishing (TVD), meaning that

$$\begin{aligned}\sum_j \left|\widehat{U}_{j-1}^{n-1}(y) - \widehat{U}_j^{n-1}(y)\right| &\leq \sum_j \left|\widehat{U}_{j-1}^{0}(y) - \widehat{U}_j^{0}(y)\right| \\ &= \sum_j \left|\widehat{u}_0^\epsilon(x_{j-1},y) - \widehat{u}_0^\epsilon(x_j,y)\right| \leq \mathrm{TV}(\widehat{u}_0^\epsilon(\cdot,y))\end{aligned} \tag{45}$$

This allows us to quantify the accuracy of our ReLU neural network approximation with respect to the cell-averages from the proof of Theorem 3.2, where we use the CFL condition $\Delta x = F\Delta t$,

$$\begin{aligned}\left\|\widehat{U}(t,\cdot,y) - \widehat{\mathcal{U}}^n(\cdot,y)\right\|_{L^1(D)} &\leq 4F\mathrm{TV}(\widehat{u}_0^\epsilon(\cdot,y))\Delta t + \mathrm{TV}(\widehat{u}_0^\epsilon(\cdot,y))\Delta x \\ &= 5F\mathrm{TV}(\widehat{u}_0^\epsilon(\cdot,y))\Delta t \\ &\leq 6(1+F)^2\mathrm{TV}(\widehat{u}_0^\epsilon(\cdot,y))TN^{-1/2}.\end{aligned} \tag{46}$$

This error bound, together with Theorem 3.2, proves (37).

Finally, we note that the complexity estimates are easy to deduct from the above construction and are the same as those of Theorem 3.2 (up to a constant). □

3.3.4. *Multidimensional scalar conservation laws.* We have so far only treated one-dimensional scalar conservation laws. In real life applications, the multidimensional case may be more prevalent. For $m \in \mathbb{N}$, the m-dimensional version of (8) is given by

$$\begin{cases}\partial_t u(t,x,y) + \sum_{j=1}^m \partial_{x_j} f_j(u(t,x,y)) = 0, \\ u(0,x,y) = u_0(x,y),\end{cases} \tag{47}$$

for all $x \in D = [a,b]^m$, $y \in Y = [0,1]^d$ and $t \in [0,T]$. The generalization of assumptions (A1)-(A3) is straightforward. In this setting, we obtain the following approximation result.

Corollary 3.7 (Extension of Theorem 3.2 to multidimensional grids)**.** *Let $T > 0$, $d, m \in \mathbb{N}$, $a, b \in \mathbb{R}$ with $a < b$ and $D = [a,b]^m$. For $y \in Y = [0,1]^d$, denote by $u(T,\cdot,y)$ the solution at time $t = T$ of (8) with flux functions $f_j : \mathbb{R} \to \mathbb{R}$, $1 \leq j \leq m$, that satisfy assumption (A1) and initial condition u_0 that satisfies assumptions (A2) and (A3). Then for every $N \in \mathbb{N}$, there exist a ReLU neural network $\widehat{\mathcal{U}}^N$ that satisfies the error bound*

$$\sup_{y\in Y} \left\|u(T,\cdot,y) - \widehat{\mathcal{U}}^N(\cdot,y)\right\|_{L^1(D)} \leq \frac{2C_{TV}T\left(C_0\left\|f''\right\|_\infty + 18\left(1 + \left\|f'\right\|_\infty\right)^2\right) + 1}{\sqrt{N}}.t \tag{48}$$

In addition, it holds that

$$\begin{aligned}\mathcal{M}(\widehat{\mathcal{U}}^N) &= O\left(d^{\sigma_{\mathcal{M}}}N^{m+\eta_{\mathcal{M}}/2} + mN^{m+3/2}\right), \\ \mathcal{L}(\widehat{\mathcal{U}}^N) &= O(d^{\sigma_{\mathcal{L}}}N^{\eta_{\mathcal{L}}/2} + N), \\ \mathcal{W}(\widehat{\mathcal{U}}^N), &= O(d^{\sigma_{\mathcal{W}}}N^{m+\eta_{\mathcal{W}}/2} + mN^{m+1/2}), \\ \mathcal{B}(\widehat{\mathcal{U}}^N) &= O(1).\end{aligned} \tag{49}$$

Proof. The proof of the corollary only requires a small generalization of the proof of Theorem 3.2. Step 1 still holds for multidimensional scalar conservation laws. We will numerically approximate the solution of (47) using dimensional splitting methods, i.e. by solving one space direction at a time. For $1 \leq i,k \leq m$, let $e_i \in \mathbb{R}^m$ be the vector that satisfies $(e_i)_k = \delta_{ik}$. The multidimensional update formula of the Lax-Friedrichs scheme (25) is then given by

$$\widehat{U}_j^{n+1}(y) = \sum_{i=1}^m \left(\frac{\widehat{U}_{j-e_i}^n(y) + \widehat{U}_{j+e_i}^n(y)}{2} - \frac{\Delta t}{2\Delta x_i}(\widehat{f}_i(\widehat{U}_{j+e_i}^n(y)) - \widehat{f}_i(\widehat{U}_{j-e_i}^n(y)))\right), \tag{50}$$

where j is an m-dimensional vector that indicates the cell location. The rest of step 2 can be taken over mutatis mutandis. It only remains to determine the size of the network. As only the connectivity and maximum width of the network depend on m, $\mathcal{L}(\widehat{\mathcal{U}}^N)$ and $\mathcal{B}(\widehat{\mathcal{U}}^N)$ are the same as in Theorem 3.2. In the notation of the proof of Theorem 3.2, we let N be the number of time steps, such that there are now $O(N^m)$ grid points. The first explains the factor $O(N^m)$ in the first term in $\mathcal{M}(\widehat{\mathcal{U}}^N)$ and $\mathcal{W}(\widehat{\mathcal{U}}^N)$. For every f_i, the size of the approximating network is given by Lemma 2.5. Consequently, the second term $\mathcal{W}(\widehat{\mathcal{U}}^N)$ follows directly as there are m flux functions of maximal width $2\sqrt{N}$ that need to be evaluated on $O(N^m)$ grid points. For $\mathcal{M}(\widehat{\mathcal{U}}^N)$, the result is obtained by multiplying with the number of time steps. This proves the result. □

4. Analysis of the generalization error

In the previous section, we have established that the solutions of scalar conservation laws with parametric flux and initial data can be efficiently approximated using ReLU neural networks, without suffering from the curse of dimensionality. In practice, one must try to retrieve (or *train*) this neural network based on an available, finite data set. Details of this training procedure can be found in the next subsection. Next, we estimate how accurately this *trained* neural network approximates the true solution. This will be done by proving an upper bound on the so-called cumulative generalization error.

4.1. Training neural networks.

For $d \in \mathbb{N}$, let $Y = [0,1]^d$ be the space of parameters of both the flux and initial data (i.e. we simplify notation by replacing $Y \times Z$ with Y) and let (Y, Σ, μ) be a complete probability space. For $T > 0$, the goal is to accurately approximate the mapping $y \in Y \mapsto u(T, \cdot, y)$ by a neural network u_θ (in the sense of Definition 2.4) where $\theta \in \Theta$ are the parameters of the network. To do so, we draw $M \in \mathbb{N}$ random parameters from Y according to μ, which we will denote by $\mathcal{S} = (y_1, \dots, y_M)$, and create the *training set* $\mathbb{S} = \{(y, u(T, \cdot, y)) \,|\, y \in \mathcal{S}\}$. Using this training set, we define the loss function

$$\mathcal{J}(\theta; \mathcal{S}, \lambda) = \frac{1}{M} \sum_{i=1}^{M} \left\| u(T, \cdot, y_i) - u_\theta(\cdot, y_i) \right\|_{L^1(D)} + \lambda \mathcal{R}(\theta), \tag{51}$$

where the first sum measures the discrepancy between the true solution and its approximation and the last term is a regularization term that prevents overfitting and hence improves the generalization capabilities of the network. The network hyperparameter $\lambda > 0$ controls the extent of regularization. A popular choice of the regularization function is $\mathcal{R}(\theta) = \|\theta\|_p^p$ for $p = 1, 2$. The *training* of the neural network (finding the best neural network approximation) then boils down to finding the parameter $\theta^*(\mathcal{S})$ that minimizes the loss function:

$$\theta^*(\mathcal{S}) = \arg\min_{\theta \in \Theta} \mathcal{J}(\theta; \mathcal{S}, \lambda). \tag{52}$$

This possibly highly non-convex optimization problem is usually solved using an algorithm based on stochastic gradient descent. If the training was successful, the network $u_{\theta^*(\mathcal{S})}$ will be a good approximation of u on $\mathcal{S}$, but there is no a priori guarantee that $u_{\theta^*(\mathcal{S})}$ is close to u on $Y \setminus \mathcal{S}$. This is the topic of the next section.

