ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Constructive Deep ReLU Neural Network Approximation

L. Herrmann and J. A. A. Opschoor and Ch. Schwab

Research Report No. 2021-04

January 2021
Latest revision: November 2021

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

Noname manuscript No.
(will be inserted by the editor)

Constructive Deep ReLU Neural Network Approximation

Lukas Herrmann · Joost A. A. Opschoor · Christoph Schwab

Received: date / Accepted: date

Abstract We propose an efficient, *deterministic algorithm for constructing exponentially convergent deep neural network (DNN) approximations* of multivariate, analytic maps $f : [-1,1]^K \to \mathbb{R}$. We address in particular networks with the rectified linear unit (ReLU) activation function. Similar results and proofs apply for many other popular activation functions. The algorithm is based on collocating f in deterministic families of grid points with small Lebesgue constants, and by a-priori (i.e., "offline") emulation of a spectral basis with DNNs to prescribed fidelity.

Assuming availability of N function values of a possibly corrupted, numerical approximation $\breve{f}$ of f in $[-1,1]^K$ and a bound on $\|f - \breve{f}\|_{L^\infty([-1,1]^K)}$, we provide an explicit, computational construction of a ReLU DNN which attains accuracy ε (depending on N and $\|f-\breve{f}\|_{L^\infty([-1,1]^K)}$) uniformly, with respect to the inputs. For analytic maps $f : [-1,1]^K \to \mathbb{R}$, we prove *exponential convergence of expression and generalization errors* of the constructed ReLU DNNs. Specifically, for every target accuracy $\varepsilon \in (0,1)$, there exists N depending also on f such that the error of the construction algorithm with N evaluations of $\breve{f}$ as input in the norm $L^\infty([-1,1]^K;\mathbb{R})$ is smaller than ε up to an additive data-corruption bound $\|f - \breve{f}\|_{L^\infty([-1,1]^K)}$ multiplied with a factor growing

ChS acknowledges stimulating exchanges with participants in the Isaac Newton Institute's Mathematics of Deep Learning (MDL) term, 1 July 2021 to 17 December 2021

L. Herrmann
Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Altenbergerstrasse 69, 4040 Linz, Austria
E-mail: lukas.herrmann@ricam.oeaw.ac.at

J.A.A. Opschoor, Ch. Schwab
Seminar for Applied Mathematics, ETH Zürich, Rämistrasse 101, CH-8092 Zürich, Switzerland
E-mail: joost.opschoor@sam.math.ethz.ch

E-mail: christoph.schwab@sam.math.ethz.ch

slowly with $1/\varepsilon$ and the number of non-zero DNN weights grows polylogarithmically with respect to $1/\varepsilon$. The algorithmic construction of the ReLU DNNs which will realize the approximations, is explicit and deterministic in terms of the function values of $\tilde{f}$ in tensorized Clenshaw–Curtis grids in $[-1,1]^K$. We illustrate the proposed methodology by a constructive algorithm for (offline) computations of posterior expectations in Bayesian PDE inversion.

Keywords Deep ReLU neural networks, exponential convergence, neural network construction, generalization error.

Mathematics Subject Classification (2010) 41A10 · 41A50 · 65D05 · 65D15

1 Introduction

Approximation by deep neural networks (DNNs) receives increasing attention recently. DNNs realize mappings by a combination of affine mappings and a coordinatewise applied (generally) non-linear function, which is referred to as the activation function. Most rigorous analyses study the approximation properties of certain DNN *architectures* and the existence of DNN weights which guarantee a small error, cf. [39,3,31].

A common theme among these references is that DNNs perform as well as the state-of-the-art numerical method in a variety of contexts.

In application the actual weights that represent the particular DNN are commonly computed by DNN training, effected with numerically minimizing a certain positive functional, the loss function. Training of DNNs is generally challenging: the occurring optimization problem is highly non-convex. This has for example been approached in computational uncertainty quantification and also in image processing, cf. [32,25,2,1].

On the other hand, there is mathematical evidence that this approach may not always be successful, cf. e.g. [5]. In the present paper, we propose to explicitly construct values of the weights of DNNs via a deterministic algorithm thereby circumventing the need of computationally costly optimization routines. Relevance of the present results is due to the fact that numerous applications in computational science and engineering aim at efficient numerical realization of input-output maps between suitable Banach spaces, such as for example data-to-solution maps for continuum models governed by partial differential equations. Equipping the input data space with suitable, affine-parametric representation systems such as (Riesz) bases or frames renders the maps of interest parametric. In many applications such maps are holomorphic, even for data spaces of inputs with possibly low spatial or temporal regularity, see for example [18]. In the present paper, following a general, algorithmic DNN construction and expression rate analysis, we develop one example consisting in data-to-prediction maps for Bayesian inverse problems of elliptic PDEs which should be contrasted with other, standard (i.e. numerical minimization of loss functions) approaches (e.g. [38,24]). Other applications include

shape-to-solution maps of differential or integral equations (e.g. [21,6,17] and the references there).

1.1 Previous Work

The idea of constructive DNN approximations via Chebyšev expansions and collocation in a tensor product Clenshaw-Curtis grid was already suggested in [27], in particular [27, Theorem 2.3] on the approximation of multivariate holomorphic functions by DNNs with smooth activations. There, robustness of the DNN approximation rates under noisy data was described as "expected", but not proved.

Constructive proofs for ReLU-based DNN emulation of polynomials were firstly seen in [23,39]. Mainly in [39] a construction for DNNs with rectified linear unit (ReLU) activation that approximate the product of two scalars was found. The convergence of these constructions is exponential with respect to the size of the DNNs meaning the number of nonzero weights. Subsequent works used DNNs and polynomial approximations for smooth (or analytic) functions and established the existence of DNNs with exponential convergence, in the L^∞- and some in the stronger $W^{1,\infty}$-norm: For example, [35] provided $W^{1,\infty}$-error bounds for the product network in [39], exponential convergence in $W^{1,\infty}$ for univariate analytic and Gevrey regular functions was shown in [29], [12] provided exponentially convergent DNN approximations of holomorphic maps on $[-1,1]^d$, with respect to the L^∞-norm. A more efficient approximation with respect to the $W^{1,\infty}$-norm and under weaker smoothness assumptions was given in [30]. The proofs in these references are constructive, in principle, and algorithmic realizations could be based on the ReLU reapproximation of polynomials as linear combinations of monomials. It is well-known that such representations may have poor stability in finite-precision arithmetic (in particular, in the context of quantized DNN weights) due to exponential w.r. to the polynomial degree coefficient growth in monomial expansions. Furthermore, these constructions use e.g. Taylor, Legendre or Chebyšev coefficients of analytic functions, which cannot be determined exactly based on a finite number of function evaluations.

An alternative approach to exponential DNN approximation of multivariate holomorphic maps was used in [8], which used ReLU DNN approximations of tensor products of univariate polynomials with real roots by using approximate ReLU DNN multiplications of their linear factors. This did not lead to better asymptotic approximation rates than for the previously mentioned monomial-based polynomial approximations. The DNNs inherit the well-known stability and conditioning issues from monomial representations of interpolation polynomials.

Constructive approximation of multivariate holomorphic parametric maps by tensorized Chebyšev polynomials with exponential convergence has been studied in the context of option pricing in [15]. The approximation of multivariate maps by DNNs based on Chebyšev polynomials has been considered

in [36]. There, Chebyšev polynomials represented exactly by so called *RePU* DNNs serve as starting value for computing DNN weights in black box optimization routines. The presented DNNs are based on a well-known identity satisfied by the product of two Chebyšev polynomials. The initial error prior to DNN optimization was not analyzed theoretically in [36].

Recently, collocation-based DNN constructions based on spline interpolation were studied in [11]. For the approximation of functions of mixed smoothness, algebraic convergence rates were obtained which are free from the curse of dimensionality.

1.2 Contributions

The principal contributions of this work are threefold: we propose and implement a constructive numerical approximation algorithm for DNN expression of maps $f : [-1,1]^K \to \mathbb{R}$. Specifically, we propose and analyze an algorithm to build by explicit construction a DNN surrogate to the map f with a) rigorous, sup-norm generalization error bounds and with b) good stability properties in finite precision arithmetic, and c) with low complexity.

The general idea of our approach is as follows: in the parameter "box" $[-1,1]^K$ of finite dimension K, we wish to construct DNN emulations of given maps $f : [-1,1]^K \to \mathbb{R}$, having at hand a finite set of samples $\{f(\boldsymbol{x}) : \boldsymbol{x} \in \Gamma\}$ for a grid (i.e. a finite subset with additional structure, to be specified) $\Gamma \subset [-1,1]^K$. Importantly, *we assume that the map f to be emulated can be numerically queried at $\boldsymbol{x} \in \Gamma$ at unit cost.* We also allow a sampling error, resulting in noisy evaluations $\breve{f}(\boldsymbol{x})$ for $\boldsymbol{x} \in \Gamma$, with a known bound on the noise $\max_{\boldsymbol{x}\in\Gamma} |f(\boldsymbol{x}) - \breve{f}(\boldsymbol{x})|$.

The construction of the DNN surrogate of f proceeds in two steps. First, for some finite index set $\Lambda \subset \mathbb{N}_0^K$ with associated polynomial space $\mathbb{P}_\Lambda = \operatorname{span}\{\boldsymbol{x}^{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \Lambda\}$ and corresponding data-sampling grid $\Gamma_\Lambda \subset [-1,1]^K$, a polynomial interpolant $I_\Lambda[f]$ of f on $[-1,1]^K$ based on samples in the grid Γ_Λ is constructed, where we require for Γ_Λ *unisolvency* for polynomial interpolation in $\mathbb{P}_\Lambda$. We consider the (unique) polynomial interpolant $f_\Lambda := I_\Lambda[f] \in \mathbb{P}_\Lambda$ which satisfies $f(\boldsymbol{x}) = f_\Lambda(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Gamma_\Lambda$. Unisolvency of Γ_Λ on $\mathbb{P}_\Lambda$ implies that on noisy evaluations $\breve{f}$ of f in Γ_Λ, there is a uniquely defined interpolant $\breve{f}_\Lambda := I_\Lambda[\breve{f}]$.

For ease of exposition, we consider here only *isotropic tensor product interpolation.* I.e., for a prescribed polynomial degree $n \in \mathbb{N}$, $\Lambda := \{0:n\}^K = \{\boldsymbol{\nu} \in \mathbb{N}_0^K : \|\boldsymbol{\nu}\|_\infty \le n\}$ so that $\dim(\mathbb{P}_\Lambda) = (n+1)^K$. We hasten to add, however, that also more general polynomial spaces could be considered which are based on, e.g., anisotropic, total degree or more general sparse grids $\Gamma_\Lambda \subset [-1,1]^K$ which will be addressed elsewhere.

The second step of our DNN construction is *DNN emulation of a basis of* $\mathbb{P}_\Lambda$: assuming $\mathbb{P}_\Lambda = \operatorname{span}\{p_{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \Lambda\}$, we shall construct DNN surrogates $\tilde{p}_{\boldsymbol{\nu}}$ of $p_{\boldsymbol{\nu}}$. Being independent of $f|_\Gamma$, these can be constructed offline, and we develop concrete constructions here. With the DNN surrogates $\tilde{p}_{\boldsymbol{\nu}}$ at hand,

the constructed DNN emulating f is

$$\tilde{\breve{f}}(\boldsymbol{x}) := \sum_{\boldsymbol{\nu}\in\Lambda} \breve{f}_{\boldsymbol{\nu}} \tilde{p}_{\boldsymbol{\nu}}(\boldsymbol{x}) \,.$$

Evidently, $\tilde{\breve{f}}$ is a DNN, being a linear combination of (pre-computed) DNNs $\tilde{p}_{\boldsymbol{\nu}}$. We also see that the coefficients $\breve{f}_{\boldsymbol{\nu}}$, computed from the data $(\breve{f}(\boldsymbol{x}))_{\boldsymbol{x}\in\Gamma_\Lambda}$, can be incorporated simply in the output layer of $\tilde{\breve{f}}$.

Efficiency, stability and accuracy of this process depend on the following key properties to be addressed here: (i) choice of the sampling designs $\Gamma_\Lambda \subset [-1,1]^K$, (ii) stability of the basis $p_{\boldsymbol{\nu}}$ of $\mathbb{P}_\Lambda$, (iii) accuracy and complexity of the emulations $\tilde{p}_{\boldsymbol{\nu}}$.

We achieve (i) and (ii) by adapting ideas from spectral collocation. Numerical stability of the polynomial interpolation process in finite precision arithmetic is required in order to preclude catastrophic amplification of, e.g., numerical errors in the function value queries $f(\boldsymbol{x})$, for $\boldsymbol{x} \in \Gamma_\Lambda$. It is well known to depend on two related issues: the choice of the sampling grid Γ_Λ and of the basis $\{p_{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \Lambda\}$ spanning $\mathbb{P}_\Lambda$. Favorable numerical conditioning of the interpolation operator $I_\Lambda : C^0([-1,1]^K) \to \mathbb{P}_\Lambda : f \mapsto f_\Lambda$ is known to be governed by the Lebesgue constant of Γ_Λ.

For our DNN approximations, it turns out that tensor product Chebyšev polynomials are a more favorable basis than e.g. Lagrange polynomials. We have

$$f_\Lambda(\boldsymbol{x}) = \sum_{\boldsymbol{\nu}\in\Lambda} f_{\boldsymbol{\nu}} T_{\boldsymbol{\nu}}(\boldsymbol{x}), \quad \breve{f}_\Lambda(\boldsymbol{x}) = \sum_{\boldsymbol{\nu}\in\Lambda} \breve{f}_{\boldsymbol{\nu}} T_{\boldsymbol{\nu}}(\boldsymbol{x}), \quad \boldsymbol{x} \in [-1,1]^K \,, \tag{1.1}$$

where each coefficient $f_{\boldsymbol{\nu}}$ (resp. $\breve{f}_{\boldsymbol{\nu}}$) can be computed from the function values $(f(\boldsymbol{x}))_{\boldsymbol{x}\in\Gamma_\Lambda}$ (resp $(\breve{f}(\boldsymbol{x}))_{\boldsymbol{x}\in\Gamma_\Lambda}$). When Γ_Λ is the tensor product Clenshaw–Curtis grid, these coefficients can be computed efficiently using the inverse fast Fourier transform (we recall the relevant details of polynomial interpolation in Section 2).

We consider here in particular DNNs $\tilde{T}_{\boldsymbol{\nu}}$ approximating $T_{\boldsymbol{\nu}}$ with ReLU activation, and we propose a DNN architecture of low complexity by exploiting certain algebraic properties of univariate Chebyšev polynomials which are tensorized to build the $T_{\boldsymbol{\nu}}$. Based on an observation in [36], we obtain smaller architectures and better stability than with other polynomial bases (e.g. in [39,30] either monomial bases were considered resulting in small DNNs with large Lebesgue constants or Legendre polynomials were considered which have better stability, but require larger DNNs for their accurate emulation). We finally obtain a DNN approximation $\tilde{\breve{f}}$ of f whose DNN weights are computable explicitly:

$$\boldsymbol{x} \mapsto \tilde{\breve{f}}_\Lambda(\boldsymbol{x}) := \sum_{\boldsymbol{\nu}\in\Lambda} \breve{f}_{\boldsymbol{\nu}} \tilde{T}_{\boldsymbol{\nu}}(\boldsymbol{x}), \quad \boldsymbol{x} \in [-1,1]^K \,. \tag{1.2}$$

The coefficients $\breve{f}_{\boldsymbol{\nu}}$ are computed from the function values $(\breve{f}(\boldsymbol{x}))_{\boldsymbol{x}\in\Gamma_\Lambda}$, which serve here as *synthetic training data*. A so-called training routine, for example the widely used stochastic gradient descent method, is not necessary. The meticulous choice of training data points $\boldsymbol{x} \in \Gamma_\Lambda$ is crucial to obtain a small Lebesgue constant and thereby good performance of our algorithm. This is in contrast to randomly or quasi-randomly chosen training points, which have been used in connection with the stochastic gradient method, see for example [25].