4.2. Estimate on the cumulative generalization error.

In this section we will prove that if the training set is large enough, the neural network $u_{\theta^*(\mathcal{S})}$ will on average be a good approximation of the true solution u. In order to do so, we will introduce some quantities. For a vector of weights and biases $\theta \in \Theta$, we define the training error

$$\mathcal{E}_T(\theta, \mathcal{S}) = \frac{1}{M} \sum_{i=1}^{M} \left\| u(T, \cdot, y_i) - u_\theta(\cdot, y_i) \right\|_{L^1(D)} \tag{53}$$

and the generalization error

$$\mathcal{E}_G(\theta) = \int_Y \left\| u(T, \cdot, y) - u_\theta(\cdot, y) \right\|_{L^1(D)} d\mu(y). \tag{54}$$

The training error is nothing more than the loss function without the regularization term and quantifies the accuracy of u_θ on $\mathcal{S}$. We will be particularly interested in the choice $\theta = \theta^*(\mathcal{S})$, where $\theta^*(\mathcal{S})$ is the vector that minimizes the loss function. In this case, the generalization error quantifies how well the approximation $u_{\theta^*(\mathcal{S})}$ generalizes from $\mathcal{S}$ to Y. Note that we are interested in the generalization error, but we have only access to the training error. Intuitively, one can generally not expect to infer bounds on the generalization error from the training error as the training set might be ill-conditioned, e.g. it might be that $y_1 = y_2 = \ldots = y_M$. However, we can hope for finding a connection between $\mathcal{E}_T(\theta, \mathcal{S})$ and

$\mathcal{E}_G(\theta)$ that holds with a certain probability or that holds averaged over all training sets. Keeping this in mind, we define the cumulative training error

$$\overline{\mathcal{E}}_T = \int_{Y^M} \mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S}) d\mu^M(\mathcal{S}) \tag{55}$$

and the cumulative generalization error

$$\overline{\mathcal{E}}_G = \int_{Y^M} \mathcal{E}_G(\theta^*(\mathcal{S})) d\mu^M(\mathcal{S}). \tag{56}$$

To prove an upper bound on $\overline{\mathcal{E}}_G$ we first decompose the error as such,

$$\mathcal{E}_G(\theta^*(\mathcal{S})) \leq \left|\mathcal{E}_G(\theta^*(\mathcal{S})) - \mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S})\right| + \mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S}), \tag{57}$$

and then take the expectation under μ^M and the supremum over all $\theta \in \Theta$ to obtain,

$$\overline{\mathcal{E}}_G \leq \overline{\mathcal{E}}_T + \mathbb{E}\left[\sup_{\theta\in\Theta} \left|\mathcal{E}_G(\theta) - \mathcal{E}_T(\theta, \mathcal{S})\right|\right]. \tag{58}$$

This inequality provides an easy recipe to bound the cumulative generalization error. We assume that $\overline{\mathcal{E}}_T$ can be made arbitrarily small by letting the stochastic gradient descent algorithm run for enough epochs. There are more rigorous upper bounds available [2, 20], but these tend to be large overestimates of the true value of $\overline{\mathcal{E}}_T$. Alternatively, it is common practice to retrain networks multiple times. One could then estimate $\overline{\mathcal{E}}_T$ by the average of the training errors. Hence, it only remains to find an upper bound on the left term of the right-hand side of (73). This term is often called the generalization gap, as it measures the discrepancy between generalization and training error. We will find an upper bound for the generalization gap by adapting [20, Corollary 4.15] to our setting. To do so, we introduce the notion of the covering number of a metric space, which will play a crucial role in bounding the generalization gap. In addition, we prove a number of auxiliary results in Appendix B.

Definition 4.1. *Let (E, d) be a metric space and let $r \in [0, +\infty]$. Then we refer to*

$$\mathcal{C}_r^{(E,d)} := \inf\left\{n \in \mathbb{N}_0 \mid \exists A \subseteq E : (|A| = n) \wedge (\forall x \in E : \exists a \in A : d(x, a) \leq r)\right\} \tag{59}$$

as the covering number of (E, d).

Theorem 4.2. *Let $R \geq 1$, $k, d, W, L, M \in \mathbb{N}$, $a, b, \alpha, \beta \in \mathbb{R}$ with $a < b$ and $\beta - \alpha \geq 1$ and let Θ be the k-dimensional parameter space corresponding to (α, β)-clipped ReLU neural networks Φ_θ such that $\mathcal{W}(\Phi_\theta) \leq W$, $\mathcal{L}(\Phi_\theta) \leq L$ and $\mathcal{B}(\Phi_\theta) \leq R$ for all $\theta \in \Theta$ (cf. Definitions 2.2 and 2.3). Let $D = [a, b]$ and let $D \times [0, 1]^d$ be the domain of the solution maps of the associated neural networks (cf. Definition 2.4). Let $\mathcal{S}$ be the training set drawn according to μ^M such that $|\mathcal{S}| = M$. For $\theta \in \Theta$, let $\mathcal{E}_T(\theta, \mathcal{S})$ and $\mathcal{E}_G(\theta)$ be given by (53) and (54). It then holds that*

$$\mathbb{E}\left[\sup_{\theta\in\Theta} \left|\mathcal{E}_T(\theta, \mathcal{S}) - \mathcal{E}_G(\theta)\right|\right] \leq \frac{12(\beta - \alpha)(b - a)L(W + 1)\sqrt{\ln\left(2M^{1/3L}R(W + 1)\right)}}{\sqrt{M}}. \tag{60}$$

Moreover, the following simplified bound also holds,

$$\mathbb{E}\left[\sup_{\theta\in\Theta} \left|\mathcal{E}_T(\theta, \mathcal{S}) - \mathcal{E}_G(\theta)\right|\right] \leq \frac{12(\beta - \alpha)(b - a)L(W + 1)^2\sqrt{\ln\left(2MR\right)}}{\sqrt{M}}. \tag{61}$$

Proof. We first generalize the proof of [20, Lemma 4.13]. For every $\theta \in \Theta$, the measurability of the mapping $Y^M \to \mathbb{R} : \mathcal{S} \to \mathcal{E}_T(\theta, \mathcal{S})$ follows from Lemma B.1. We have that for every $\theta \in \Theta$ it holds that $\mathcal{E}_T(\theta, \{y_i\})$, $1 \leq i \leq M$, are i.i.d. random variables and therefore

$$\mathbb{E}\left[\mathcal{E}_T(\theta, \mathcal{S})\right] = \mathcal{E}_G(\theta). \tag{62}$$

Furthermore the assumption that $u_\theta, u \in [\alpha, \beta]$ together with definitions (53) and (54) ensure that for all $\theta \in \Theta$ and $1 \leq i \leq M$ that

$$\left(\mathbb{E}\left[\sup_{\theta\in\Theta} \left|\mathcal{E}_T(\theta, \{y_i\}) - \mathcal{E}_G(\theta)\right|^p\right]\right)^{1/p} \leq (\beta - \alpha)(b - a). \tag{63}$$