The DNN expression and generalization error analysis proceeds according to

$$\|f - \breve{\breve{f}}_\Lambda\| \leq \|f - f_\Lambda\| + \|f_\Lambda - \breve{f}_\Lambda\| + \|\breve{f}_\Lambda - \breve{\breve{f}}_\Lambda\|. \tag{1.3}$$

The first term is an error of polynomial interpolation, the second term is due to the error in the numerical approximation $\breve{f}$ of the true response f used in the DNN construction, and the third error term is, essentially, the DNN emulation error of the polynomial basis elements $T_{\boldsymbol{\nu}}$.

The choice of the norm $\|\circ\|$ will be either the L^∞- or also the $W^{1,\infty}$-norm on the (scaled) input data domain $[-1,1]^K$. This means we provide upper bounds on what is often referred to as *generalization error* of the DNN and its first order sensitivities.

The decay of the first error in (1.3) is determined by the error of best polynomial approximation in $\mathbb{P}_\Lambda$ and the Lebesgue constant of Γ_Λ, with respect to the norm $\|\circ\|$, which we denote by $\|I_\Lambda\|$. We recall that

$$\sup_{f\in B} \|f - f_\Lambda\| \leq (1 + \|I_\Lambda\|) \sup_{f\in B} \inf_{p\in\mathbb{P}_\Lambda} \|f - p\|. \tag{1.4}$$

That is, for *every* function class $B \subset C^0([-1,1]^K)$ the interpolation operator I_Λ obtains, uniformly over all $f \in B$, the convergence rate of best polynomial approximation in $\mathbb{P}_\Lambda$, up to a factor $(1+\|I_\Lambda\|)$. In addition, the second term in (1.3) is bounded by $\|I_\Lambda\|\|f - \breve{f}\|$ for all $f, \breve{f} \in C^0([-1,1]^K)$. The third term in (1.3) is the error of approximating tensorized Chebyšev polynomials by ReLU DNNs. It is bounded in Proposition 3.3 for arbitrary index sets $\Lambda \subset \mathbb{N}_0^K$, which is a result of independent interest.

For our choice of index sets $\Lambda = \{0,\dots,n\}^K$, $n \in \mathbb{N}$, and Γ_Λ the tensorized Clenshaw–Curtis grids, the bound $\|I_\Lambda\| \leq C(1+\log(n))^K$ for the L^∞-Lebesgue constant implies that I_Λ obtains in L^∞, in terms of the number of degrees of freedom $|\Lambda|$ and up to logarithmic factors in n, the rate of best polynomial approximation of f. The size of the ReLU DNN approximations is bounded by $C|\Lambda|$ multiplied by factors logarithmic in n and the reciprocal of the desired accuracy, and algebraic in K. This shows exponential convergence of ReLU DNN approximations of holomorphic functions f, as analyzed in detail in this paper. By the same arguments, ReLU DNN approximations of $f \in C^0([-1,1]^K)$ satisfying $\inf_{p\in\mathbb{P}_\Lambda}\|f-p\| \sim Cn^{-\theta}$ for $\theta > 0$, obtain in L^∞ the rate θ/K of best polynomial approximation in terms of the network size, up to logarithmic factors in n stemming from the Lebesgue constant and

the fact that the size of a ReLU DNN approximating a multiplication of two numbers depends logarithmically on the reciprocal of the desired accuracy.

For many other activation functions than ReLU, smallness of the third term in (1.3) can be guaranteed for DNNs whose size is independent of the accuracy. We comment on such networks, obtaining the same convergence rates as polynomial interpolation, in Section 3.4.

1.3 Notation

We will denote vectors and multiindices by bold characters. We denote $\mathbb{N} = \{1, 2, ...\}$ and $\mathbb{N}_0 = \{0, 1, 2, ...\}$. For $k \in \mathbb{N}_0$ and a subset $S \subset \mathbb{N}_0$, we define $\mathbb{1}_S(k) := 1$ if $k \in S$, and $\mathbb{1}_S(k) := 0$ otherwise. For $K \in \mathbb{N}$ and $\boldsymbol{k} \in \mathbb{N}_0^K$, we define $\mathbb{1}_S(\boldsymbol{k}) := \sum_{j=1}^K \mathbb{1}_S(k_j)$ and denote by $|\boldsymbol{k}|_0 := \mathbb{1}_{\mathbb{N}}(\boldsymbol{k})$ the number of nonzero components of $\boldsymbol{k}$. For finite index sets $\Lambda \subset \mathbb{N}_0^K$, we denote the number of elements by $|\Lambda|$ and the maximum coordinatewise degree by $m_\infty(\Lambda) := \max_{\boldsymbol{k}\in\Lambda} \|\boldsymbol{k}\|_{\ell^\infty}$.

We denote by T_k, $k \in \mathbb{N}_0$, the univariate Chebyšev polynomials of the first kind, normalized such that $T_k(1) = 1$ for all $k \in \mathbb{N}_0$. For $K \in \mathbb{N}$ and $\boldsymbol{k} = (k_j)_{j=1}^K \in \mathbb{N}_0^K$, we denote tensor product Chebyšev polynomials by $T_{\boldsymbol{k}}(\boldsymbol{x}) := \prod_{j=1}^K T_{k_j}(x_j)$, for $\boldsymbol{x} = (x_j)_{j=1}^K \in [-1, 1]^K$. We write $\cos(\boldsymbol{\theta}) := (\cos(\theta_1), \dots, \cos(\theta_K))$ for vectors $\boldsymbol{\theta} = (\theta_1, \dots, \theta_K) \in \mathbb{R}^K$ and $\boldsymbol{x}^{\boldsymbol{\nu}} := \prod_{j=1}^K x_j^{\nu_j}$ for $\boldsymbol{x} \in \mathbb{R}^K$ and $\boldsymbol{\nu} \in \mathbb{N}_0^K$, with the convention $0^0 = 1$.

We introduce the following notation for circles in the complex plane: For all $r > 0$ we define $\Gamma_r := \{z \in \mathbb{C} : |z| = r\}$. For all $r \geq 1$, the image of the annulus $\{z \in \mathbb{C} : 1 \leq |z| \leq \rho\}$ under the map $z \mapsto \frac{z+z^{-1}}{2}$ is a closed *Bernstein ellipse*, denoted by $\mathcal{E}_\rho := \left\{\frac{z+z^{-1}}{2} \in \mathbb{C} : 1 \leq |z| \leq \rho\right\}$. For $p \geq 0$, the space of polynomials of degree at most p is denoted by $\mathbb{P}_p$. The space of polynomials in $K \in \mathbb{N}$ variables of coordinatewise degree at most p is denoted by $\mathbb{Q}_p := \operatorname{span}\{\boldsymbol{x}^{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \mathbb{N}_0^K \text{ and } \|\boldsymbol{\nu}\|_{\ell^\infty} \leq p\}$.

Error estimates will be expressed in terms of the $W^{1,\infty}$-Sobolev norm, which is defined, for an open and bounded domain $\Omega \subset \mathbb{R}^K$, as $\|u\|_{W^{1,\infty}(\Omega)} = \max\{\|u\|_{L^\infty(\Omega)}, \max_{i=1}^K \|\frac{\partial}{\partial x_i} u\|_{L^\infty(\Omega)}\}$, $\frac{\partial}{\partial x_i}$ denoting weak derivatives. For integer $s \geq 1$, the $W^{s,\infty}$-norm is defined analogously, taking the maximum over all weak derivatives of order at most s.

Throughout, a superscript tilde (e.g. $\tilde{f}$) shall denote a DNN. With a superscript breve (e.g. $\breve{f}$) we shall denote a corrupted quantity. Typically, $\breve{f}$ denotes a *numerical approximation* of the map f, due to some measurement or, in our case, due to some discretization error in approximating the map f.

1.4 Outline

In Section 2, we first recapitulate classical results on constructive polynomial approximation of multivariate, holomorphic functions on $[-1, 1]^K$. In Section

3, we recapitulate ReLU DNN approximation rates of multivariate maps $f : [-1,1]^K \to \mathbb{R}$ from [30]. We also develop a constructive DNN approximation. It is based on standard spectral collocation approximation of f in a tensorized Clenshaw–Curtis grid, and on ReLU DNN emulation of tensorized Chebyšev polynomials. In Section 3.4, we comment on generalizations of our results to DNNs with activation functions other than ReLU. In Section 4, a construction algorithm is proposed and analyzed that converges exponentially for analytic functions. An application of the presented algorithm to a Bayesian inverse problem is discussed in Section 5 and numerical experiments confirming the theory are provided in Section 6.

2 Polynomial Approximation of Multivariate Holomorphic Functions

It is classical that univariate functions $f : [-1,1] \to \mathbb{R}$ which are real-analytic in $[-1,1]$ admit sequences of *polynomial approximations* $\{f_p\}_{p\geq 0}$ with $f_p \in \mathbb{P}_p$ which converge at an exponential rate. These univariate polynomial approximations can be tensorized to produce exponentially convergent, tensorized polynomial approximations to multivariate maps $f : [-1,1]^K \to \mathbb{R}$. This argument was used in [30] to infer existence of tensorized truncated Legendre expansions of co-ordinatewise polynomial degree at most $p \in \mathbb{N}$ of f which could, in principle, serve as building block for the corresponding multivariate ReLU DNNs which emulate the map f.

In this section, we present an alternative proof of such multivariate polynomial approximation of holomorphic maps $f : [-1,1]^K \to \mathbb{R}$, which admit a holomorphic complex extension to the isotropic Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}} = \mathcal{E}_\rho^K$ with polyradius $\boldsymbol{\rho} = (\rho, \ldots, \rho) \in \mathbb{R}^K$ for some $\rho \in (1, \infty)$. The interpolation results are basically known, being based on tensorized Chebyšev polynomials. We detail them here, as they are the basis for the ensuing ReLU DNN approximations. Bounds on Lebesgue constants of Chebyšev points in $[-1,1]$ are essential in quantifying numerical stability of the interpolation and, more importantly, of the DNN approximation process.

2.1 Chebyšev Expansion

We first recall the tensor product Chebyšev expansion, which is obtained by inductively taking the Chebyšev expansion with respect to each of the K coordinates. The following results are classical, we refer to [37, Theorem 3.1] for the arguments in the univariate case, which we apply K times. Denoting the Chebyšev measure on $[-1,1]$ by λ, which has Lebesgue density $(1-x^2)^{-1/2}$

for $x \in (-1,1)$, it holds for all $\boldsymbol{y} = (y_1, \dots, y_K) \in [-1,1]^K$

$$
\begin{aligned}
f(y_1,\dots,y_K) &= \sum_{k_1=0}^{\infty} 2^{\mathbb{1}_{\mathbb{N}}(k_1)} \pi^{-1} T_{k_1}(y_1) \int_{-1}^{1} f(x_1, y_2, \dots, y_K) T_{k_1}(x_1)\,\mathrm{d}\lambda(x_1) \\
&= \sum_{k_1=0}^{\infty} 2^{\mathbb{1}_{\mathbb{N}}(k_1)} \pi^{-1} T_{k_1}(y_1) \int_{-1}^{1} \cdots \\
&\qquad \sum_{k_K=0}^{\infty} 2^{\mathbb{1}_{\mathbb{N}}(k_K)} \pi^{-1} T_{k_K}(y_K) \int_{-1}^{1} \\
&\qquad\qquad f(x_1,\dots,x_K) T_{k_K}(x_K)\,\mathrm{d}\lambda(x_K) \cdots T_{k_1}(x_1)\,\mathrm{d}\lambda(x_1) \\
&= \sum_{\boldsymbol{k} \in \mathbb{N}_0^K} T_{\boldsymbol{k}}(\boldsymbol{y}) 2^{|\boldsymbol{k}|_0} \pi^{-K} \int_{-1}^{1} \cdots \int_{-1}^{1} f(\boldsymbol{x}) T_{\boldsymbol{k}}(\boldsymbol{x})\,\mathrm{d}\lambda(x_1) \cdots \mathrm{d}\lambda(x_K) \\
&=: \sum_{\boldsymbol{k} \in \mathbb{N}_0^K} T_{\boldsymbol{k}}(\boldsymbol{y}) f_{\boldsymbol{k}}. \qquad (2.1)
\end{aligned}
$$

In the third step, interchanging summation and integration is justified by the dominated convergence theorem. It implies that for all $j = 2, \dots, K$ and $\ell = 1, \dots, j-1$, all $(k_\ell, \dots, k_{j-1}) \in \mathbb{N}_0^{j-\ell}$, all $\boldsymbol{y} = (y_1, \dots, y_K) \in [-1,1]^K$ and all $(x_1, \dots, x_{\ell-1}) \in [-1,1]^{\ell-1}$

$$
\begin{aligned}
&\int_{-1}^{1} \sum_{k_j=0}^{\infty} \pi^{-(j-\ell)} \prod_{i=\ell+1}^{j} \left(2^{\mathbb{1}_{\mathbb{N}}(k_i)} T_{k_i}(y_i)\right) \\
&\left[\int_{[-1,1]^{j-\ell}} f(x_1,\dots,x_j,y_{j+1},\dots,y_K) T_{k_j}(x_j)\,\mathrm{d}\lambda(x_j) \cdots T_{k_{\ell+1}}(x_{\ell+1})\,\mathrm{d}\lambda(x_{\ell+1})\right] \\
&\qquad T_{k_\ell}(x_\ell)\,\mathrm{d}\lambda(x_\ell) \\
&= \lim_{N\to\infty} \int_{-1}^{1} \sum_{k_j=0}^{N} \pi^{-(j-\ell)} \prod_{i=\ell+1}^{j} \left(2^{\mathbb{1}_{\mathbb{N}}(k_i)} T_{k_i}(y_i)\right) \\
&\left[\int_{[-1,1]^{j-\ell}} f(x_1,\dots,x_j,y_{j+1},\dots,y_K) T_{k_j}(x_j)\,\mathrm{d}\lambda(x_j) \cdots T_{k_{\ell+1}}(x_{\ell+1})\,\mathrm{d}\lambda(x_{\ell+1})\right] \\
&\qquad T_{k_\ell}(x_\ell)\,\mathrm{d}\lambda(x_\ell).
\end{aligned}
$$