In addition, Lemma B.4 with $p = 1$ tells us that $\mathcal{E}_T(\theta, \mathcal{S})$ is Lipschitz in θ with a Lipschitz constant of at most $(b-a)L^*$ where $L^* := LR^{L-1}(W+1)^L$. Combining all the previous observations allows us to use [20, Corollary 4.12] to conclude that for any $C > 0$, $q \geq 2$,

$$\begin{aligned}&\left(\mathbb{E}\left[\sup_{\theta\in\Theta}\left|\mathcal{E}_T(\theta,\mathcal{S}) - \mathcal{E}_G(\theta)\right|^q\right]\right)^{1/q}\\ &\qquad\leq \left(\mathcal{C}^{(\Theta,\|\cdot\|_\infty)}_{\frac{C(\beta-\alpha)(b-a)\sqrt{q-1}}{(b-a)L^*\sqrt{M}}}\right)^{1/q}\left(\frac{2(C+1)(\beta-\alpha)(b-a)\sqrt{q-1}}{\sqrt{M}}\right),\end{aligned} \tag{64}$$

which is the equivalent of [20, Lemma 4.13]. We proceed by following the steps from the proof of [20, Proposition 4.14], mutatis mutandis. We restrict ourselves to the main steps and refer to [20] for the detailed calculations. Lemma B.5 gives us that for $r > 0$,

$$\mathcal{C}^{(\Theta,\|\cdot\|_\infty)}_r \leq \max\{1, (2R/r)^k\}. \tag{65}$$

Following the steps from [20, Proposition 4.14] with $\kappa_C \leftarrow 2RL^*\sqrt{M}/(C(\beta-\alpha))$ and $C \leftarrow 1$, we obtain that

$$\mathbb{E}\left[\sup_{\theta\in\Theta}\left|\mathcal{E}_T(\theta,\mathcal{S}) - \mathcal{E}_G(\theta)\right|\right] \leq \frac{4(\beta-\alpha)(b-a)\sqrt{e\max\{1, k\ln\left(4R^2M(L^*)^2\right)\}}}{\sqrt{M}} \tag{66}$$

Next, we roughly follow the steps of [20, Corollary 4.15]. Note that $k \leq LW(W+1) \leq L(W+1)^2$ and $4L^2 \leq 2^2 \cdot 2^{2(L-1)} \leq 2^{2L}$ to see that

$$\begin{aligned}k\ln\left(4R^2M(L^*)^2\right) &\leq L(W+1)^2\ln\left(4R^2M(L^*)^2\right)\\ &= L(W+1)^2\ln\left(4ML^2R^{2L}(W+1)^{2L}\right)\\ &\leq L(W+1)^2\ln\left(M2^{2L}R^{2L}(W+1)^{2L}\right)\end{aligned} \tag{67}$$

Since it holds that $L, R, W \geq 1$ we also have that

$$\begin{aligned}\ln\left(M2^{2L}R^{2L}(W+1)^{2L}\right) &= 3L\ln\left(\left[M2^{2L}R^{2L}(W+1)^{2L}\right]^{1/3L}\right)\\ &\leq 3L\ln\left(2M^{1/3L}R(W+1)\right).\end{aligned} \tag{68}$$

Combining the previous observations with the fact that $2M^{1/3L}R(W+1) \geq e$ and therefore $3L\ln\left(2M^{1/3L}R(W+1)\right) \geq 1$ gives us

$$\begin{aligned}&\mathbb{E}\left[\sup_{\theta\in\Theta}\left|\mathcal{E}_T(\theta,\mathcal{S}) - \mathcal{E}_G(\theta)\right|\right]\\ &\qquad\leq \frac{4(\beta-\alpha)(b-a)\sqrt{e\cdot 3L^2(W+1)^2\ln\left(2M^{1/3L}R(W+1)\right)}}{\sqrt{M}}\\ &\qquad\leq \frac{12(\beta-u)(b-a)L(W+1)\sqrt{\ln\left(2M^{1/3L}R(W+1)\right)}}{\sqrt{M}},\end{aligned} \tag{69}$$

which gives us our main result. We now further simplify the bound. First notice that

$$2(W+1) \leq 2\cdot 2^{(W+1)-1} \leq 2^{W+1}, \tag{70}$$

leading us to the bound

$$\begin{aligned}\ln\left(2M^{1/3L}R(W+1)\right) &\leq (W+1)\ln\left(\left[2M^{1/3L}R2^{W+1}\right]^{1/(W+1)}\right)\\ &\leq (W+1)\ln(2MR).\end{aligned} \tag{71}$$

We finally obtain

$$\mathbb{E}\left[\sup_{\theta\in\Theta}\left|\mathcal{E}_T(\theta,\mathcal{S}) - \mathcal{E}_G(\theta)\right|\right] \leq \frac{12(\beta-\alpha)(b-a)L(W+1)^2\sqrt{\ln\left(2MR\right)}}{\sqrt{M}}. \tag{72}$$

□

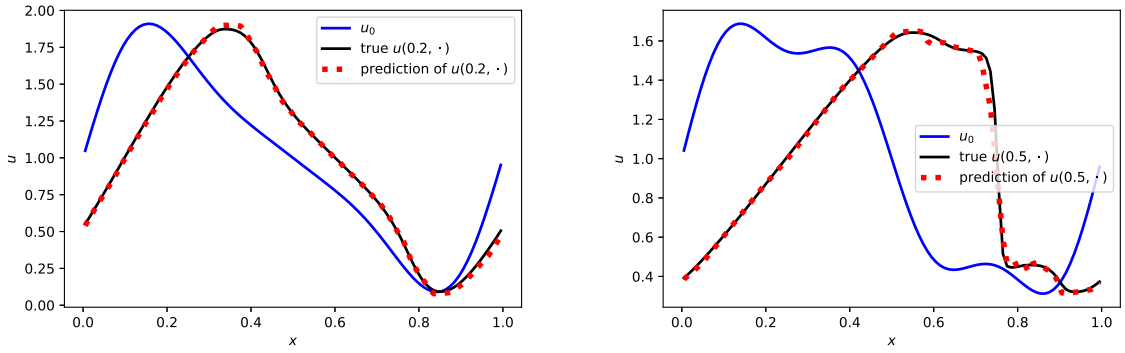


FIGURE 3. Example of random initial data, the corresponding true solution at $T = 0.2$ and $T = 0.5$ and the solution predicted by the neural network.

Combining (58) with Theorem 4.2 then provides an elegant, yet explicit and computable upper bound on the cumulative generalization error $\overline{\mathcal{E}}_G$, as $\overline{\mathcal{E}}_T$ can be estimated by e.g. the average of the training errors for multiple random seeds.

Corollary 4.3. *Assume the setting and notation of Theorem 4.2. It holds that,*

$$\overline{\mathcal{E}}_G \leq \overline{\mathcal{E}}_T + \frac{12(\beta-\alpha)(b-a)L(W+1)^2\sqrt{\ln(2MR)}}{\sqrt{M}}. \tag{73}$$

5. NUMERICAL EXPERIMENTS

We present some numerical experiments to empirically validate the theoretical results from the previous section. In particular, we will show that the generalization error grows at most polynomially as a function of the dimension of the parameter space for a neural network of fixed size.