Use of the dominated convergence theorem is justified because for all $N \in \mathbb{N}_0$, Lemma 2.1 below can be applied to

$$
g : [-1,1]^{j-\ell} \to \mathbb{R} : (x_{\ell+1}, \dots, x_j) \mapsto f(x_1, \dots, x_j, y_{j+1}, \dots, y_K),
$$

which implies that

$$
\begin{aligned}
&\Bigg| \sum_{k_j=0}^{N} \pi^{-(j-\ell)} \prod_{i=\ell+1}^{j} \left(2^{\mathbb{1}_{\mathbb{N}}(k_i)} T_{k_i}(y_i)\right) T_{k_\ell}(x_\ell) \\
&\left[\int_{[-1,1]^{j-\ell}} f(x_1,\dots,x_j,y_{j+1},\dots,y_K) T_{k_j}(x_j)\,\mathrm{d}\lambda(x_j)\cdots T_{k_{\ell+1}}(x_{\ell+1})\,\mathrm{d}\lambda(x_{\ell+1})\right]\Bigg| \\
&\leq \sum_{k_j=0}^{N} \pi^{-(j-\ell)} \prod_{i=\ell+1}^{j} \left(2^{\mathbb{1}_{\mathbb{N}}(k_i)} |T_{k_i}(y_i)|\right) |T_{k_\ell}(x_\ell)| \\
&\left|\int_{[-1,1]^{j-\ell}} f(x_1,\dots,x_j,y_{j+1},\dots,y_K) T_{k_j}(x_j)\,\mathrm{d}\lambda(x_j)\cdots T_{k_{\ell+1}}(x_{\ell+1})\,\mathrm{d}\lambda(x_{\ell+1})\right| \\
&\leq \sum_{k_j=0}^{N} \pi^{-(j-\ell)} \prod_{i=\ell+1}^{j} 2^{\mathbb{1}_{\mathbb{N}}(k_i)} \\
&\left|\int_{[-1,1]^{j-\ell}} f(x_1,\dots,x_j,y_{j+1},\dots,y_K) T_{k_j}(x_j)\,\mathrm{d}\lambda(x_j)\cdots T_{k_{\ell+1}}(x_{\ell+1})\,\mathrm{d}\lambda(x_{\ell+1})\right| \\
&= \sum_{k_j=0}^{N} |g_{(k_{\ell+1},\dots,k_j)}| \leq \sum_{k_j=0}^{N} \prod_{i=\ell+1}^{j} 2^{\mathbb{1}_{\mathbb{N}}(k_i)} \max_{\boldsymbol{z}\in\mathcal{E}_\rho^{j-\ell}} |g(\boldsymbol{z})| \boldsymbol{\rho}^{-(k_{\ell+1},\dots,k_j)} \\
&< 2^{j-\ell} \tfrac{\rho}{\rho-1} \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|,
\end{aligned}
$$

where $g_{(k_{\ell+1},\dots,k_j)}$ denotes a Chebyšev coefficient of g, analogous to $f_{\boldsymbol{k}}$ defined in Equation (2.1).

We next estimate the size of the coefficients, which we later need to bound the DNN error.

Lemma 2.1 *Let $K \in \mathbb{N}$, and let $f : [-1,1]^K \to \mathbb{R}$ be a map which admits a holomorphic complex extension to the isotropic Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}} \subset \mathbb{C}^K$ with $\boldsymbol{\rho} = (\rho,\dots,\rho) \in (1,\infty)^K$ for some $\rho > 1$. Then, for every $\boldsymbol{k} \in \mathbb{N}_0^K$*

$$
|f_{\boldsymbol{k}}| \leq 2^{|\boldsymbol{k}|_0} \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| \boldsymbol{\rho}^{-\boldsymbol{k}}, \qquad \sum_{\boldsymbol{k}\in\mathbb{N}_0^K} |f_{\boldsymbol{k}}| \leq \left(\tfrac{2\rho}{\rho-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|. \tag{2.2}
$$

Proof The Chebyšev coefficients of f satisfy (cf. e.g. [37, Equations (3.12) – (3.14)] and [33, Theorem 3.8])

$$
\begin{aligned}
f_{\boldsymbol{k}} &= 2^{|\boldsymbol{k}|_0} \pi^{-K} \int_{-1}^{1} \cdots \int_{-1}^{1} f(\boldsymbol{x}) T_{\boldsymbol{k}}(\boldsymbol{x})\,\mathrm{d}\lambda(x_1)\cdots\mathrm{d}\lambda(x_K) \qquad (2.3)\\
&= 2^{|\boldsymbol{k}|_0} (2\pi)^{-K} \int_{-\pi}^{\pi} \cdots \int_{-\pi}^{\pi} f(\cos(\boldsymbol{\theta})) \cos(k_1\theta_1)\,\mathrm{d}\theta_1 \cdots \cos(k_K\theta_K)\,\mathrm{d}\theta_K \\
&= 2^{|\boldsymbol{k}|_0} (2\pi i)^{-K} \int_{\Gamma_1} \cdots \int_{\Gamma_1}
\end{aligned}
$$

$$f\left(\tfrac{z_1+z_1^{-1}}{2},\dots,\tfrac{z_K+z_K^{-1}}{2}\right)\left(\tfrac{z_1^{k_1}+z_1^{-k_1}}{2}\right)\tfrac{\mathrm{d}z_1}{z_1}\cdots\left(\tfrac{z_K^{k_K}+z_K^{-k_K}}{2}\right)\tfrac{\mathrm{d}z_K}{z_K}$$
$$=2^{|\boldsymbol{k}|_0}(2\pi i)^{-K}\int_{\Gamma_1}\cdots\int_{\Gamma_1} f\left(\tfrac{z_1+z_1^{-1}}{2},\dots,\tfrac{z_K+z_K^{-1}}{2}\right) z_1^{-k_1}\tfrac{\mathrm{d}z_1}{z_1}\cdots z_K^{-k_K}\tfrac{\mathrm{d}z_K}{z_K},$$

where the last step follows by invariance of $\frac{z+z^{-1}}{2}$ under the transformation $z\mapsto z^{-1}$, for $z\in\mathbb{C}\backslash\{0\}$. By holomorphy of f, we can change the curve of integration from Γ_1 to Γ_ρ, so that

$$\begin{aligned}|f_{\boldsymbol{k}}| &= \left|2^{|\boldsymbol{k}|_0}(2\pi i)^{-K}\int_{\Gamma_\rho}\cdots\int_{\Gamma_\rho} f\left(\tfrac{z_1+z_1^{-1}}{2},\dots,\tfrac{z_K+z_K^{-1}}{2}\right)\boldsymbol{z}^{-\boldsymbol{k}}\tfrac{\mathrm{d}z_1}{z_1}\cdots\tfrac{\mathrm{d}z_K}{z_K}\right|\\ &\le 2^{|\boldsymbol{k}|_0}\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})|(2\pi)^{-K}\left|\int_{\Gamma_\rho}\cdots\int_{\Gamma_\rho}\boldsymbol{z}^{-\boldsymbol{k}}\tfrac{\mathrm{d}z_1}{z_1}\cdots\tfrac{\mathrm{d}z_K}{z_K}\right|\\ &\le 2^{|\boldsymbol{k}|_0}\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})|\boldsymbol{\rho}^{-\boldsymbol{k}},\end{aligned}$$
$$\sum_{\boldsymbol{k}\in\mathbb{N}_0^K}|f_{\boldsymbol{k}}|\le 2^K\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})|\left(\tfrac{\rho}{\rho-1}\right)^K=\left(\tfrac{2\rho}{\rho-1}\right)^K\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})|.$$

2.2 Chebyšev Interpolation

Because in general the Chebyšev coefficients of f are not known explicitly, we next approximate the coefficients and thus consider polynomial interpolation of f: We approximate f by a polynomial whose coefficients with respect to the Chebyšev basis only depend on function values of f in the tensor product Clenshaw–Curtis grid. For $n\in\mathbb{N}$, we consider the approximation $\hat{f}_{\boldsymbol{k};n}$ of the Chebyšev coefficients, for $\boldsymbol{k}\in\{0,\dots,n\}^K$, obtained by applying an $(n+1)^K$ point tensor product quadrature in the Clenshaw–Curtis points $\boldsymbol{x}_{\boldsymbol{j}}^n:=\cos(\boldsymbol{j}\pi/n)$ for $\boldsymbol{j}\in\{0,\dots,n\}^K$ to the integrals in (2.3)

$$\begin{aligned}w_{\boldsymbol{j}} &:= \left(\tfrac{\pi}{2n}\right)^K\prod_{\ell=1}^K 2^{\mathbb{1}_{\{1,\dots,n-1\}}(j_\ell)},\qquad \boldsymbol{j}\in\{0,\dots,n\}^K,\\ \hat{f}_{\boldsymbol{k};n} &:= 2^{\mathbb{1}_{S(n)}(\boldsymbol{k})}\pi^{-K}\sum_{\boldsymbol{j}\in\{0,\dots,n\}^K} w_{\boldsymbol{j}}f(\cos(\boldsymbol{j}\pi/n))\cos(\boldsymbol{k}\boldsymbol{j}\pi/n) \qquad (2.4)\\ &= 2^{\mathbb{1}_{S(n)}(\boldsymbol{k})}(2n)^{-K}\sum_{\boldsymbol{j}\in\{0,\dots,2n-1\}^K} f(\cos(\boldsymbol{j}\pi/n))\cos(\boldsymbol{k}\boldsymbol{j}\pi/n)\\ &= 2^{\mathbb{1}_{S(n)}(\boldsymbol{k})}\,\mathrm{IFFT}\left((f(\boldsymbol{x}_{\boldsymbol{j}}^n))_{\boldsymbol{j}\in\{0,\dots,2n-1\}^K}\right)_{\boldsymbol{k}},\end{aligned}$$

where $S(n):=\{1,\dots,n-1\}$ and $\boldsymbol{k}\boldsymbol{j}=(k_1j_1,\dots,k_Kj_K)\in\mathbb{N}_0^K$ for $\boldsymbol{j}\in\{0,\dots,2n-1\}^K$. In addition, IFFT denotes the inverse fast Fourier transform. Note that $\cos((2n-j)\pi/n)=\cos(j\pi/n)$ for $j=1,\dots,n-1$, thus $(f(\boldsymbol{x}_{\boldsymbol{j}}^n))_{\boldsymbol{j}\in\{0,\dots,n\}^K}$ are sufficient to determine $(f(\boldsymbol{x}_{\boldsymbol{j}}^n))_{\boldsymbol{j}\in\{0,\dots,2n-1\}^K}$. If $k_j=n$

for some $j = 1, \dots, K$, the definition of $\hat{f}_{\boldsymbol{k};n}$ differs from the result of applying quadrature to (2.3) by a constant factor: In $2^{\mathbb{1}_{S(n)}(\boldsymbol{k})}$, the factor 2 is omitted for each $k_j = n$, $j = 1, \dots, K$, because of aliasing in the coefficients of the Chebyšev interpolant (cf. [37, Chapter 4]).

The approximation

$$\hat{p}_{f,n} := I_n^{\mathrm{CC}}[f] := \sum_{\boldsymbol{k} \in \{0,\dots,n\}^K} \hat{f}_{\boldsymbol{k};n} T_{\boldsymbol{k}} \tag{2.5}$$

of f is in fact the Lagrange interpolant in the nodes $(\boldsymbol{x}_{\boldsymbol{j}}^n)_{\boldsymbol{j} \in \{0,\dots,n\}^K}$ (cf. [33, Theorem 3.13] for $K = 1$, which can be applied inductively). It is known that the Lebesgue constant of tensor product Lagrange interpolation in these nodes satisfies

$$\left\| I_n^{\mathrm{CC}} \right\|_{L^\infty, L^\infty} := \left\| I_n^{\mathrm{CC}} \right\|_{L^\infty([-1,1]^K), L^\infty([-1,1]^K)} \leq (\tfrac{2}{\pi} \log(n+1) + 1)^K$$

(cf. [33, Theorem 1.2] for $K = 1$ and the zeroes of Chebyšev polynomials as nodes, from which the result for $(\boldsymbol{x}_{\boldsymbol{j}}^n)_{\boldsymbol{j} \in \{0,\dots,n\}^K}$ and $K = 1$ follows with [13, Theorem 4, Equation (4.6)]). Using Markov's inequality, a bound on the Lebesgue constant with respect to the $W^{1,\infty}$-norm can be obtained as follows:

$$\begin{aligned}
\left\| I_n^{\mathrm{CC}} \right\|_{W^{1,\infty}, W^{1,\infty}} &:= \left\| I_n^{\mathrm{CC}} \right\|_{W^{1,\infty}([-1,1]^K), W^{1,\infty}([-1,1]^K)} \\
&= \sup_{g \in W^{1,\infty}([-1,1]^K) \setminus \{0\}} \frac{\left\| I_n^{\mathrm{CC}}[g] \right\|_{W^{1,\infty}([-1,1]^K)}}{\|g\|_{W^{1,\infty}([-1,1]^K)}} \\
&\leq \sup_{g \in W^{1,\infty}([-1,1]^K) \setminus \{0\}} \frac{n^2 \left\| I_n^{\mathrm{CC}}[g] \right\|_{L^\infty([-1,1]^K)}}{\|g\|_{W^{1,\infty}([-1,1]^K)}} \\
&\leq \sup_{g \in W^{1,\infty}([-1,1]^K) \setminus \{0\}} \frac{n^2 \left\| I_n^{\mathrm{CC}} \right\|_{L^\infty, L^\infty} \|g\|_{L^\infty([-1,1]^K)}}{\|g\|_{W^{1,\infty}([-1,1]^K)}} \\
&\leq n^2 \left\| I_n^{\mathrm{CC}} \right\|_{L^\infty, L^\infty} .
\end{aligned}$$

In the third step, we used that for all $q \in \mathbb{Q}_n([-1,1]^K)$

$$\begin{aligned}
&\max_{i=1}^K \left\| \tfrac{\partial}{\partial x_i} q \right\|_{L^\infty([-1,1]^K)} \\
=& \max_{i=1}^K \max_{\substack{x_1,\dots,x_{i-1}, \\ x_{i+1},\dots,x_K \in [-1,1]}} \left\| (\tfrac{\partial}{\partial x_i} q)(x_1, \dots, x_{i-1}, \cdot, x_{i+1}, \dots, x_K) \right\|_{L^\infty([-1,1])} \\
\leq& \max_{i=1}^K \max_{\substack{x_1,\dots,x_{i-1}, \\ x_{i+1},\dots,x_K \in [-1,1]}} n^2 \left\| q(x_1, \dots, x_{i-1}, \cdot, x_{i+1}, \dots, x_K) \right\|_{L^\infty([-1,1])} \qquad (2.6) \\
=& n^2 \|q\|_{L^\infty([-1,1]^K)} ,
\end{aligned}$$

where we applied Markov's inequality to the univariate polynomial

$$z \mapsto q(x_1, \dots, x_{i-1}, z, x_{i+1}, \dots, x_K).$$

This computation holds in particular for polynomials $q \in \mathbb{Q}_n([-1,1]^K)$ which are themselves derivatives of other polynomials in $\mathbb{Q}_n([-1,1]^K)$, hence the previous argument can be iterated to obtain for all $s \in \mathbb{N}$

$$\left\|I_n^{\mathrm{CC}}\right\|_{W^{s,\infty},W^{s,\infty}} \leq n^{2s} \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}.$$

The arguments used above also prove the following, slightly stronger result:

$$\begin{aligned}\left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{1,\infty}} &:= \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty([-1,1]^K),W^{1,\infty}([-1,1]^K)} \\ &= \sup_{g\in W^{1,\infty}([-1,1]^K)\setminus\{0\}} \frac{\left\|I_n^{\mathrm{CC}}[g]\right\|_{W^{1,\infty}([-1,1]^K)}}{\|g\|_{L^\infty([-1,1]^K)}} \\ &\leq n^2 \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}. \end{aligned} \tag{2.7}$$

Again, we get the analogous result for arbitrary $s \in \mathbb{N}$ by iterating (2.6) s times:

$$\left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{s,\infty}} \leq n^{2s} \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}.$$