5.1. **Methodology.** We will train networks that approximate the mapping from (x, y) to $u(T, x, y)$ for $T = 0.1$. Instead of calculating the L^1-norms in the training (53) and generalization error (54) exactly, we make the approximation

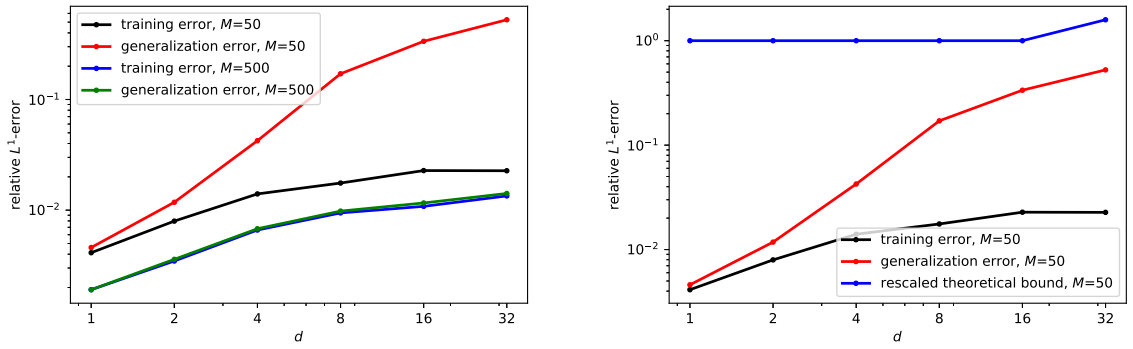
$$\left\|u(T,\cdot,y) - u_\theta(\cdot,y)\right\|_{L^1(D)} \approx \sum_{j=1}^{J} \left|u(T,x_j,y) - u_\theta(x_j,y)\right|, \tag{74}$$

where x_j are $J = 100$ equidistant points on $[a, b]$. Hence, M iid generated parameters $y_1, \ldots, y_M \sim U([0,1]^d)$ give rise to a training set of size $M \cdot J$. In the first experiment, we will consider random initial data and a fixed flux and vice versa in the second experiment. For both experiments, we consider networks with $2, 4, 6, 8$ hidden layers with each $5, 10, 15, 20$ neurons and we choose the network architecture that minimizes the median validation error over 5 runs for $d = 4$. In our experiment, this is a DNN with 4 hidden layers and 20 neurons per layer. We then use this fixed architecture for all input dimensions and report the training and generalization error averaged over 5 runs. We minimize the training error (53) using Adam [21] and train the network for 10^4 epochs.

5.2. **Fixed flux.** We first consider the one-dimensional inviscid Burgers' equation, i.e. scalar conservation law (8) with $f(u) = u^2/2$. We take the initial data in the form of a truncated Karhunen-Loève expansion,

$$u_0(x,y) = 1 + \sum_{k=1}^{d} y_k 2^{1-k} \sin(kx), \tag{75}$$

where $x \in [0,1], y \in [0,1]^d$. In order to create the training set, we generate M independent realizations of y and do the same for the test set. We generate the training (and test) samples using the Rusanov flux, second-order reconstruction with the minmod limiter and SSP2-RK time stepping. An example of $u_0(\cdot, y)$ and $u(T, \cdot, y)$ for some y is given in Figure 3. In the same figure, the prediction by the trained neural network is shown as well. In Figure 4a, we examine how the training and generalization error depend on the parameter dimension d for two training sets of different size. In both cases, the logarithmic plot reveals that the training and generalization errors only polynomially depend on the parameter dimension d. For the smallest training set, of size $M = 50$, the generalization error is high for



(A) Dependence of the training and generalization error on d for a DNN approximating the solution of Burgers' equation with parameteric initial data.

(B) Training and generalization error for $M = 50$ in the fixed flux case. A rescaled version of our theoretical bound is shown (rescaled by a factor of $9 \cdot 10^2$).

FIGURE 4. Experiments for Burgers' equation with parameteric initial data with fixed flux.

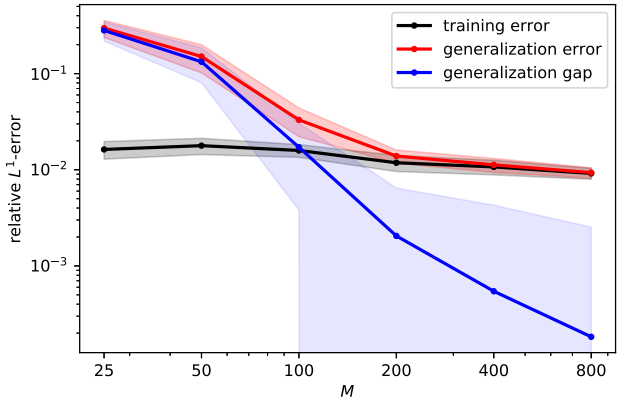


FIGURE 5. Dependence of the average training error, generalization error and generalization gap on M for $d = 8$ and 25 realizations. The shaded area indicates the standard error.

large d and significantly larger than the training data. This signals that the network is not trained in a satisfactory way, in this case due to insufficient training data. For $M = 500$, the errors are much smaller and the network generalizes well.

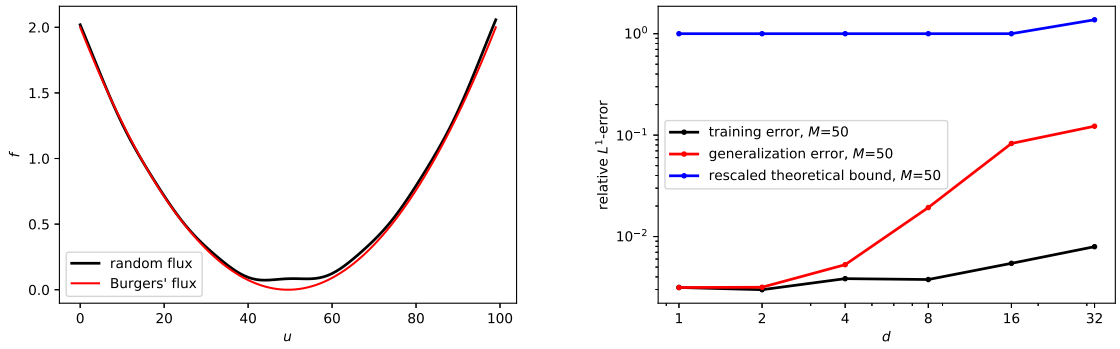
Figure 4b also shows a comparison between a theoretical bound and the empirically observed errors. We use Corollary 4.3 with $L = 4$ and $W = \max\{d, 20\}$ for the generalization gap and Corollary 3.4 to get an estimate on the approximation error. The theoretical error is rescaled by a factor of $9 \cdot 10^2$ for clarity, making clear that the bound is a gross overestimate, as is common for these type of machine learning estimates.

From Figure 4a, it was already clear that the generalization gap decreases as the size of the training set increases. To further explore this observation, we train DNNs for $d = 8$ and $M = 25, 50, 100, 200, 400, 800$ and repeat this 25 times. In Figure 5, the average training and generalization errors (and standard errors) are shown. It is clear that the generalization gap decreases as M grows large. As a consequence, the generalization error converges to the training error.

5.3. **Fixed initial condition.** We repeat the experiment from the previous section for the case where the flux function is random, but the initial data is fixed. We choose for $x \in [0, 1]$,

$$u_0(x) = 1 + \sin(x). \tag{76}$$

Following the numerical experiment in [30], the random component of the flux function will be approximated by a Karhunen-Loève expansion, of which the eigenvalues λ_k and eigenfunctions φ_k are those



(A) Typical example of a random flux function together with the unperturbed flux function of the inviscid Burgers' equation.

(B) Dependence of the training and generalization error on d. Our theoretical bound is shown for comparison, after rescaling with a factor of $1 \cdot 10^3$.

FIGURE 6. Experiments for Burgers' equation with fixed initial data and parametric flux.

given by

$$\int_A C(u_1, u_2)\varphi_k(u_1)du_1 = \lambda_k \varphi_k(u_1), \quad u_2 \in [0, 2], \tag{77}$$

where C is for instance an exponential covariance kernel,

$$C(u_1, u_2) = \sigma^2 \exp\{-|u_1 - u_2|/\eta\}, \tag{78}$$

where $\sigma = 1$ and $\eta = 3$. We then consider flux functions of the form

$$f(u, z) = \frac{u^2}{2} + \sum_{k=1}^{s} z_k \sqrt{\lambda_i}\varphi_k(u), \tag{79}$$

where $z \in [0, 1]^s$. In [30, Section 6.1] it is demonstrated that the eigenvalues decay as $\lambda_i \sim i^{-2.5}$, which is in agreement with our assumptions on the fast decay of Karhunen-Loève eigenvalues. In addition, the eigenfunctions are linear combinations of sines and cosines.

A typical realization of this flux function is shown in Figure 6a. In order to create the training set, we generate M independent realizations of z and do the same for the test set. We generate the training (and test) samples using the Godunov flux, second-order reconstruction with the minmod limiter and SSP2-RK time stepping.

In Figure 6b, we see that the dependence of the training and generalization error on d is only polynomially, as in the previous experiment. This time the network already generalizes well for $M = 50$. A comparison between our theoretical error bound and the empirically observed errors reveals a large difference that is often seen for such bounds. Note that the theoretical bound in Figure 6b is rescaled by a factor of $1 \cdot 10^3$ for clarity. As in the fixed flux case, the bound massively overestimates the empirically observed error.

6. Discussion

ReLU neural networks are widely used in the approximation of PDE solutions and have been proven to overcome the curse of dimensionality (CoD) in a number of settings. However, there is a paucity of theoretical results on the neural network approximation of (parametric) hyperbolic conservation laws. The contributions of this paper to overcoming this paucity can be summarized as follows,

- Under mild assumptions and for parametric initial data that can be approximated by a ReLU neural network without incurring the CoD in the parameter dimension, we have proven that ReLU neural networks overcome the CoD for scalar conservation laws with parametric initial data (Theorem 3.2). The error bound is explicit and the bounds on the weights and biases are independent of the chosen accuracy. The proof relies amongst others on the convergence of the Lax-Friedrichs method and a stability result for hyperbolic conservation laws.
- Theorem 3.5 proves under similar assumptions that CoD in the parameter dimensions can also be overcome when the flux function is parametric as well. Multiple extensions are considered,

such as a neural network approximation in space-time (Corollary 3.6) and an approximation for multi-dimensional scalar conservation laws (Corollary 3.7).
- By adapting the results of [20] to our setting, we prove a rigorous upper bound on the generalization gap i.e., the gap between generalization and training error (Theorem 4.2). This allows us to prove an estimate on the (accumulative) generalization error in terms of the (accumulative) training error, the number of training samples and the size of the neural network.

The theoretical error bounds are illustrated by a number of numerical experiments in Section 5. These clearly demonstrate how the training and generalization error only grow polynomially in terms of the parameter dimension and how the gap between generalization and training error converges to zero, when when the number of training samples is increased, as predicted by the theoretical results.

We conclude our discussion by pointing out possible limitations and suggestions for further research,

- We have only focused on neural netwok approximations in the *supervised learning* paradigm. In this setting, for every parameter in the training set one needs to (approximately) solve the conservation law, e.g. by using a traditional numerical method, which might still be computationally expensive. Fortunately, it is possible to extend to ideas of the paper to suitable unsupervised learning methods such as *weak PINNs* [12], as well as to operator learning methods using the generic error estimates of [11].
- Corollary 4.3 does not prove that the generalization error can be made arbitrarily small, it only provides a computable upper bound in terms of the size of the training set and network, as well as the (cumulative) training error. It is straightforward to argue that the latter is small when the optimization algorithm finds a global minimum and the training set and hypothesis space are large enough [9, Remark 3.12]. More rigorous results are available under mild assumptions, e.g. [2, 20], but are generally worst-case upper bounds that are very large overestimates of the true value.
- The numerical experiments in Section 5 show that Corollary 4.3 is an overestimate. One could investigate whether this bound can be made sharper using different techniques. In addition, one could try to lift the restriction to *clipped* ReLU neural networks. Extensions to other bounded activation functions such as tanh are immediate using results from [10, 16].
- We consider the case of parametric conservation laws, which arise in many different applications such as in UQ and optimal design. However, in many other applications, one will encounter situations where such parametrizations are not necessarily available [26]. In such contexts, it is imperative to learn the entire solution operator, for instance corresponding to the scalar conservation law. This is the domain of *operator learning* and rigorous error bounds for one such operator learning framework (DeepONets) for scalar conservation laws, adapting techniques already considered here, have been presented in [23].

References

[1] R. Abgrall and R. Crisovan. Model reduction using l1-norm minimization as an application to nonlinear hyperbolic problems. *Internat. J. Numer. Methods Fluids*, 87:628–651, 2018.

[2] C. Beck, A. Jentzen, and B. Kuckuck. Full error analysis for the training of deep neural networks. *Infinite Dimensional Analysis, Quantum Probability and Related Topics*, page 2150020, 2022.

[3] J. Berner, P. Grohs, and A. Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

[4] H. Bijl, D. Lucor, S. Mishra, and S. Ch. *Uncertainty quantification in computational fluid dynamics.* Springer, 2014.

[5] A. Borzi and V. Schulz. *Computational optimization of systems governed by partial differential equations.* SIAM, 2012.

[6] R. E. Caflisch et al. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 1998:1–49, 1998.

[7] R. Crisovan, D. Torlo, R. Abgrall, and S. Tokareva. Model order reduction for parametrized nonlinear hyperbolic problems as an application to uncertainty quantification. *J. Comput. Appl. Math*, 348:466–489, 2019.

[8] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[9] T. De Ryck, A. D. Jagtap, and S. Mishra. Error estimates for physics informed neural networks approximating the Navier-Stokes equations. *arXiv preprint arXiv:2203.09346*, 2022.

[10] T. De Ryck, S. Lanthaler, and S. Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 2021.

[11] T. De Ryck and S. Mishra. Generic bounds on the approximation error for physics-informed (and) operator learning. *arXiv preprint arXiv:2205.11393*, 2022.

[12] T. De Ryck, S. Mishra, and R. Molinaro. wPINNs: Weak physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws. *arXiv preprint*, 2022.

[13] T. De Ryck, S. Mishra, and D. Ray. On the approximation of rough functions with deep neural networks. *SeMA*, 2022.

[14] W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.
[15] D. Elbrächter, D. Perekrestenko, P. Grohs, and H. Bölcskei. Deep neural network approximation theory. *IEEE Transactions on Information Theory, submitted Jan. 2019, revised*, June 2020.
[16] I. Gühring and M. Raslan. Approximation rates for neural networks with encodable weights in smoothness spaces. *Neural Networks*, 134:107–130, 2021.
[17] J. H. Hesthaven. *Numerical methods for conservation laws: From analysis to algorithms*. SIAM, 2018.
[18] H. Holden and N. H. Risebro. *Front tracking for hyperbolic conservation laws*, volume 152. Springer, 2011.
[19] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.
[20] A. Jentzen and T. Welti. Overall error analysis for the training of deep neural networks via stochastic gradient descent with random initialisation, 2020.
[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[22] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric pdes. *Constructive Approximation*, pages 1–53, 2021.
[23] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for DeepOnets: A deep learning framework in infinite dimensions, 2021.
[24] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861 – 867, 1993.
[25] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.
[26] L. Lu, P. Jin, and G. E. Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.
[27] K. O. Lye, S. Mishra, and R. Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms. *European Journal of Applied Mathematics*, 2020.
[28] K. O. Lye, S. Mishra, and D. Ray. Deep learning observables in computational fluid dynamics. *Journal of Computational Physics*, page 109339, 2020.
[29] K. O. Lye, S. Mishra, D. Ray, and P. Chandrashekar. Iterative surrogate model optimization (ISMO): An active learning algorithm for pde constrained optimization with deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 374:113575, 2021.
[30] S. Mishra, N. H. Risebro, C. Schwab, and S. Tokareva. Numerical solution of scalar conservation laws with random flux functions. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):552–591, 2016.
[31] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: an introduction*. Springer, 2015.
[32] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
[33] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
[34] C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Analysis and Applications*, 17(01):19–55, 2019.
[35] A. Stuart. Inverse problems: A bayesian perspective. *Acta Numerica*, pages 451–559, 2010.
[36] D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