2.3 Chebyšev Interpolation Based On Approximate Function Values

In the case that the function f is only accessible through a (possibly corrupted) numerical approximation $\breve{f}$ in the Clenshaw–Curtis points $(\boldsymbol{x}_{\boldsymbol{j}}^n)_{\boldsymbol{j}\in\{0,\dots,n\}^K}$, a further approximation of the coefficients $\hat{f}_{\boldsymbol{k};n}$ is made. The function values of f in Equation (2.4) are replaced by function values of $\breve{f}$. For all $\boldsymbol{k} \in \{0,\dots,n\}^K$

$$\begin{aligned}\breve{f}_{\boldsymbol{k};n} &:= 2^{\mathbb{1}_{S(n)}(\boldsymbol{k})}\pi^{-K} \sum_{\boldsymbol{j}\in\{0,\dots,n\}^K} w_{\boldsymbol{j}} \breve{f}(\cos(\boldsymbol{j}\pi/n)) \cos(\boldsymbol{k}\boldsymbol{j}\pi/n) \qquad (2.8)\\ &= 2^{\mathbb{1}_{S(n)}(\boldsymbol{k})}(2n)^{-K} \sum_{\boldsymbol{j}\in\{0,\dots,2n-1\}^K} \breve{f}(\cos(\boldsymbol{j}\pi/n)) \cos(\boldsymbol{k}\boldsymbol{j}\pi/n) \\ &= 2^{\mathbb{1}_{S(n)}(\boldsymbol{k})} \operatorname{IFFT}\left((\breve{f}(\boldsymbol{x}_{\boldsymbol{j}}^n))_{\boldsymbol{j}\in\{0,\dots,2n-1\}^K}\right)_{\boldsymbol{k}}. \qquad (2.9)\end{aligned}$$

We define the corresponding interpolant $\breve{p}_{f,n}$ as

$$\breve{p}_{f,n} := \sum_{\boldsymbol{k}\in\{0,\dots,n\}^K} \breve{f}_{\boldsymbol{k};n} T_{\boldsymbol{k}} = I_n^{\mathrm{CC}}[\breve{f}]. \tag{2.10}$$

Lemma 2.2 *Let $K \in \mathbb{N}$, and let $f : [-1,1]^K \to \mathbb{R}$ be a map which admits a holomorphic complex extension to the isotropic Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}}$ with $\boldsymbol{\rho} = (\rho,\dots,\rho) \in (1,\infty)^K$ for some $\rho > 1$. We assume that an approximation $\breve{f}$ of f is available, and an upper bound on $\|f - \breve{f}\|_{L^\infty([-1,1]^K)}$. Then, for every $\rho' \in (1,\rho)$ and every $s \in \mathbb{N}$, there exists $C'(s,\rho,\rho') > 0$ such that*

$$\begin{aligned}\|f - \breve{p}_{f,n}\|_{L^\infty([-1,1]^K)} \leq{}& \left(1 + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}\right) K \left(\tfrac{2\rho}{\rho-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|\, \rho^{-n-1} \\ &+ \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}, \end{aligned} \tag{2.11}$$

$$\begin{aligned}
&\|f - \breve{p}_{f,n}\|_{W^{s,\infty}([-1,1]^K)} \\
&\quad \leq \left(1 + \left\|I_n^{\mathrm{CC}}\right\|_{W^{s,\infty},W^{s,\infty}}\right) K \left(\tfrac{2C'\rho'}{\rho'-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|\, {\rho'}^{-n-1} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{s,\infty}} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}. \qquad (2.12)
\end{aligned}$$

Proof We can estimate the error as follows. For all $q \in \mathbb{Q}_n$ we use $q = I_n^{\mathrm{CC}}[q]$ and obtain

$$\begin{aligned}
&\|f - \breve{p}_{f,n}\|_{L^\infty([-1,1]^K)} \\
&\leq \|f - q\|_{L^\infty([-1,1]^K)} + \|q - \hat{p}_{f,n}\|_{L^\infty([-1,1]^K)} + \|\hat{p}_{f,n} - \breve{p}_{f,n}\|_{L^\infty([-1,1]^K)} \\
&= \|f - q\|_{L^\infty([-1,1]^K)} + \left\|I_n^{\mathrm{CC}}[q - f]\right\|_{L^\infty([-1,1]^K)} + \left\|I_n^{\mathrm{CC}}[f - \breve{f}]\right\|_{L^\infty([-1,1]^K)} \\
&\leq \|f - q\|_{L^\infty([-1,1]^K)} + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \|f - q\|_{L^\infty([-1,1]^K)} \\
&\quad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}. \qquad (2.13)
\end{aligned}$$

By the same argument it follows that

$$\begin{aligned}
&\|f - \breve{p}_{f,n}\|_{W^{s,\infty}([-1,1]^K)} \\
&\quad \leq \|f - q\|_{W^{s,\infty}([-1,1]^K)} + \left\|I_n^{\mathrm{CC}}\right\|_{W^{s,\infty},W^{s,\infty}} \|f - q\|_{W^{s,\infty}([-1,1]^K)} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{s,\infty}} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}. \qquad (2.14)
\end{aligned}$$

Now, we can take the infimum over $q \in \mathbb{Q}_n$ and replace $\|f - q\|$ by the error of best polynomial approximation. Next, we discuss two upper bounds on the error of best polynomial approximation, which are sufficient for our purposes.

In case f is not only holomorphic on $\mathcal{E}_{\boldsymbol{\rho}}$, but also on the polydisk $B_{\boldsymbol{\rho}'} := \times_{j=1}^K \{z_j \in \mathbb{C} : |z_j| \leq \rho'\}$, with $\boldsymbol{\rho}' := (\rho', \ldots, \rho') \in (1,\infty)^K$ for some $\rho' \in (1,\infty)$, an upper bound on the error of best approximation follows by taking $q \in \mathbb{Q}_n$ to be the Taylor approximation of f in 0 and using a bound on the error of the Taylor approximation, e.g. [18, Theorem 5.1]. It then follows that $\|f - q\|_{L^\infty([-1,1]^K)} \leq C(\rho', f){\rho'}^{-n}$. Using that $\frac{\partial}{\partial x_j} q$ is the Taylor approximation of $\frac{\partial}{\partial x_j} f$ for $j = 1, \ldots, K$, the same bound on the error of the Taylor approximation gives that $\|f - q\|_{W^{1,\infty}([-1,1]^K)} \leq C(\rho', f){\rho'}^{-n+1} \leq C(\rho', f){\rho'}^{-n}$, where the constant was increased in the last step. Applying this argument s times gives a bound of the $W^{s,\infty}$-error.

If such a stronger condition of holomorphy on a polydisk does not hold, we can take q to be a truncation of the Chebyšev expansion: $q := \sum_{\boldsymbol{k}\in\{0,\ldots,n\}^K} f_{\boldsymbol{k}} T_{\boldsymbol{k}}$. We obtain a bound on the truncation error using the bound (2.2) on the Chebyšev coefficients. We use as notation $\Lambda_n^c := \mathbb{N}_0^K \setminus \{0,\ldots,n\}^K$ and $\boldsymbol{k}^{(j)} := (0,\ldots,0,n+1,0,\ldots,0) \in \mathbb{N}_0^K$ for $j = 1,\ldots,K$, with the nonzero component in the j'th position. Using that $\Lambda_n^c = \bigcup_{j=1}^K \{\boldsymbol{k}^{(j)} + \boldsymbol{k} : \boldsymbol{k} \in \mathbb{N}_0^K\}$, and the fact

that $\|T_{\boldsymbol{k}}\|_{L^\infty([-1,1]^K)} = 1$ for all $\boldsymbol{k} \in \mathbb{N}_0^K$, it follows that

$$\begin{aligned}\|f-q\|_{L^\infty([-1,1]^K)} &\leq \sum_{\Lambda_n^c} |f_{\boldsymbol{k}}| \, \|T_{\boldsymbol{k}}\|_{L^\infty([-1,1]^K)} \leq \sum_{j=1}^K \sum_{\boldsymbol{k}\in\mathbb{N}_0^K} \left|f_{\boldsymbol{k}^{(j)}+\boldsymbol{k}}\right| \\ &\leq \sum_{j=1}^K \sum_{\boldsymbol{k}\in\mathbb{N}_0^K} 2^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| \boldsymbol{\rho}^{-\boldsymbol{k}^{(j)}-\boldsymbol{k}} \\ &\leq K \left(\tfrac{2\rho}{\rho-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| \, \rho^{-n-1}.\end{aligned}$$

For all $s \in \mathbb{N}$, bounds on the $W^{s,\infty}$-error follow similarly, using that for all $\rho' \in (1,\rho)$ there exists $C'(s,\rho,\rho') > 0$ such that $\rho^{-k}k^{2s} \leq C'\rho'^{-k}$ for all $k \in \mathbb{N}_0$. By the Markov inequality, the definition of the $\|\circ\|_{W^{s,\infty}}$-norm (cf. Sec. 1.3), and the fact that $\|T_{\boldsymbol{k}}\|_{L^\infty([-1,1]^K)} = 1$ for all $\boldsymbol{k} \in \mathbb{N}_0^K$, we get

$$\begin{aligned}\|T_{\boldsymbol{k}}\|_{W^{s,\infty}([-1,1]^K)} &= \max_{\boldsymbol{i}\in\mathbb{N}_0^K:\|\boldsymbol{i}\|_{\ell^1}\leq s} \left\| \partial_{\boldsymbol{x}}^{\boldsymbol{i}} \prod_{\ell=1}^K T_{k_\ell}(x_\ell) \right\|_{L^\infty([-1,1]^K)} \\ &= \max_{\boldsymbol{i}\in\mathbb{N}_0^K:\|\boldsymbol{i}\|_{\ell^1}\leq s} \prod_{\ell=1}^K \left\| \frac{\partial^{i_\ell}}{\partial x_\ell^{i_\ell}} T_{k_\ell}(\cdot) \right\|_{L^\infty([-1,1])} \\ &\leq \max_{\boldsymbol{i}\in\mathbb{N}_0^K:\|\boldsymbol{i}\|_{\ell^1}\leq s} \boldsymbol{k}^{2\boldsymbol{i}} \leq \prod_{\ell=1}^K \max\{1, k_\ell^{2s}\}.\end{aligned}$$

We obtain that for all $\rho' \in (1,\rho)$

$$\begin{aligned}&\|f-q\|_{W^{s,\infty}([-1,1]^K)} \\ &\leq \sum_{\Lambda_n^c} |f_{\boldsymbol{k}}| \, \|T_{\boldsymbol{k}}\|_{W^{s,\infty}([-1,1]^K)} \\ &\leq \sum_{j=1}^K \sum_{\boldsymbol{k}\in\mathbb{N}_0^K} \left|f_{\boldsymbol{k}^{(j)}+\boldsymbol{k}}\right| \prod_{\ell=1}^K \max\{1, (\boldsymbol{k}^{(j)}+\boldsymbol{k})_\ell^{2s}\} \\ &\leq \sum_{j=1}^K \sum_{\boldsymbol{k}\in\mathbb{N}_0^K} 2^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| \boldsymbol{\rho}^{-\boldsymbol{k}^{(j)}-\boldsymbol{k}} \prod_{\ell=1}^K \max\{1, (\boldsymbol{k}^{(j)}+\boldsymbol{k})_\ell^{2s}\} \\ &\leq K 2^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| \left(\sum_{k=0}^\infty \max\{1,k^{2s}\}\rho^{-k}\right)^{K-1} \left(\sum_{k=n+1}^\infty \max\{1,k^{2s}\}\rho^{-k}\right) \\ &\leq K 2^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| \left(\sum_{k=0}^\infty C'\rho'^{-k}\right)^{K-1} \left(\sum_{k=n+1}^\infty C'\rho'^{-k}\right) \\ &\leq K \left(\tfrac{2C'\rho'}{\rho'-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| \, \rho'^{-n-1}.\end{aligned}$$

With (2.13) and (2.14) we obtain

$$
\begin{aligned}
&\|f - \breve{p}_{f,n}\|_{L^\infty([-1,1]^K)} \\
&\quad \le \|f - q\|_{L^\infty([-1,1]^K)} + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \|f - q\|_{L^\infty([-1,1]^K)} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)} \\
&\quad \le \left(1 + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}\right) K \left(\tfrac{2\rho}{\rho-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|\, \rho^{-n-1} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}, \\
&\|f - \breve{p}_{f,n}\|_{W^{s,\infty}([-1,1]^K)} \\
&\quad \le \|f - q\|_{W^{s,\infty}([-1,1]^K)} + \left\|I_n^{\mathrm{CC}}\right\|_{W^{s,\infty},W^{s,\infty}} \|f - q\|_{W^{s,\infty}([-1,1]^K)} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{s,\infty}} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)} \\
&\quad \le \left(1 + \left\|I_n^{\mathrm{CC}}\right\|_{W^{s,\infty},W^{s,\infty}}\right) K \left(\tfrac{2C'\rho'}{\rho'-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|\, {\rho'}^{-n-1} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{s,\infty}} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}.
\end{aligned}
$$

Remark 2.3 The estimates in Equations (2.13) and (2.14) apply more generally also for functions of finite regularity, using Jackson-type estimates on the error of best polynomial approximation. Error estimates for interpolation of functions of finite regularity on a Clenshaw–Curtis grid could also be based on other arguments. For example, see the proof of [27, Theorem 2.1], which uses such a result to show a lower bound on the convergence rate of DNNs with a smooth activation function.

The previous error bound leads to the following bound on the approximate Chebyšev coefficients:

Corollary 2.4 *Let $K \in \mathbb{N}$, and let $f : [-1,1]^K \to \mathbb{R}$ be a map which admits a holomorphic complex extension to the Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}}$ with $\boldsymbol{\rho} = (\rho,\dots,\rho) \in (1,\infty)^K$ for some $\rho > 1$. We assume that an approximation $\breve{f}$ of f is available, and an upper bound on $\|f - \breve{f}\|_{L^\infty([-1,1]^K)}$. Then,*

$$
\begin{aligned}
&\sum_{\boldsymbol{k}\in\{0,\dots,n\}^K} \left|\breve{f}_{\boldsymbol{k};n}\right| \qquad (2.15)\\
&\quad \le C(K,\rho) \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + (n+1)^K \pi^K \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}.
\end{aligned}
$$

Proof We compute

$$\begin{aligned}
&\left|f_{\boldsymbol{k}} - \breve{f}_{\boldsymbol{k};n}\right| \\
&\leq \left|\int_{-1}^{1}\cdots\int_{-1}^{1}(f-\breve{p}_{f,n})T_{\boldsymbol{k}}\,\mathrm{d}\lambda(x_1)\cdots\mathrm{d}\lambda(x_K)\right| \\
&\leq \|f-\breve{p}_{f,n}\|_{L^\infty([-1,1]^K)}\,\|T_{\boldsymbol{k}}\|_{L^\infty([-1,1]^K)}\left|\int_{-1}^{1}\cdots\int_{-1}^{1}\mathrm{d}\lambda(x_1)\cdots\mathrm{d}\lambda(x_K)\right| \\
&= \pi^K\,\|f-\breve{p}_{f,n}\|_{L^\infty([-1,1]^K)}
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{\boldsymbol{k}\in\{0,\ldots,n\}^K}\left|\breve{f}_{\boldsymbol{k};n}\right| \\
&\leq \sum_{\boldsymbol{k}\in\mathbb{N}_0^K}|f_{\boldsymbol{k}}| + \sum_{\boldsymbol{k}\in\{0,\ldots,n\}^K}\left|f_{\boldsymbol{k}}-\breve{f}_{\boldsymbol{k};n}\right| \\
&\leq \left(\tfrac{2\rho}{\rho-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})| + (n+1)^K\pi^K\,\|f-\breve{p}_{f,n}\|_{L^\infty([-1,1]^K)} \\
&\leq \left(\tfrac{2\rho}{\rho-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})|\left[1+K(n+1)^K\pi^K\left(1+\left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}\right)\rho^{-n-1}\right] \\
&\quad + (n+1)^K\pi^K\left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}\left\|f-\breve{f}\right\|_{L^\infty([-1,1]^K)}.
\end{aligned}$$

Here, $\left(\frac{2\rho}{\rho-1}\right)^K\left[1+K(n+1)^K\pi^K\left(1+\left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}\right)\rho^{-n-1}\right]$ is bounded by a constant $C(K,\rho)>0$ independent of n.