Appendix A. Additional material for Section 3

A.1. **Proof of Lemma 2.5.**

Proof. Let $J \in \mathbb{N}$ a natural number. We will explicitly construct Φ^J as a linear interpolation of f and prove that the resulting network satisfies the claimed bounds. Define

$$\mathcal{X}_J = \left\{ (x_{-1}, \dots, x_{J+1}) \,|\, x_{-1} + \left|f(a)\right| = a = x_0 < \dots < x_J = b = x_{J+1} - \left|f(b)\right| \right\}. \tag{80}$$

Note that every vector in $\mathcal{X}_J$ defines a (not necessarily uniform) grid on $[a, b]$, complemented by two additional grid points outside of $[a, b]$. For every $j \in \{0, \dots, J\}$ we define the hat function ρ_j through

$$\rho_j(x) = \frac{\sigma(x - x_{j-1}) - \sigma(x - x_j)}{x_j - x_{j-1}} - \frac{\sigma(x - x_j) - \sigma(x - x_{j+1})}{x_{j+1} - x_j}. \tag{81}$$

Note that the functions ρ_j form a partition of unity on $[a,b]$, i.e. $\sum_j \rho_j \equiv 1$ on $[a,b]$. Define

$$\begin{aligned}\Phi^J(x) &= \sum_{j=0}^{J} f(x_j)\rho_j(x) \\ &= \sum_{j=0}^{J}\left(\frac{f(x_{j+1})-f(x_j)}{x_{j+1}-x_j} - \frac{f(x_j)-f(x_{j-1})}{x_j-x_{j-1}}\right)\sigma(x-x_j) \\ &\quad + \frac{f(a)}{\left|f(a)\right|}\sigma(x-x_{-1}) + \frac{f(b)}{\left|f(b)\right|}\sigma(x-x_{J+1}).\end{aligned} \tag{82}$$

This rewriting reveals that Φ^J is a ReLU neural network with one hidden layer consisting of $J+3$ neurons and $3J+10$ weights and biases all of which are bounded by $\max\{1, |a|+\left|f(a)\right|, |b|+\left|f(b)\right|, 2\left\|f'\right\|_\infty\}$.

We proceed by bounding the approximation error made when approximating f by Φ^J. For every $j \in \{0,\dots,J\}$, let $\xi_j \in [x_{j-1}, x_j]$ be the point that satisfies $f'(\xi_j)(x_j - x_{j-1}) = f(x_j) - f(x_{j-1})$, the existence of which is guaranteed by the mean value theorem. Now denote by $(\Phi^J)'$ the weak derivative of Φ^J. One then has that

$$\begin{aligned}\left|f(x) - \Phi^J(x) - (f(y) - \Phi^J(y))\right| &= \left|\int_y^x (f'(t) - (\Phi^J)'(t))dt\right| \\ &\leq \left\|f' - (\Phi^J)'\right\|_\infty |x-y|.\end{aligned} \tag{83}$$

By Taylor's theorem, there exists for every $x \in [x_{j-1}, x_j]$ a $\zeta_x \in [x_{j-1}, x_j]$ such that $f'(x) = f'(\xi_j) + f''(\zeta_x)(x-\xi_j)$. Combining this with the fact that $(\Phi^J)'(x) = f'(\xi_j)$ for all $x \in (x_{j-1}, x_j)$, we get that

$$\left\|f' - (\Phi^J)'\right\|_\infty = \max_j \sup_{x\in(x_{j-1},x_j)} \left|f'(x) - f'(\xi_j)\right| \leq \max_j \left|x_j - x_{j-1}\right| \sup_{x\in(x_{j-1},x_j)} \left|f''(x)\right|. \tag{84}$$

This estimate holds for any grid in $\mathcal{X}_J$. Combining the previous inequalities, we obtain

$$\begin{aligned}\left\|f - \Phi^J\right\|_{\mathrm{Lip}([a,b];\mathbb{R})} &\leq \left\|f' - (\Phi^J)'\right\|_\infty \\ &\leq \inf_{(x_{-1},\dots,x_{J+1})\in\mathcal{X}_J} \max_j \left|x_j - x_{j-1}\right| \sup_{x\in(x_{j-1},x_j)} \left|f''(x)\right| \\ &\leq \frac{b-a}{J}\left\|f''\right\|_\infty,\end{aligned} \tag{85}$$

where we used that the error made using the optimal grid is bounded by the error on a uniform grid. □

A.2. **Proof of Lemma 2.6.**

Proof. In [36, Prop. 2 and 3], a parametrized class of ReLU neural networks f_m, $m \in \mathbb{N}$, that approximate $f : [0,1] \to [0,1] : x \mapsto x^2$ is constructed. In particular, it holds that $\mathcal{M}(f_m) = \mathcal{L}(f_m) = O(m)$ and $\mathcal{W}(f_m) = \mathcal{B}(f_m) = 4$. For $m \in \mathbb{N}$, f_m is piecewise affine and for $x \in I_k^m = [k/2^m, (k+1)/2^m]$, $0 \leq k \leq 2^m - 1$ given by

$$f_m(x) = \left(\frac{k}{2^m}\right)^2 + \left(x - \frac{k}{2^m}\right)\frac{2k+1}{2^m}. \tag{86}$$

Similar to the proof of Lemma 2.5 and using the above expression, we obtain that

$$\begin{aligned}\|f - f_m\|_{\mathrm{Lip}([0,1];\mathbb{R})} &\leq \max_k \left\|f' - f_m'\right\|_{L^\infty(I_k^m;\mathbb{R})} \\ &= \max_k \sup_{x\in I_k^m} \left|2x - \frac{2k+1}{2^m}\right| = \frac{1}{2^m}.\end{aligned} \tag{87}$$

In the spirit of [36, Prop. 3] and based on the observation that $xy = \frac{1}{4}((x+y)^2 - (x-y)^2)$ we then define $\widehat{\times}_m$ for $(x,y) \in [-M,M]\times[-N,N]$ by

$$\widehat{\times}_m(x,y) = \frac{(M+N)^2}{4}\left(f_m\left(\frac{|x+y|}{M+N}\right) - f_m\left(\frac{|x-y|}{M+N}\right)\right). \tag{88}$$

Since the map $x \mapsto |x \pm y|/(M+N)$ is $1/(M+N)$-Lipschitz, we get for all $y \in [-N,N]$ that

$$\left\|\times(\cdot,y) - \widehat{\times}_m(\cdot,y)\right\|_{\mathrm{Lip}([-M,M];\mathbb{R})} \leq 2\cdot\frac{(M+N)^2}{4}\cdot\frac{1}{M+N}\cdot\|f - f_m\|_{\mathrm{Lip}([0,1];\mathbb{R})} = \frac{M+N}{2^{m+1}}. \tag{89}$$

Finally, it follows directly that $\mathcal{M}(f_m) = O(m)$, $\mathcal{L}(f_m) = m + 1$, $\mathcal{W}(f_m) = 8$ and $\mathcal{B}(f_m) = O((M + N)^2)$. □

A.3. **Convergence rate of Lax-Friedrichs scheme.** We follow Example 3.14 from [18] to obtain an explicit error bound on the numerical approximation of a scalar conservation law based on the Lax-Friedrichs scheme. This is result is used in Theorem 3.2.