Remark 2.5 For all $\breve{f},\breve{g}\in\mathbb{C}^0([-1,1]^K)$, it follows from (2.8) and the positivity of the quadrature weights $w_{\boldsymbol{j}}$, which are defined just above (2.4) and sum up to π^K, that for all $n\in\mathbb{N}$ and $\boldsymbol{k}\in\{0,\ldots,n\}^K$

$$\left|\breve{f}_{\boldsymbol{k};n}-\breve{g}_{\boldsymbol{k};n}\right| \leq 2^{\mathbb{1}_{S(n)}(\boldsymbol{k})}\left\|\breve{f}-\breve{g}\right\|_{L^\infty([-1,1]^K)} \leq 2^K\left\|\breve{f}-\breve{g}\right\|_{L^\infty([-1,1]^K)}.$$

3 Constructive Deep Neural Network Approximation of Multivariate Holomorphic Functions

As we showed in [30] by (straightforward) tensorization of the univariate results, multivariate holomorphic maps $f:[-1,1]^K\to\mathbb{R}$ admit DNN surrogates which approximate f to any accuracy $\varepsilon>0$ with DNN size scaling as $O(|\log(\varepsilon)|^{K+1})$ (with the constant implied in $O(\cdot)$ still depending on the input dimension K).

In the present section, we shall recapitulate this result, with a new proof via multivariate, tensorized Chebyšev expansions.

3.1 Definitions and Architecture of ReLU DNNs

We consider *feed-forward deep neural networks*. These DNNs result from repeated application of affine mappings and a specific non-linear map. This nonlinearity is specified via the so-called *activation function* $\sigma : \mathbb{R} \to \mathbb{R}$ of the DNN. Here, we take $\sigma(x) = \max\{0, x\}$, $x \in \mathbb{R}$, to be the *rectified linear unit (ReLU)* activation function. The architecture of the DNN comprises a fixed *number of hidden layers* $L \in \mathbb{N}_0$, numbers $N_\ell \in \mathbb{N}$ of *computation nodes in layer* $\ell \in \{1, \dots, L+1\}$, the map $\Phi : \mathbb{R}^{N_0} \to \mathbb{R}^{N_{L+1}}$ is said to be realized by a feedforward neural network, if for certain *weights* $A^\ell_{i,j} \in \mathbb{R}$, and *biases* $b^\ell_j \in \mathbb{R}$ it holds for all $\boldsymbol{x} = (x_i)_{i=1}^{N_0}$

$$w^1_j = \sigma\left(\sum_{i=1}^{N_0} A^1_{i,j} x_i + b^1_j\right) , \quad j \in \{1, \dots, N_1\} ,$$

and

$$w^{\ell+1}_j = \sigma\left(\sum_{i=1}^{N_\ell} A^{\ell+1}_{i,j} w^\ell_i + b^{\ell+1}_j\right) , \quad \ell \in \{1, \dots, L-1\}, \;\; j \in \{1, \dots, N_{\ell+1}\} ,$$

and finally

$$\Phi(\boldsymbol{x}) = (w^{L+1}_j)_{j=1}^{N_{L+1}} = \left(\sum_{i=1}^{N_L} A^{L+1}_{i,j} w^L_i + b^{L+1}_j\right)_{j=1}^{N_{L+1}} .$$

In this case N_0 is the dimension of the DNN input, and N_{L+1} is the dimension of the output. The number of hidden layers L of a DNN is referred to as its *depth*, denoted by $\mathrm{depth}(\Phi)$. If $L = 0$, then the previous equation holds with $w^0_i := x_i$ for $i = 1, \dots, N_0$ and we refer to such networks, realizing affine functions, as DNNs of depth 0. Also, we define the total number of nonzero weights and biases as the *size* of the DNN, i.e. $\mathrm{size}(\Phi) := |\{(i, j, \ell) : A^\ell_{i,j} \neq 0\}| + |\{(j, \ell) : b^\ell_j \neq 0\}|$.

We do emphasize that different DNN architectures, weights and biases can realize the same function. In this paper, we focus on a constructive procedure to find weights and biases such that the resulting DNN as a function approximates a given holomorphic map to a specified accuracy.

To construct such networks from smaller subnetworks, we use DNN concatenation from [31, Remark 2.6], parallelization from [31, Definition 2.7] and [14, Setting 5.2], and networks emulating the identity from [31, Lemma 2.3], which we all recall below.

Let f and g be two DNNs with the same depth $L \in \mathbb{N}_0$ and the same input dimension $n \in \mathbb{N}$. Denote by m_f the output dimension of f and by m_g the output dimension of g. There exists a neural network (f, g), called *parallelization* of f and g, which in parallel emulates f and g, i.e.

$$(f, g) : \mathbb{R}^n \to \mathbb{R}^{m_f} \times \mathbb{R}^{m_g} : \boldsymbol{x} \mapsto (f(\boldsymbol{x}), g(\boldsymbol{x})),$$

and it satisfies $\operatorname{depth}((f,g)) = L$ and $\operatorname{size}((f,g)) = \operatorname{size}(f) + \operatorname{size}(g)$ ([31, Definition 2.7]).

Let f and g be ReLU DNNs, such that the number of nodes in the output layer of g equals the number of nodes in the input layer of f. Denote by n the number of nodes in the input layer of g, and by m the number of nodes in the output layer of f. There exists a DNN $f \circ g$, called *sparse concatenation of the DNNs f and g*, which we will refer to simply as *concatenation of f and g*, which realizes the composition of f and g, i.e.

$$f \circ g : \mathbb{R}^n \to \mathbb{R}^m : \boldsymbol{x} \mapsto f(g(\boldsymbol{x})). \tag{3.1}$$

It satisfies $\operatorname{depth}(f \circ g) = \operatorname{depth}(f) + 1 + \operatorname{depth}(g)$ and $\operatorname{size}(f \circ g) \leq 2\operatorname{size}(f) + 2\operatorname{size}(g)$ ([31, Remark 2.6]).

Next, let f and g be two DNNs with the same depth $L \in \mathbb{N}_0$, whose input dimensions n_f and n_g may be different, and whose output dimensions we will denote by m_f and m_g, respectively. There exists a DNN $(f,g)_{\mathrm{d}}$, called *full parallelization of networks with distinct inputs* of f and g, which in parallel emulates f and g, i.e.

$$(f,g)_{\mathrm{d}} : \mathbb{R}^{n_f} \times \mathbb{R}^{n_g} \to \mathbb{R}^{m_f} \times \mathbb{R}^{m_g} : (\boldsymbol{x}, \tilde{\boldsymbol{x}}) \mapsto (f(\boldsymbol{x}), g(\tilde{\boldsymbol{x}})).$$

It satisfies $\operatorname{depth}((f,g)_{\mathrm{d}}) = L$ and $\operatorname{size}((f,g)_{\mathrm{d}}) = \operatorname{size}(f) + \operatorname{size}(g)$ ([14, Setting 5.2]).

Finally, for all $n \in \mathbb{N}$ and $L \in \mathbb{N}_0$ there exists an *identity network* $\Phi^{\mathrm{Id}}_{n,L}$ of depth L which emulates the identity map $\mathrm{Id}_{\mathbb{R}^n} : \mathbb{R}^n \to \mathbb{R}^n : \boldsymbol{x} \mapsto \boldsymbol{x}$. It satisfies $\operatorname{size}(\Phi^{\mathrm{Id}}_{n,L}) \leq 2n(L+1)$ ([31, Lemma 2.3]).

3.2 ReLU DNN Approximations of Chebyšev Polynomials

We construct the ReLU DNN approximation of f as a reapproximation of $\breve{p}_{f,n}$ defined in (2.10). The coefficients of $\breve{p}_{f,n}$ can be computed with (2.9). The polynomial approximation $\breve{p}_{f,n}$ is a tensor product of univariate Chebyšev polynomials. We use the ReLU DNN approximation of univariate Chebyšev polynomials from [28], and for the ReLU DNN approximation of products with multiple arguments, we recall [30, Proposition 2.6], based on [35, Proposition 3.1].

Lemma 3.1 ([28]) *There exists $C > 0$ such that for all $n \in \mathbb{N}$ there exist ReLU DNNs $\left\{\Phi^{\mathrm{Cheb},n}_{\delta}\right\}_{\delta \in (0,1)}$ with input dimension one and output dimension n which satisfy*

$$\begin{aligned}
\left\| T_\ell - \left(\Phi^{\mathrm{Cheb},n}_{\delta}\right)_\ell \right\|_{W^{1,\infty}([-1,1])} &\leq \delta, \qquad \ell = 1, \dots, n, \\
\operatorname{depth}\left(\Phi^{\mathrm{Cheb},n}_{\delta}\right) &\leq C(1+\log(n))\log(1/\delta) + C(1+\log(n))^3, \\
\operatorname{size}\left(\Phi^{\mathrm{Cheb},n}_{\delta}\right) &\leq Cn\log(1/\delta) + Cn(1+\log(n)).
\end{aligned}$$

The construction of these networks is described in Appendix A.

Proposition 3.2 ([30, Proposition 2.6]) *For any $\delta \in (0,1)$, $n \in \mathbb{N}$ and $M \geq 1$ there exists a ReLU DNN $\tilde{\prod}_{\delta,M}^n : [-M,M]^n \to \mathbb{R}$ such that*

$$\sup_{(x_i)_{i=1}^n \in [-M,M]^n} \left| \prod_{j=1}^n x_j - \tilde{\prod}_{\delta,M}^n (x_1, \ldots, x_n) \right| \leq \delta, \tag{3.2}$$

$$\operatorname*{ess\,sup}_{(x_i)_{i=1}^n \in [-M,M]^n} \sup_{i=1,\ldots,n} \left| \frac{\partial}{\partial x_i} \prod_{j=1}^n x_j - \frac{\partial}{\partial x_i} \tilde{\prod}_{\delta,M}^n (x_1, \ldots, x_n) \right| \leq \delta. \tag{3.3}$$

There exists a constant C independent of $\delta \in (0,1)$, $n \in \mathbb{N}$ and $M \geq 1$ such that

$$\begin{aligned} \operatorname{size}\left(\tilde{\prod}_{\delta,M}^n\right) &\leq C(1 + n \log(nM^n/\delta)), \\ \operatorname{depth}\left(\tilde{\prod}_{\delta,M}^n\right) &\leq C(1 + \log(n) \log(nM^n/\delta)). \end{aligned} \tag{3.4}$$

Similar to [30, Proposition 2.13], there holds the following result on ReLU DNN emulation rates for Chebyšev polynomials, which is of independent interest. As compared to the results in [30] and as observed in [36], the functional structure of Chebyšev polynomials affords gains in DNN depth and size bounds. To state the depth and size bounds, we use the notation m_∞ defined in Section 1.3.

Proposition 3.3 *There exists a constant $C > 0$, such that for every $K \in \mathbb{N}$, every finite subset $\Lambda \subset \mathbb{N}_0^K$ and every $\delta \in (0,1)$ there exists a ReLU DNN $\Phi_{\Lambda,\delta}$ with input dimension K and output dimension $|\Lambda|$, such that the outputs of $\Phi_{\Lambda,\delta}$, which we denote by $\{\tilde{T}_{\boldsymbol{k},\delta}\}_{\boldsymbol{k}\in\Lambda}$, satisfy*

$$\forall \boldsymbol{k} \in \Lambda : \qquad \left\| T_{\boldsymbol{k}} - \tilde{T}_{\boldsymbol{k},\delta} \right\|_{W^{1,\infty}([-1,1]^K)} \leq \delta,$$

$$\begin{aligned} \operatorname{depth}(\Phi_{\Lambda,\delta}) &\leq C(1 + \log m_\infty(\Lambda))^3 + C(1 + \log(K) + \log m_\infty(\Lambda)) \log(1/\delta) \\ &\quad + CK \log(m_\infty(\Lambda)) + CK \log K, \\ \operatorname{size}(\Phi_{\Lambda,\delta}) &\leq CK|\Lambda| \log(m_\infty(\Lambda)) + CK|\Lambda| \log(1/\delta) + CK^2|\Lambda| \\ &\quad + CKm_\infty(\Lambda) \log(m_\infty(\Lambda)) + CKm_\infty(\Lambda) \log(1/\delta) + CK^2 m_\infty(\Lambda). \end{aligned}$$

Proof This proof consists of three steps. In the first step, we define the network. In the second step, we estimate the error. In the third step, we give bounds on the network depth and size.

Step 1. We construct the network $\Phi_{\Lambda,\delta}$ as the concatenation of two subnetworks:

$$\Phi_{\Lambda,\delta} := \Phi_{\Lambda,\delta}^{(1)} \circ \Phi_{\Lambda,\delta}^{(2)}.$$

See Figure 3.1 for a sketch of the network structure.

For $\beta = \frac{1}{2}\delta(1+\delta)^{-K}K^{-1}(m_\infty(\Lambda)^2+1)^{-1} \leq \delta$, the network $\Phi^{(2)}_{\Lambda,\delta}$ in parallel approximates univariate Chebyšev polynomials up to degree $m_\infty(\Lambda)$, in all K input variables, and is based on Lemma 3.1: Denoting the input of $\Phi_{\Lambda,\delta}$ by $\boldsymbol{y} = (y_1, \ldots, y_K) \in [-1,1]^K$,

$$\Phi^{(2)}_{\Lambda,\delta} := \left(\Phi^{\mathrm{Cheb},m_\infty(\Lambda)}_\beta, \ldots, \Phi^{\mathrm{Cheb},m_\infty(\Lambda)}_\beta\right)_{\mathrm{d}},$$

which contains K copies of $\Phi^{\mathrm{Cheb},m_\infty(\Lambda)}_\beta$, the j'th of which receives y_j as input, for $j = 1, \ldots, K$.

For $\beta' = \frac{1}{2}\delta(m_\infty(\Lambda)^2+1)^{-1}$, the network $\Phi^{(1)}_{\Lambda,\delta}$ computes tensor products of univariate Chebyšev polynomials using networks from Proposition 3.2. Denoting the output of $\Phi^{(2)}_{\Lambda,\delta}$ by $\left\{\tilde{T}_{k,\beta}(y_j)\right\}_{\substack{j=1,\ldots,K\\ k=1,\ldots,m_\infty(\Lambda)}}$, it holds that

$$\Phi^{(1)}_{\Lambda,\delta} \circ \Phi^{(2)}_{\Lambda,\delta}(y_1, \ldots, y_K) = \left(\left\{\tilde{\prod}^K_{\beta',1+\delta}\left(\tilde{T}_{k_1,\beta}(y_1), \ldots, \tilde{T}_{k_K,\beta}(y_K)\right)\right\}_{\boldsymbol{k}\in\Lambda}\right),$$

where, in case $k_j = 0$, the factor $\tilde{T}_{0,\beta} \equiv 1$ is implemented by a bias in the first layer of $\tilde{\prod}^K_{\beta',1+\delta}$. We used that

$$\begin{aligned}\left\|\tilde{T}_{k_j,\beta}\right\|_{L^\infty([-1,1])} &\leq \left\|T_{k_j}\right\|_{L^\infty([-1,1])} + \left\|T_{k_j} - \tilde{T}_{k_j,\beta}\right\|_{L^\infty([-1,1])}\\ &\leq 1+\beta \leq 1+\delta \leq 2,\end{aligned}$$

so that $\{\tilde{T}_{k_j,\beta}(y_j)\}_{j=1}^K$ can be used as inputs of $\tilde{\prod}^K_{\beta',1+\delta}$.

Step 2. The error can be estimated as follows:

$$\begin{aligned}&\sup_{\boldsymbol{y}\in[-1,1]^K}\left|T_{\boldsymbol{k}}(\boldsymbol{y}) - \tilde{T}_{\boldsymbol{k},\delta}(\boldsymbol{y})\right|\\ \leq\ &\sup_{\boldsymbol{y}\in[-1,1]^K}\left|T_{\boldsymbol{k}}(\boldsymbol{y}) - \prod_{j=1}^K \tilde{T}_{k_j,\beta}(y_j)\right|\\ &+ \sup_{\boldsymbol{y}\in[-1,1]^K}\left|\prod_{j=1}^K \tilde{T}_{k_j,\beta}(y_j) - \tilde{\prod}^K_{\beta',1+\delta}\left(\left\{\tilde{T}_{k_j,\beta}(y_j)\right\}_{j=1}^K\right)\right|\\ \leq\ &\sup_{\boldsymbol{y}\in[-1,1]^K}\sum_{i=1}^K\left|\prod_{j=1,\ldots,i-1}\tilde{T}_{k_j,\beta}(y_j)\right|\cdot\left|T_{k_i}(y_i) - \tilde{T}_{k_i,\beta}(y_i)\right|\cdot\left|\prod_{j=i+1,\ldots,K}T_{k_j}(y_j)\right|\\ &+\beta'\\ \leq\ &\left(\sum_{i=1}^K(1+\delta)^{i-1}\beta\right) + \beta' \leq K(1+\delta)^K\beta + \beta' \leq \delta.\end{aligned}$$

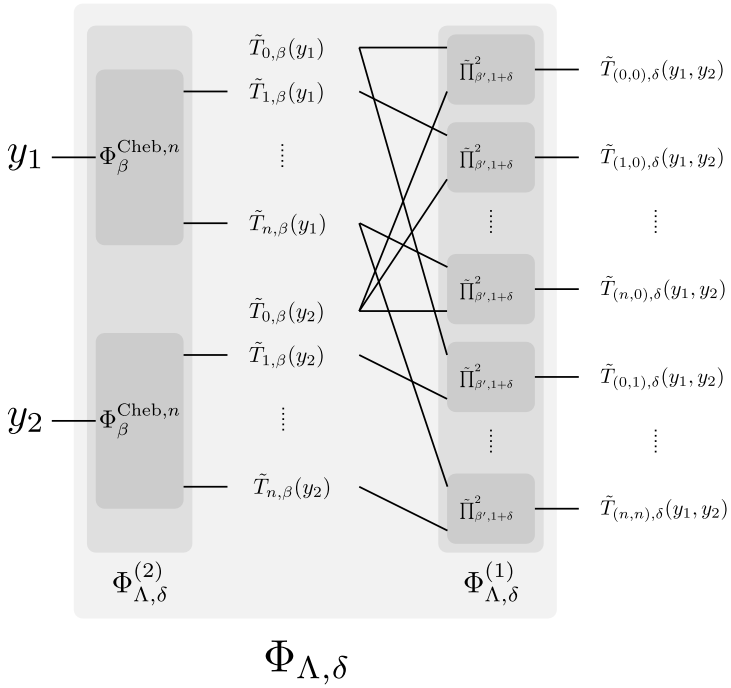


Fig. 3.1: Sketch of $\Phi_{\Lambda,\delta}$ for $\Lambda = \{0,\dots,n\}^K$, with $n \in \mathbb{N}$, $K = 2$, and $\delta \in (0,1)$. The subnetwork $\Phi_\beta^{\mathrm{Cheb},n}$ from Lemma 3.1 approximates univariate Chebyšev polynomials of degrees $1,\dots,n$, which are then multiplied by the networks $\tilde{\prod}^2_{\beta',1+\delta}$ from Proposition 3.2 to obtain approximations of tensor product Chebyšev polynomials. The Chebyšev polynomial of degree 0 is identically equal to 1 and implemented by a bias in the first layer of the product networks $\tilde{\prod}^2_{\beta',1+\delta}$.

For the estimate on the error in the derivative, we consider only the derivative with respect to y_1. The derivatives with respect to the other inputs satisfy analogous bounds. Using that

$$\begin{aligned}\left|\tilde{T}_{k_j,\beta}\right|_{W^{1,\infty}([-1,1])} &\leq \left|T_{k_j}\right|_{W^{1,\infty}([-1,1])} + \left|T_{k_j} - \tilde{T}_{k_j,\beta}\right|_{W^{1,\infty}([-1,1])} \\ &\leq m_\infty(\Lambda)^2 + \beta \leq m_\infty(\Lambda)^2 + 1,\end{aligned}$$

we obtain

$$\begin{aligned}&\operatorname*{ess\,sup}_{\boldsymbol{y}\in[-1,1]^K} \left|\frac{\partial}{\partial y_1} T_{\boldsymbol{k}}(\boldsymbol{y}) - \frac{\partial}{\partial y_1}\tilde{T}_{\boldsymbol{k},\delta}(\boldsymbol{y})\right| \\ \leq &\operatorname*{ess\,sup}_{\boldsymbol{y}\in[-1,1]^K} \left|\frac{\partial}{\partial y_1} T_{\boldsymbol{k}}(\boldsymbol{y}) - \frac{\partial}{\partial y_1}\prod_{j=1}^K \tilde{T}_{k_j,\beta}(y_j)\right|\end{aligned}$$

$$
\begin{aligned}
&+ \operatorname*{ess\,sup}_{\boldsymbol{y}\in[-1,1]^K} \left| \frac{\partial}{\partial y_1} \prod_{j=1}^{K} \tilde{T}_{k_j,\beta}(y_j) - \frac{\partial}{\partial y_1} \tilde{\prod}_{\beta',1+\delta}^{K} \left(\left\{ \tilde{T}_{k_j,\beta}(y_j) \right\}_{j=1}^{K} \right) \right| \\
&\leq \operatorname*{ess\,sup}_{\boldsymbol{y}\in[-1,1]^K} \left| \frac{\partial}{\partial y_1} T_{k_1}(y_1) - \frac{\partial}{\partial y_1} \tilde{T}_{k_1,\beta}(y_1) \right| \cdot \left| \prod_{j=2,\dots,K} T_{k_j}(y_j) \right| \\
&\quad + \operatorname*{ess\,sup}_{\boldsymbol{y}\in[-1,1]^K} \sum_{i=2,\dots,K} \left| \frac{\partial}{\partial y_1} \tilde{T}_{k_1,\beta}(y_1) \right| \cdot \left| \prod_{j=2,\dots,i-1} \tilde{T}_{k_j,\beta}(y_j) \right| \\
&\qquad \cdot \left| T_{k_i}(y_i) - \tilde{T}_{k_i,\beta}(y_i) \right| \cdot \left| \prod_{j=i+1,\dots,K} T_{k_j}(y_j) \right| \\
&\quad + \operatorname*{ess\,sup}_{\boldsymbol{y}\in[-1,1]^K} \left| \prod_{j=2,\dots,K} \tilde{T}_{k_j,\beta}(y_j) - \left(\frac{\partial}{\partial x_1} \tilde{\prod}_{\beta',1+\delta}^{K} \right) \left(\left\{ \tilde{T}_{k_j,\beta}(y_j) \right\}_{j=1}^{K} \right) \right| \\
&\qquad \cdot \left| \frac{\partial}{\partial y_1} \tilde{T}_{k_1,\beta}(y_1) \right| \\
&\leq \beta + \left(\sum_{i=2}^{K} (m_\infty(\Lambda)^2+1)(1+\delta)^{i-2}\beta \right) + \beta'(m_\infty(\Lambda)^2+1) \\
&\leq K(1+\delta)^K \beta (m_\infty(\Lambda)^2+1) + \beta'(m_\infty(\Lambda)^2+1) \\
&\leq \tfrac{\delta}{2} + \tfrac{\delta}{2} \leq \delta,
\end{aligned}
$$

where $\frac{\partial}{\partial x_1}\tilde{\prod}_{\beta',1+\delta}^{K}$ denotes the (weak) derivative of $\tilde{\prod}_{\beta',1+\delta}^{K} : [-1-\delta, 1+\delta]^K \to \mathbb{R}$ with respect to its first argument, cf. Proposition 3.2.

Step 3. As a bound on the network depth and size we obtain, using $\log(1+\delta) \leq 1$ in the last step,

$$
\begin{aligned}
\operatorname{depth}(\Phi_{\Lambda,\delta}) &\leq \operatorname{depth}\left(\Phi_{\Lambda,\delta}^{(1)}\right) + 1 + \operatorname{depth}\left(\Phi_{\Lambda,\delta}^{(2)}\right) \\
&\leq \operatorname{depth}\left(\tilde{\prod}_{\beta',1+\delta}^{K}\right) + 1 + \operatorname{depth}\left(\Phi_{\beta}^{\mathrm{Cheb},m_\infty(\Lambda)}\right) \\
&\leq C(1+\log(K)\log(K(1+\delta)^K/\beta')) + 1 \\
&\quad + C(1+\log m_\infty(\Lambda))\log(1/\beta) + C(1+\log m_\infty(\Lambda))^3 \\
&\leq C\big(\log(K)^2 + \log(K)K\log(1+\delta) + \log(K)\log(m_\infty(\Lambda))\big) \\
&\quad + C\big(\log(K)\log(1/\delta) + (1+\log m_\infty(\Lambda))^2 \\
&\qquad + (\log(K) + K\log(1+\delta))(1+\log m_\infty(\Lambda)) \\
&\qquad + (1+\log m_\infty(\Lambda))\log(1/\delta)\big) + C(1+\log m_\infty(\Lambda))^3 \\
&\leq C(1+\log m_\infty(\Lambda))^3 + C(1+\log(K)+\log m_\infty(\Lambda))\log(1/\delta) \\
&\quad + CK\log(m_\infty(\Lambda)) + CK\log K, \\
\operatorname{size}(\Phi_{\Lambda,\delta}) &\leq C\left[\operatorname{size}\left(\Phi_{\Lambda,\delta}^{(1)}\right) + \operatorname{size}\left(\Phi_{\Lambda,\delta}^{(2)}\right)\right]
\end{aligned}
$$

$$
\begin{aligned}
&\leq C\left[|\Lambda|\,\mathrm{size}\left(\tilde{\prod}^{K}_{\beta',1+\delta}\right) + K\,\mathrm{size}\left(\Phi_\beta^{\mathrm{Cheb},m_\infty(\Lambda)}\right)\right]\\
&\leq C\big[|\Lambda|\big(1+K\log(K(1+\delta)^K/\beta')\big) + K\big(m_\infty(\Lambda)\log(1/\beta)\\
&\quad + m_\infty(\Lambda)(1+\log m_\infty(\Lambda))\big)\big]\\
&\leq C\big[|\Lambda|\big(K\log(K) + K^2\log(1+\delta) + K\log(m_\infty(\Lambda)) + K\log(1/\delta)\big)\\
&\quad + K\big(m_\infty(\Lambda)\log(m_\infty(\Lambda)) + m_\infty(\Lambda)\log(1/\delta)\\
&\quad + (\log(K) + K\log(1+\delta))m_\infty(\Lambda) + m_\infty(\Lambda)(1+\log m_\infty(\Lambda))\big)\big]\\
&\leq C\big[K|\Lambda|\log(m_\infty(\Lambda)) + K|\Lambda|\log(1/\delta) + K^2|\Lambda|\\
&\quad + Km_\infty(\Lambda)\log(m_\infty(\Lambda)) + Km_\infty(\Lambda)\log(1/\delta) + K^2 m_\infty(\Lambda)\big].
\end{aligned}
$$

This finishes the proof.

Remark 3.4 For $\delta \leq \exp(1/K) - 1$ it holds $K\log(1+\delta) \leq 1$, so that the network depth is of the order $\log(K)^2$ and the explicit K-dependent factors in the bound on the network size are of the order $K\log(K)$ (in addition, the network size may depend on K indirectly through $|\Lambda|$). A sufficient condition is that $\delta \leq 1/K$, because $\exp(x) - 1 \geq x$ for all $x \geq 0$.

3.3 Construction of ReLU DNN Approximations of Multivariate Holomorphic Functions

Proposition 3.5 *Let $K \in \mathbb{N}$, and let $f : [-1,1]^K \to \mathbb{R}$ be a map which admits a holomorphic complex extension to the Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}}$ with $\boldsymbol{\rho} = (\rho,\dots,\rho) \in (1,\infty)^K$ for some $\rho > 1$. We assume that an approximation $\breve{f}$ of f is available, and an upper bound on $\|f - \breve{f}\|_{L^\infty([-1,1]^K)}$.*

Then, for all $n \in \mathbb{N}$ there exists a ReLU DNN $\Phi_n^{\breve{f}}$ with input dimension K and output dimension one such that for all $\rho' \in (1,\rho)$, there exists a constant $C > 0$ depending on K, ρ and ρ' and a constant $c > 0$ depending on ρ, but not on K and ρ', such that

$$
\begin{aligned}
\left\|f - \Phi_n^{\breve{f}}\right\|_{L^\infty([-1,1]^K)} &\leq C\left(\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})| + \left\|f-\breve{f}\right\|_{L^\infty([-1,1]^K)}\right){\rho'}^{-n}\\
&\quad + \left(\tfrac{2}{\pi}\log(n+1)+1\right)^K\left\|f-\breve{f}\right\|_{L^\infty([-1,1]^K)},\\
\left\|f - \Phi_n^{\breve{f}}\right\|_{W^{1,\infty}([-1,1]^K)} &\leq C\left(\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}}|f(\boldsymbol{z})| + \left\|f-\breve{f}\right\|_{L^\infty([-1,1]^K)}\right){\rho'}^{-n}\\
&\quad + n^2\left(\tfrac{2}{\pi}\log(n+1)+1\right)^K\left\|f-\breve{f}\right\|_{L^\infty([-1,1]^K)},\\
\mathrm{depth}\left(\Phi_n^{\breve{f}}\right) &\leq cKn(1+\log(Kn)),\quad \mathrm{size}\left(\Phi_n^{\breve{f}}\right) \leq cK^2(n+1)^{K+1}.
\end{aligned}
$$

The DNN weights and biases only depend on f through $\{\breve{f}(\boldsymbol{x}_{\boldsymbol{j}}^n)\}_{\boldsymbol{j}\in\{0,\dots,n\}^K}$, where $\boldsymbol{x}_{\boldsymbol{j}}^n = \cos(\boldsymbol{j}\pi/n)$. These are $(n+1)^K$ points.