Lemma A.1. *Let u be the solution to the scalar conservation law (8) with a flux $f \in C^1(\mathbb{R})$ and $u_0 \in BV([a,b])$. Let $u_{\Delta t}$ be the numerical approximation to u based on the Lax-Friedrichs scheme with time step Δt. It holds that*

$$\left\|u_{\Delta t}(\cdot,T) - u(\cdot,T)\right\|_{L^1([a,b])} \leq 31\mathrm{TV}(u_0)\frac{T(1+\left\|f'\right\|_\infty)^2}{\sqrt{N}}. \tag{90}$$

Proof. We follow Example 3.14 from [18] and apply it specifically to the Lax-Friedrichs scheme. We redo the calculations of [18, p. 91] (using their notation) to get an explicit expression for the there mentioned constants. We denote define $F = \left\|f'\right\|_\infty$ and denote by $\lambda > 0$ the CFL number such that $\Delta t = \lambda \Delta x$. In this case we let $\lambda = F^{-1}$. First note that $p = 0$ and $p' = 1$ for Lax-Friedrichs. This then gives

$$\begin{aligned}
-\Lambda_{\epsilon,\epsilon_0}(u_{\Delta t}, u) &\leq 4\mathrm{TV}(u_0)F\sum_{n=0}^{N-1}\left(\frac{1}{2}(p+p'+1)^2\frac{(\Delta x)^2}{\epsilon} + \frac{(\Delta t)^2}{\epsilon_0} + \lambda\left(\frac{(\Delta x)^2}{\epsilon} + \frac{\Delta x \Delta t}{\epsilon_0}\right)\right) \\
&= 8\mathrm{TV}(u_0)FN(\Delta t)^2\left(\frac{1}{\lambda^2\epsilon} + \frac{1}{\epsilon_0} + \lambda\left(\frac{1}{\lambda^2\epsilon} + \frac{1}{\lambda\epsilon_0}\right)\right) \\
&= 8\mathrm{TV}(u_0)FT\Delta t\left(\frac{2}{\epsilon_0} + \frac{F^2+F}{\epsilon}\right) \\
&\leq 16\mathrm{TV}(u_0)(F+1)^3\left(\frac{1}{\epsilon_0} + \frac{1}{\epsilon}\right)T\Delta t.
\end{aligned} \tag{91}$$

Next we want to find an upper bound for

$$\nu_t(u_{\Delta t}, \epsilon) = \sup_{|\tau|\leq\epsilon}\left\|u_{\Delta t}(\cdot, t+\tau) - u_{\Delta t}(\cdot,t)\right\|_1. \tag{92}$$

We have that for the Lax-Friedrichs scheme

$$\begin{aligned}
&\Delta x\sum_j\left|U_j^{n+1} - U_j^n\right| \\
&\quad= \Delta x\sum_j\left|\frac{1}{2}(U_{j-1}^n + U_{j+1}^n) - \frac{\Delta t}{2\Delta x}(\widehat{f}(U_{j+1}^n) - \widehat{f}(U_{j-1}^n)) - U_j^n\right| \\
&\quad\leq \frac{\Delta x}{2}\sum_j\left(\left|U_{j-1}^n - U_j^n\right| + \left|U_{j+1}^n - U_j^n\right| + \lambda\|f\|_{\mathrm{Lip}}\left|U_{j+1}^n - U_{j-1}^n\right|\right) \\
&\quad\leq \frac{\Delta t}{2\lambda}(1+\lambda F)\sum_j\left(\left|U_{j-1}^n - U_j^n\right| + \left|U_{j+1}^n - U_j^n\right|\right) \\
&\quad\leq 2F\mathrm{TV}(u_0)\Delta t.
\end{aligned} \tag{93}$$

As a direct consequence, we obtain

$$\nu_t(u_{\Delta t}, \epsilon) \leq 2F\mathrm{TV}(u_0)(\epsilon + \Delta t). \tag{94}$$

Following [18], choose the initial approximation such that

$$\left\|u_{\Delta t}(\cdot,0) - u_0(\cdot)\right\|_1 \leq \Delta x\mathrm{TV}(u_0). \tag{95}$$

Now recalling Kuznetsov's lemma [18, Theorem 3.11],

$$\begin{aligned}
\left\|u_{\Delta t}(\cdot,T) - u(\cdot,T)\right\|_1 \leq &\left\|u_{\Delta t}(\cdot,0) - u_0(\cdot)\right\|_1 + \mathrm{TV}(u_0)(2\epsilon + \epsilon_0\|f\|_{\mathrm{Lip}}) \\
&+ \frac{1}{2}(\nu_T(u_{\Delta t},\epsilon_0) + \nu_0(u_{\Delta t},\epsilon_0)) - \Lambda_{\epsilon,\epsilon_0},
\end{aligned} \tag{96}$$

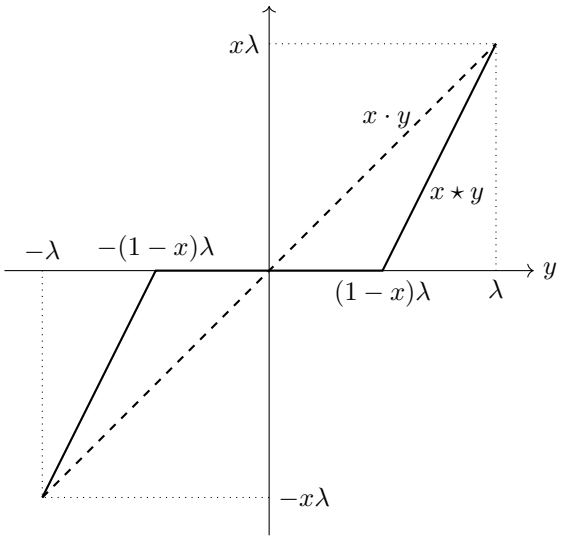


FIGURE 7. Plot of $x \star y$ and $x \cdot y$ for fixed $x \in [0,1]$ and $\lambda > 0$.

and all the previous inequalities, we get that

$$\begin{aligned}\left\|u_{\Delta t}(\cdot,T) - u(\cdot,T)\right\|_1 \leq & \mathrm{TV}(u_0)\big[(\Delta x + 2\epsilon + \epsilon_0\|f\|_{\mathrm{Lip}}) \\ & + 2F(\epsilon_0 + \Delta t) + 16(F+1)^3\left(\frac{1}{\epsilon_0} + \frac{1}{\epsilon}\right)T\Delta t\big] \\ \leq & \mathrm{TV}(u_0)\big[3F\Delta t + 3(1+F)(\epsilon+\epsilon_0) + 16(F+1)^3\left(\frac{1}{\epsilon_0} + \frac{1}{\epsilon}\right)T\Delta t\big].\end{aligned} \tag{97}$$

Minimizing with respect to ϵ, ϵ_0 then gives us

$$\begin{aligned}\left\|u_{\Delta t}(\cdot,T) - u(\cdot,T)\right\|_1 \leq & \mathrm{TV}(u_0)\left[3F\Delta t + 4\cdot 4(1+F)^2\sqrt{3T\Delta t}\right] \\ = & \mathrm{TV}(u_0)\left[\frac{3FT}{N} + 16T(1+F)^2\sqrt{\frac{3}{N}}\right] \\ \leq & 31\mathrm{TV}(u_0)\frac{T(1+F)^2}{\sqrt{N}}.\end{aligned} \tag{98}$$

□

A.4. **Replacing the multiplication operator with a ReLU network of fixed architecture.** The cornerstone of many constructive ReLU network approximations is the approximation of the multiplication operator as proposed by Yarotsky [36]. As per usual, the size of the network corresponding to this approximation depends on the desired approximation accuracy. We discuss an alternative method to approximate the calculation of a convex combination using a ReLU neural network of which the size does not depend on the approximation accuracy, as proposed in [13] based on Yarotsky's work. This method is used in Corollary 3.6.

Definition A.2. *For $\lambda > 0$, we denote by $\star$ the operation given by*

$$\star : [0,1] \times [-\lambda,\lambda] \to [-\lambda,\lambda] : (x,y) \mapsto x \star y := \sigma(y + \lambda x - \lambda) - \sigma(-y + \lambda x - \lambda), \tag{99}$$

where σ is the ReLU activation function (cf. Definition 2.1).

We compare $x \star y$ with $x \cdot y$ for fixed $x \in [0,1]$ and $\lambda > 0$ in Figure 7. In addition, the following properties hold.

Lemma A.3. *For $\lambda > 0$, the operation $\star$ satisfies the following properties:*

1. *For all $x \in \{0,1\}$ and $y \in [-\lambda,\lambda]$ it holds true that $x \star y = xy$.*
2. *For all $x \in [0,1]$ and $y \in [0,\lambda]$ we have $0 \leq x \star y \leq xy$.*
3. *For all $x \in [0,1]$ and $y \in [-\lambda,0]$ we have $xy \leq x \star y \leq 0$.*
4. *There exist $x \in [0,1]$ and $y_1, y_2 \in [-\lambda,\lambda]$ such that*

$$\min\{y_1,y_2\} \leq (1-x) \star y_1 + x \star y_2 \leq \max\{y_1,y_2\}$$

does not hold.

(5) *For all* $x \in [0,1]$ *and* $y_1, y_2 \in [-\lambda, \lambda]$ *it holds true that*

$$\min\{y_1, y_2\} \leq y_1 + x \star (y_2 - y_1) \leq \max\{y_1, y_2\}.$$

In conclusion, suppose we need to approximate $xy_1 + (1-x)y_2$ where $x \in [0,1]$ and $y_1, y_2 \in [-\lambda, \lambda]$ for some $\lambda > 0$. Lemma A.3 then assures us that $y_1 + x \star (y_2 - y_1) = xy_1 + (1-x)y_2$ for $x \in \{0,1\}$. Furthermore, if $0 < x < 1$ then $y_1 + x \star (y_2 - y_1)$ is still a convex combination of y_1 and y_2. For applications where this suffices, the operation $\star$ provides a more efficient alternative to Yarotsky's approximation of the multiplication operator.

Appendix B. Additional material for Section 4

We prove some auxiliary results on the measurability of the training error and the Lipschitz continuity of the training error with respect to the parameters of the ReLU neural network.

Lemma B.1. *Let* $T > 0$, $\theta \in \Theta$ *and let* $D \times Y \to \mathbb{R} : (x,y) \mapsto u(T,x,y)$ *and* $D \times Y \to \mathbb{R} : (x,y) \mapsto u_\theta(x,y)$ *measurable mappings. The the mapping* $Y \to \mathbb{R} : y \mapsto \left\|u(T,\cdot,y) - u_\theta(\cdot,y)\right\|_{L^1(D)}$ *is* $\Sigma/\mathcal{B}([0,\infty]$*-measurable.*

Proof. Let $T > 0$ and $\theta \in \Theta$. The result follows directly from the measurability of $u(T,\cdot,\cdot)$ and u_θ, the Fubini-Tonelli theorem and the fact that the composition of continuous and measurable functions is again measurable. □

Lemma B.2. *Let* $d, L, W, R \in \mathbb{N}$, $a, b, \alpha, \beta \in \mathbb{R}$ *with* $a < b$ *and* $\alpha + 1 \leq \beta$, $D = [a,b]$, $Y = [0,1]^d$ *and let* Θ *be the parameters of the class of* (α,β)*-clipped ReLU neural networks* $\Phi_\theta : D \to [\alpha,\beta]$ *(cf. Definitions 2.2 and 2.3), with* $\mathcal{L}(\Phi_\theta) = L$, $\mathcal{W}(\Phi_\theta) = W$ *and such that* $\mathcal{B}(\Phi_\theta) \leq R$. *Then it holds for all* $\theta, \eta \in \Theta$ *that*

$$\left\|\Phi_\theta - \Phi_\eta\right\|_{L^\infty} \leq LR^{L-1}(W+1)^L \|\theta - \eta\|_\infty. \tag{100}$$

Proof. This is [2, Cor. 2.37]. □

In the following, we use the notation from Lemma B.2 without explicitly defining it again.

Lemma B.3. *For* $y \in Y$, *it holds that*

$$\left\|u_{\theta_1}(\cdot,y) - u_{\theta_2}(\cdot,y)\right\|_{L^1(D)} \leq L(b-a)R^{L-1}(W+1)^L \|\theta_1 - \theta_2\|_\infty. \tag{101}$$

Proof. Because we identified the output of neural networks with functions (cf. Definition 2.4), the Lipschitz continuity of the associated solution maps with respect to the parameters does not follow immediately from Lemma B.2. Recall from Definition 2.4, that the outputs of the actual networks are $u_\theta(x_j, y)$. Hence, Lemma B.2 can be applied on the subnetworks $y \mapsto u_\theta(x_j, y)$. We then get that

$$\begin{aligned} \left\|u_{\theta_1}(\cdot,y) - u_{\theta_2}(\cdot,y)\right\|_{L^1(D)} &= \frac{b-a}{J} \sum_{j=1}^{J} \left|u_{\theta_1}(x_j,y) - u_{\theta_2}(x_j,y)\right| \\ &\leq (b-a) \cdot LR^{L-1}(W+1)^L \|\theta_1 - \theta_2\|_\infty \end{aligned} \tag{102}$$

□

Lemma B.4. *Let* $p \in \mathbb{N}$. *Define* $\mathcal{Y}_{\theta,i} = \mathcal{E}_T(\theta, \{y_i\})$. *If* $u(T,x,y) \in [\alpha,\beta]$ *and* $\sup_{\theta\in\Theta} u_\theta(x,y) \in [\alpha,\beta]$, *then it holds for* θ_1, θ_2 *that*

$$\left|(\mathcal{Y}_{\theta_1,i})^p - (\mathcal{Y}_{\theta_2,i})^p\right| \leq p(\beta-\alpha)^{p-1}(b-a)^p LR^{L-1}(W+1)^L \|\theta_1 - \theta_2\|_\infty. \tag{103}$$

Proof. Let $p \in \mathbb{N}$. For $x, y \in \mathbb{R}$ it holds that $x^p - y^p = (x-y)\sum_{k=1}^{p} x^{k-1}y^{p-k}$. From the assumptions of the lemma statement, it follows that $\left|\mathcal{Y}_{\theta,i}\right| \leq (\beta-\alpha)(b-a)$. Therefore,

$$\left|(\mathcal{Y}_{\theta_1,i})^p - (\mathcal{Y}_{\theta_2,i})^p\right| \leq p(\beta-\alpha)^{p-1}(b-a)^{p-1}\left|\mathcal{Y}_{\theta_1,i} - \mathcal{Y}_{\theta_2,i}\right|. \tag{104}$$

Using the reverse triangle inequality we obtain that

$$\begin{aligned} \left|\mathcal{Y}_{\theta_1,i} - \mathcal{Y}_{\theta_2,i}\right| &= \left|\left\|u(T,\cdot,y_i) - u_{\theta_1}(\cdot,y_i)\right\|_{L^1(D)} - \left\|u(T,\cdot,y_i) - u_{\theta_2}(\cdot,y_i)\right\|_{L^1(D)}\right| \\ &\leq \int_D \left|\left|u(T,x,y_i) - u_{\theta_1}(x,y_i)\right| - \left|u(T,x,y_i) - u_{\theta_2}(x,y_i)\right|\right| dx \\ &\leq \int_D \left|u_{\theta_1}(x,y_i) - u_{\theta_2}(x,y_i)\right| dx \\ &= \left\|u_{\theta_1}(\cdot,y_i) - u_{\theta_2}(\cdot,y_i)\right\|_{L^1(D)}. \end{aligned} \tag{105}$$

Now using Lemma B.3, we find that

$$\left\|u_{\theta_1}(\cdot, y_i) - u_{\theta_2}(\cdot, y_i)\right\|_{L^1(D)} \leq L(b-a)R^{L-1}(W+1)^L \|\theta_1 - \theta_2\|_\infty. \tag{106}$$

Combining all previous inequalities gives the wanted result. □

Lemma B.5. *Let $a, b \in \mathbb{R}$, $r > 0$ and $k \in \mathbb{N}$. Then it holds that*

$$\mathcal{C}_r^{([a,b]^k, \|\cdot\|_\infty)} \leq \left\lceil \frac{b-a}{r} \right\rceil^k. \tag{107}$$