This means that for C as in Proposition 3.5 and for some $b > 0$ depending on ρ and ρ', but not on K, that for all $n \in \mathbb{N}$ holds

$$\begin{aligned}
&\left\|f - \Phi_n^{\breve{f}}\right\|_{W^{1,\infty}([-1,1]^K)} \\
&\quad \le \rho' C \left(\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}\right) \exp\left(-b\left(\operatorname{size}\left(\Phi_n^{\breve{f}}\right)\right)^{1/(K+1)}\right) \\
&\quad + n^2\left(\tfrac{2}{\pi}\log(n+1)+1\right)^K \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}. \qquad (3.5)
\end{aligned}$$

The L^∞-error satisfies the same estimate, even without the factor n^2 in the second term. Equation (3.5) follows from the error estimate in Proposition 3.5 and from the following estimate, where we assume, without loss of generality, that $c \ge 1$:

$$\begin{aligned}
\log(\rho')(n+1) &\ge \log(\rho') c^{-1/(K+1)} K^{-2/(K+1)} \left(\operatorname{size}\left(\Phi_n^{\breve{f}}\right)\right)^{1/(K+1)} \\
&\ge \log(\rho') c^{-1} c' \left(\operatorname{size}\left(\Phi_n^{\breve{f}}\right)\right)^{1/(K+1)} =: b\left(\operatorname{size}\left(\Phi_n^{\breve{f}}\right)\right)^{1/(K+1)}.
\end{aligned}$$

We also used the existence of $c' > 0$ such that $x^{-2/(x+1)} = \exp(-2\log(x)/(x+1)) \ge c'$ for all $x \in [1,\infty)$.

Proof of Proposition 3.5. We consider the approximation $\breve{p}_{f,n}$ of f from (2.10). For the ReLU DNN approximation of tensor product Chebyšev polynomials, we apply Proposition 3.3 with $\Lambda = \{0,\dots,n\}^K$, $|\Lambda| = (n+1)^K$, $m_\infty(\Lambda) = n$ and $\delta = \rho^{-n}(n+1)^{-K}$.

We write $\boldsymbol{k}^{(\ell)}$ for the element of Λ satisfying $(\Phi_{\Lambda,\delta})_\ell = \tilde{T}_{\boldsymbol{k}^{(\ell)},\delta}$ (the latter is defined in Proposition 3.3). With that notation, we define $A \in \mathbb{R}^{1\times|\Lambda|}$ by $A_{1,\ell} = \breve{f}_{\boldsymbol{k}^{(\ell)};n}$ for $\ell = 1,\dots,|\Lambda|$. With $b := [0] \in \mathbb{R}^1$, we define

$$\Phi_n^{\breve{f}} := ((A,b)) \circ \Phi_{\Lambda,\delta}$$

and observe that $\operatorname{depth}(((A,b))) = 0$ and $\operatorname{size}(((A,b))) \le |\Lambda|$. Its realization satisfies for some $C > 0$ depending on K and ρ

$$\begin{aligned}
\Phi_n^{\breve{f}} &= \sum_{\boldsymbol{k}\in\Lambda} \breve{f}_{\boldsymbol{k};n} \tilde{T}_{\boldsymbol{k},\delta}, \\
&\left\|\breve{p}_{f,n} - \Phi_n^{\breve{f}}\right\|_{W^{1,\infty}([-1,1]^K)} \\
&\quad \le \sum_{\boldsymbol{k}\in\Lambda} |\breve{f}_{\boldsymbol{k};n}| \left\|T_{\boldsymbol{k}} - \tilde{T}_{\boldsymbol{k},\delta}\right\|_{W^{1,\infty}([-1,1]^K)} \\
&\quad \le \delta \sum_{\boldsymbol{k}\in\Lambda} |\breve{f}_{\boldsymbol{k};n}| \\
&\quad \le \delta C \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + \delta(n+1)^K \pi^K \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)} \\
&\quad \le C\rho^{-n}\left(\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}\right), \qquad (3.6)
\end{aligned}$$

where we used the bound (2.15) on the sum of the coefficients. With Lemma 2.2, for some $C > 0$ depending on K and ρ, it further follows that

$$\begin{aligned}
&\left\|f - \Phi_n^{\breve{f}}\right\|_{L^\infty([-1,1]^K)} \\
&\quad \le \|f - \breve{p}_{f,n}\|_{L^\infty([-1,1]^K)} + \left\|\breve{p}_{f,n} - \Phi_n^{\breve{f}}\right\|_{L^\infty([-1,1]^K)} \\
&\quad \le \left(1 + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty}\right) K \left(\tfrac{2\rho}{\rho-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|\, \rho^{-n-1} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)} \\
&\qquad + C\rho^{-n} \left(\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}\right) \\
&\quad \le C \left(\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}\right) \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \rho^{-n} \\
&\qquad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}.
\end{aligned}$$

For all $\rho' \in (1,\rho)$ there exists a constant C depending on ρ, ρ' and K such that $\left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \rho^{-n} \le (\frac{2}{\pi}\log(n+1)+1)^K \rho^{-n} \le C{\rho'}^{-n}$ for all $n \in \mathbb{N}$, resulting in the error bound stated in the proposition. Similarly, for every $\rho' \in (1,\rho)$ and with $C'(\rho,\rho') > 0$ as in Lemma 2.2, with Equation (3.6) it follows that there exists $C(K,\rho,\rho') > 0$ such that

$$\begin{aligned}
&\left\|f - \Phi_n^{\breve{f}}\right\|_{W^{1,\infty}([-1,1]^K)} \\
&\le \|f - \breve{p}_{f,n}\|_{W^{1,\infty}([-1,1]^K)} + \left\|\breve{p}_{f,n} - \Phi_n^{\breve{f}}\right\|_{W^{1,\infty}([-1,1]^K)} \\
&\le \left(1 + \left\|I_n^{\mathrm{CC}}\right\|_{W^{1,\infty},W^{1,\infty}}\right) K \left(\tfrac{2C'\rho'}{\rho'-1}\right)^K \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})|\, {\rho'}^{-n-1} \\
&\quad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{1,\infty}} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)} \\
&\quad + C\rho^{-n} \left(\max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}\right) \\
&\le C{\rho'}^{-n} \left\|I_n^{\mathrm{CC}}\right\|_{W^{1,\infty},W^{1,\infty}} \max_{\boldsymbol{z}\in\mathcal{E}_{\boldsymbol{\rho}}} |f(\boldsymbol{z})| + C\rho^{-n} \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)} \\
&\quad + \left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,W^{1,\infty}} \left\|f - \breve{f}\right\|_{L^\infty([-1,1]^K)}.
\end{aligned}$$

Reasoning as for the bound on the L^∞-error, for all $\rho'' \in (1,\rho')$ there exists a constant $C > 0$ depending on K, ρ, ρ' and ρ'' such that $\left\|I_n^{\mathrm{CC}}\right\|_{L^\infty,L^\infty} \rho^{-n} \le C{\rho''}^{-n}$ and $\left\|I_n^{\mathrm{CC}}\right\|_{W^{1,\infty},W^{1,\infty}} {\rho'}^{-n} \le n^2(\frac{2}{\pi}\log(n)+1)^K {\rho'}^{-n} \le C{\rho''}^{-n}$ for all $n \in \mathbb{N}$. Using these estimates and Equation (2.7), the bound in the proposition is obtained when we write ρ' instead of ρ''. To bound DNN depth and size, we observe that there is $c > 0$ depending on ρ, but not on K, such that for every

$n \in \mathbb{N}$,

$$\begin{aligned}
\operatorname{depth}\left(\Phi_n^{\breve{f}}\right) &\leq \operatorname{depth}(((A,b))) + 1 + \operatorname{depth}(\Phi_{\Lambda,\delta}) \\
&\leq 1 + c(1+\log m_\infty(\Lambda))^3 + c(1+\log(K)+\log m_\infty(\Lambda))\log(1/\delta) \\
&\quad + cK\log(m_\infty(\Lambda)) + cK\log(K) \qquad (3.7) \\
&\leq cKn(1+\log(Kn)), \\
\operatorname{size}\left(\Phi_n^{\breve{f}}\right) &\leq c\operatorname{size}(((A,b))) + c\operatorname{size}(\Phi_{\Lambda,\delta}) \\
&\leq c|\Lambda| + c|\Lambda|K\log(m_\infty(\Lambda)) + c|\Lambda|K\log(1/\delta) + c|\Lambda|K^2 \\
&\quad + cKm_\infty(\Lambda)\log(m_\infty(\Lambda)) + cKm_\infty(\Lambda)\log(1/\delta) + cK^2 m_\infty(\Lambda) \\
&\leq cK^2(n+1)^{K+1}. \qquad (3.8)
\end{aligned}$$

□

Remark 3.6 Vector-valued maps mapping into $\mathbb{R}^N$ for $N \in \mathbb{N}$ can be treated in the same way: we only need to replace the matrix $A \in \mathbb{R}^{1\times|\Lambda|}$ in the proof by a matrix $A \in \mathbb{R}^{N\times|\Lambda|}$. This does not affect the depth, and adds at most $cN(n+1)^K$ to the network size.

Remark 3.7 For $K \in \mathbb{N}$ assume that $f, g : [-1,1]^K \to \mathbb{R}$ satisfy the requirements of Proposition 3.5 for the same $\rho \in (1,\infty)$, with available approximations $\breve{f}$ and $\breve{g}$. For all $n \in \mathbb{N}$, the networks $\Phi_n^{\breve{f}}$ and $\Phi_n^{\breve{g}}$ have the same architecture. In fact, all layers but the output layer are identical, and equal the DNN $\Phi_{\Lambda,\delta}$ from Proposition 3.3, for $\Lambda = \{0,\dots,n\}^K$ and $\delta = \rho^{-n}(n+1)^{-K}$. The weights in the output layer equal $\{\breve{f}_{\boldsymbol{k};n}\}_{\boldsymbol{k}\in\Lambda}$ and $\{\breve{g}_{\boldsymbol{k};n}\}_{\boldsymbol{k}\in\Lambda}$, respectively. These depend linearly on $\breve{f}, \breve{g}$, thus in particular continuously. For example, with Remark 2.5 it follows that

$$\max_{\boldsymbol{k}\in\Lambda} |\breve{f}_{\boldsymbol{k};n} - \breve{g}_{\boldsymbol{k};n}| \leq 2^K \|\breve{f} - \breve{g}\|_{L^\infty([-1,1]^K)}.$$

3.4 More General Activations

The results derived here should extend to DNNs with activation functions other than ReLU. Specifically, the DNNs constructed in our proofs essentially consist of concatenations and parallelizations of identity networks and DNN approximations of products. The proposed ReLU DNN constructions therefore extend to DNNs with other activation functions, as long as they allow an exact emulation or efficient approximation of identity networks and products. We discuss several particular cases.

Poly-ReLU: Exact emulation of both is possible with DNNs with the *rectified power unit* (*RePU*) activation function $\sigma_r : x \mapsto \max\{0,x\}^r$, for $r \in \mathbb{N}$, $r \geq 2$ ([22]). Polynomial emulation with such networks was studied e.g. in [26, Proof of Theorem 3.3], [22]. For DNNs with the RePU activation

function, efficient polynomial emulation based on the recursion in Equation (A.1) was already obtained in [36].

Rational Activation: Exact emulation of the identity and of multiplications is also possible by DNNs with rational activation. In [4], networks were studied which have in each computational node $\sigma = p/q$ as activation, for polynomials p, q of prescribed degrees, but whose coefficients can be trained and may be different for each node. Such networks can emulate both the identity and products exactly if the prescribed degrees satisfy $\deg(p) \geq 2$ and $\deg(q) \in \mathbb{N}_0$ (cf. [4, Proposition 10] and its proof).

Because Lemma 2.2 holds for all $s \in \mathbb{N}$, networks whose activation function allows for the exact emulation of polynomials obtain exponential convergence with respect to the $W^{s,\infty}$-norm for a larger range of $s \in \mathbb{N}$, with only the constant C in the error bound depending on s. This applies in particular to DNNs with Poly-ReLU and rational activation.

Sigmoidal Activation: The present results for RePU networks directly generalize to sigmoidal activation functions of order $k \in \mathbb{N}$, $k \geq 2$. A function $\tau : \mathbb{R} \to \mathbb{R}$ is *sigmoidal of order* $k \in \mathbb{N}_0$, if it satisfies $\lim_{x\to-\infty} \tau(x)/x^k = 0$, $\lim_{x\to\infty} \tau(x)/x^k = 1$, and $|\tau(x)| \leq C(1 + |x|^k)$, $x \in \mathbb{R}$, for a constant $C > 0$ that does not depend on x, see for example [26, Equations (2.3) and (2.4)]. In [26, Lemma 3.6], it was shown that for every continuous sigmoidal activation function τ of order $k \geq 2$ and arbitrary $A > 0$, the RePU σ_k can be approximated on the interval $[-A, A]$ with arbitrarily small error ε with respect to the $L^\infty([-A, A])$-norm by a τ-DNN which has depth 1 and fixed network size independent of A and ε. When τ is uniformly continuous on $\mathbb{R}$, this result remains true w.r.t. the $L^\infty(\mathbb{R})$-norm (instead of $L^\infty([-A, A])$), as remarked directly below [26, Lemma 3.6]. There it was also noted that a similar statement holds for the approximation of the ReLU by a DNN with arbitrary sigmoidal activation function of the order $k = 1$. In the proof of [26, Lemma 3.6], it was observed that for every $k \in \mathbb{N}$ and every continuous, sigmoidal τ of order k, the τ-DNN that approximates σ_k is uniformly continuous on $[-A, A]$. From this, it follows that σ_k-DNNs can be approximated up to arbitrarily small $L^\infty([-A, A])$-error ε by a τ-DNN with network size independent of A and ε. If τ is uniformly continuous on $\mathbb{R}$, the same holds w.r.t. the $L^\infty(\mathbb{R})$-norm.

Genuinely Nonlinear Activation: The existence of DNNs that approximate a product with arbitrarily small $L^\infty([-A, A])$-error ε and network size independent of A and ε, holds also for DNNs with "genuinely nonlinear" activation function $\tau \in C^2(\mathbb{R})$, i.e. for which there exists $x \in \mathbb{R}$ where $\tau''(x) \neq 0$. In [34, Theorem 3.4] it was shown that for all $A, \varepsilon > 0$ there exists a τ-DNN which approximates the product of any two numbers in $[-A, A]$ up to accuracy ε with respect to the $L^\infty([-A, A]^2)$-norm and the network size is bounded independently of A and ε. See also [35, Section 3.3]. The identity can now be approximated based on $x = \frac{1}{4}((x + 1)^2 - (x - 1)^2)$ for all $x \in \mathbb{R}$.

4 DNN Emulation Algorithm

In this section, we develop an algorithm which, assuming at hand a procedure $\breve{f}$ to approximately evaluate a map $f : [-1,1]^K \to \mathbb{R}$, will construct, for a given accuracy threshold $\varepsilon > 0$, an approximating DNN whose estimated error is at most ε and whose depth and size satisfy the theoretical bounds from Section 3. Importantly, the proposed algorithm is deterministic, and does not resort to "black-box" loss function minimization algorithms.

As before, we assume that f admits a holomorphic extension to a closed Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}}$, with $\boldsymbol{\rho} = (\rho, \dots, \rho) \in \mathbb{R}^K$ for some $\rho > 1$. The algorithm is based on the observation that the proof of Proposition 3.5 is constructive, i.e., for $n \in \mathbb{N}$, $\Lambda = \{0, \dots, n\}^K$ and $\delta = (n+1)^{-K}\rho^{-n}$, the network $\Phi_n^{\breve{f}} := ((A,b)) \circ \Phi_{\Lambda,\delta}$ has been described explicitly. In the proof of Proposition 3.5, the accuracy $\delta = (n+1)^{-K}\rho^{-n}$ of the network $\Phi_{\Lambda,\delta}$ approximating tensor product Chebyšev polynomials is sufficient, but may be much smaller than needed, and because a priori no good estimate on ρ may be known, part of the algorithm is dedicated to determining a moderate value of δ which is sufficient.

Thus, assuming access to an approximation $\breve{f}$ of f, the following algorithm determines values $n_* \in \mathbb{N}$ and $\delta_* \in (0,1)$ and constructs a ReLU DNN $\Phi_{n_*,\delta_*}^{\breve{f}}$ approximating f. When determining n_*, we note that in (2.13) the interpolation error $\|f - \breve{p}_{f,n}\|_{L^\infty([-1,1]^K)}$ is bounded by $\|I_n^{\mathrm{CC}}\|_{L^\infty,L^\infty}\|f - \breve{f}\|_{L^\infty([-1,1]^K)}$, growing polylogarithmically in n, plus a term that decays exponentially with n. Because we cannot access f directly, the algorithm below cannot ensure smallness of $\|I_n^{\mathrm{CC}}\|_{L^\infty,L^\infty}\|f - \breve{f}\|_{L^\infty([-1,1]^K)}$. It could be taken into account by setting an upper bound on n, we will not discuss this in detail. The DNN error $\|\breve{p}_{f,n} - \Phi_n^{\breve{f}}\|_{L^\infty([-1,1]^K)}$ is bounded by $\delta \sum_{\boldsymbol{k}\in\Lambda} |\breve{f}_{\boldsymbol{k};n}|$. In the proof of Proposition 3.5, we used that in theory, when neglecting rounding errors in the DNN's affine transformations, this error can be made small with respect to other error terms by choosing δ sufficiently small, namely $\delta = (n+1)^{-K}\rho^{-n}$. In practice, it holds that choosing a smaller δ cannot increase the error (as this holds for the networks in Proposition 3.2, and therefore also for those in Proposition 3.3). However, because the accuracy of affine transformations is limited by the machine precision, δ is in practice limited from below. Therefore, as the upper bound (2.15) on $\sum_{\boldsymbol{k}\in\Lambda} |\breve{f}_{\boldsymbol{k};n}|$ increases with n, the DNN error $\|\breve{p}_{f,n} - \Phi_n^{\breve{f}}\|_{L^\infty([-1,1]^K)}$ may increase with n for large n, regardless of our choice of δ. Therefore, we use a greedy algorithm to determine moderate n_*, δ_* for which the estimated error w.r.t. $\breve{f}$ is smaller than a prescribed tolerance.

The construction in Algorithm 4.1 uses some elementary DNNs, for which explicit constructions are available in the literature [39,35]. In [39, Propositions 2 and 3], a DNN is proposed that approximates the square denoted by $\Phi_{\ell,M}^{\mathrm{sq}}$ and the product of two scalars denoted by $\tilde{\times}_{\ell,M}$ in a bounded interval $[-M,M]$ with ℓ layers (see also [35, Proposition 3.1] for a different construction). In Step 1 of the proof of [35, Proposition 3.3], an explicit DNN is proposed that approximates the product of K scalars (already previously) denoted by $\tilde{\prod}_{\beta',M}^K$ with accuracy β' on the hypercube $[-M,M]^K$. Based on the

scalar product DNN $\tilde{\times}_{\ell,M}$, the DNN $\Phi_\beta^{\mathrm{Cheb},n}$ is constructed in Appendix A. Its properties are given by Lemma 3.1. Recall that this DNN has input dimension one and output dimension n such that the ℓ'th component of the output layer approximates the Chebyšev polynomial of degree ℓ with accuracy β over the interval $[-1,1]$. A DNN that approximates tensorized Chebyšev polynomials is constructed by a concatenation of K copies of the DNN $\Phi_\beta^{\mathrm{Cheb},n}$ with copies of the DNN $\tilde{\prod}_{\beta',M}^K$ and is denoted by $\Phi_{\Lambda,\delta}$, where $\Lambda = \{0,\dots,n\}^K \subset \mathbb{N}_0^K$ is the index set of polynomial degrees of tensorized Chebyšev polynomials. The construction of $\Phi_{\Lambda,\delta}$ is described in detail in Step 1 of the proof of Proposition 3.3. These explicitly constructed DNNs with certified accuracy over all inputs in an a priori known hypercube[1] are used in the following algorithm. Finally, as in the proof of Proposition 3.5, let $b := [0] \in \mathbb{R}^1$ and $A \in \mathbb{R}^{1\times|\Lambda|}$ be defined by $A_{1,\ell} = \breve{f}_{\boldsymbol{k}^{(\ell)};n}$, where $\{\boldsymbol{k}^{(\ell)}\}_{\ell=1,\dots,|\Lambda|} \subset \Lambda$ is an enumeration of Λ such that $(\Phi_{\Lambda,\delta})_\ell = \tilde{T}_{\boldsymbol{k}^{(\ell)},\delta}$ (the latter is defined in Proposition 3.3). With this notation, we define $\Phi_{n,\delta}^{\breve{f}} := ((A,b)) \circ \Phi_{\Lambda,\delta}$ for all $n \in \mathbb{N}$ and $\delta \in (0,1)$.

Regarding line 5 of Algorithm 4.1, recall that $\cos((2n-j)\pi/n) = \cos(j\pi/n)$ for $j = 1,\dots,n-1$, thus $(\breve{f}(\boldsymbol{x}_{\boldsymbol{j}}^n))_{\boldsymbol{j}\in\{0,\dots,n\}^\kappa}$ are sufficient to determine the values $(\breve{f}(\boldsymbol{x}_{\boldsymbol{j}}^n))_{\boldsymbol{j}\in\{0,\dots,2n-1\}^\kappa}$.

Next, we compute the complexity of the algorithm in the setting that has been outlined in this section.

Proposition 4.1 *Let the assumptions of Section 4 and of Algorithm 4.1 be satisfied. In total, under the assumption that the complexity of constructing a DNN from its weights and biases and evaluating a DNN are both proportional to the number of nonzero network weights and biases, the computational complexity of Algorithm 4.1 is bounded by* $C\left[K^2(n_*+1)^{2K+1}\left(\log(1/\delta_*)+\log(n_*)\right)+K^2(n_*+1)^{2K}\log(1/\delta_*)^2\right]$, *in addition to at most* $(n_*+1)^{K+1}$ *evaluations of* $\breve{f}$. *Here, the constant* $C>0$ *does not depend on* n_*, δ_*, K.

The constructed network $\Phi_{n_*,\delta_*}^{\breve{f}}$ *satisfies*

$$\begin{aligned}\operatorname{depth}(\Phi_{n_*,\delta_*}^{\breve{f}}) &\le C\left[1+\log(n_*)^3+\left(\log(K)+\log(n_*)\right)\log(1/\delta_*)+K\log(n_*)\right.\\ &\qquad\left.+K\log(K)\right],\\ \operatorname{size}(\Phi_{n_*,\delta_*}^{\breve{f}}) &\le C\left[1+K^2(n_*+1)^K\left(\log(1/\delta_*)+\log(n_*)\right)\right].\end{aligned}$$

We remark that in practice the input value n_0 in Algorithm 4.1 should be chosen sufficiently large to prevent pathological situations, where Algorithm 4.1 terminates before reaching the desired accuracy. This could for example happen when $\breve{f}$ and $\Phi_{n,\delta}^{\breve{f}}$ are zero in all $x_{\boldsymbol{j}}^{n-1}$ and $x_{\boldsymbol{j}}^n$ but $\breve{f}$ is non-zero as a function. This is possible for certain polynomials $\breve{f}$, which then results in $\Phi_{n,\delta}^{\breve{f}} \equiv 0$. Again, this can be prevented, when choosing n_0 sufficiently large.

[1] The error bounds have been derived under the assumption that the affine transformations in the DNNs are evaluated in exact arithmetic, without rounding.

Algorithm 4.1 Construction of a ReLU DNN that approximates a function f, based on pointwise evaluations of an approximation $\breve{f}$ of f. Suppose that f admits a complex holomorphic extension to $\mathcal{E}_{\boldsymbol{\rho}}$ for $\boldsymbol{\rho} = (\rho, \dots, \rho) \in \mathbb{R}^K$ and (possibly unknown) $\rho > 1$, and that $\breve{f} \in C^0([-1,1]^K)$.

Input: $K \in \mathbb{N}$, a routine to evaluate $\breve{f} : [-1,1]^K \to \mathbb{R}$, a target accuracy $\varepsilon \in (0,1)$, $\alpha \in (0,1)$, $\delta_0 \in (0,1)$, $n_0 \geq 3$

Output: $\Phi^{\breve{f}}_{n_*,\delta_*}$, for $n_* \in \mathbb{N}$ and $\delta_* \in (0,1)$ to be determined.

1: Set $n \leftarrow n_0$, $\delta \leftarrow \delta_0$, err $\leftarrow 1$, and state $\leftarrow$ true
2: **while** err $> \varepsilon$ **do**
3: Set err$_{-1} \leftarrow$ err.
4: Compute $\{\breve{f}(\boldsymbol{x}^n_{\boldsymbol{j}})\}_{\boldsymbol{j}\in\{0,\dots,n\}^K}$. ▷ $\boldsymbol{x}^n_{\boldsymbol{j}} = \cos(\boldsymbol{j}\pi/n)$
5: Compute $\{\breve{f}_{\boldsymbol{k};n}\}_{\boldsymbol{k}\in\{0,\dots,n\}^K}$ with (2.9). Store as $((A,b))$, as in the proof of Proposition 3.5.
6: Construct $\Phi_{\Lambda,\delta}$, depending on n and δ with $\Lambda = \{0,\dots,n\}^K$.
7: Construct $\Phi^{\breve{f}}_{n,\delta} = ((A,b)) \circ \Phi_{\Lambda,\delta}$ with $\Lambda = \{0,\dots,n\}^K$.
8: Update err $\leftarrow \max_{x\in\{\boldsymbol{x}^{n-1}_{\boldsymbol{j}}:\boldsymbol{j}\in\{0,\dots,n-1\}^K\}} |\breve{f}(x) - \Phi^{\breve{f}}_{n,\delta}(x)|$.
▷ $\boldsymbol{x}^{n-1}_{\boldsymbol{j}} = \cos(\boldsymbol{j}\pi/(n-1))$
9: **if** err $> \alpha$err$_{-1}$ **then**
10: state $\leftarrow$ complement(state).
11: **end if**
12: **if** state $=$ true **then**
13: $n \leftarrow n+1$.
14: **else**
15: $\delta \leftarrow \delta/2$.
16: **end if**
17: **end while**
18: **if** state $=$ true **then**
19: $n_* \leftarrow n-1$ and $\delta_* \leftarrow \delta$.
20: **else**
21: $n_* \leftarrow n$ and $\delta_* \leftarrow 2\delta$.
22: **end if**
23: **return** $\Phi^{\breve{f}}_{n_*,\delta_*}$.

Proof Let $C > 0$ denote some generic constant. The while loop of lines 2–17 is executed at most $n_* - n_0 + 1 + \log_2(\delta_0/\delta_*)$ times: after the initial execution of the while loop, it is executed $n_* - n_0$ times after increasing n and $\log_2(\delta_0/\delta_*)$ times after decreasing δ.

Per execution of the while loop: Line 4 takes $(n+1)^K$ evaluations of $\breve{f}$, and is executed for $n = n_0, \dots, n_*$, i.e. only when state $=$ true. Line 5 computes the IFFT of a K-dimensional array of size $(n+1)^K$. The complexity of each execution is $CK(n+1)^K \log(n+1) = C(n+1)^K \log((n+1)^K)$. Line 6 contains the construction of $\Phi_{\Lambda,\delta}$, which has computational cost $CK^2(n+1)^K(\log(1/\delta) + \log(n))$ (Proposition 3.3). Line 8 requires n^K evaluations of $\breve{f}$ and of $\Phi^{\breve{f}}_{n,\delta}$, where each evaluation of $\Phi^{\breve{f}}_{n,\delta}$ has complexity $CK^2(n+1)^K(\log(1/\delta) + \log(n))$. The evaluations of $\breve{f}$ are the same as in line 4 (except for $n = n_0$, which adds n_0^K evaluations of $\breve{f}$), the evaluation of $\Phi^{\breve{f}}_{n,\delta}$ contributes $CK^2(n+1)^{2K}(\log(1/\delta) + \log(n))$ for each execution of line 8.

In total, the computational complexity is upper bounded by $(n_*+1)^{K+1}$ evaluations of $\breve{f}$ in addition to $CK^2(n_*+1)^{2K+1}\,(\log(1/\delta_*)+\log(n_*))+CK^2(n_*+1)^{2K}\log(1/\delta_*)^2$, with C depending on δ_0 and n_0.

The bounds on the network depth and size follow from Proposition 3.3 and

$$\begin{aligned}\operatorname{depth}\left(\Phi^{\breve{f}}_{n_*,\delta_*}\right) &\leq \operatorname{depth}(((A,b))) + 1 + \operatorname{depth}(\Phi_{\Lambda,\delta}),\\ \operatorname{size}\left(\Phi^{\breve{f}}_{n_*,\delta_*}\right) &\leq C\operatorname{size}(((A,b))) + C\operatorname{size}(\Phi_{\Lambda,\delta}).\end{aligned}$$

If the first order partial derivatives of $\breve{f}$ can be evaluated as well, replacing line 8 by

$$\begin{aligned}\text{Update err} \leftarrow \max_{x\in\{\boldsymbol{x}_{\boldsymbol{j}}^{n-1}:\boldsymbol{j}\in\{0,\dots,n-1\}^K\}} \max\Big\{&|\breve{f}(x)-\Phi^{\breve{f}}_{n,\delta}(x)|,\\ &\max_{i=1}^K \left|\tfrac{\partial}{\partial x_i}\breve{f}(x)-\tfrac{\partial}{\partial x_i}\Phi^{\breve{f}}_{n,\delta}(x)\right|\Big\}\end{aligned}$$

will give a similar result, now with $W^{1,\infty}$-error at most ε.

5 DNN Data-to-QoI Maps in Bayesian Inversion

In the present section, we illustrate the foregoing results, including the exponential convergence, see Proposition 3.5, in the context of PDE constrained Bayesian estimation.

Let X be a separable Banach space and π_0 be a *prior* probability measure on X. The expectation with respect to π_0 is denoted by $\mathbb{E}^{\pi_0}$. Let $\mathcal{X}$ be a separable Banach space and let *the forward map* $\mathcal{S}: X\to\mathcal{X}$ be continuous. For given $u\in X$, we suppose that we can measure the *response* $\mathcal{S}(u)$ with observation functionals $O_i\in\mathcal{X}^*$, $i=1,\dots,K$, where $\mathcal{X}^*$ denotes the dual space of $\mathcal{X}$. We aim at a reconstruction of u from these measurements $O_i(\mathcal{S}(u))$, $i=1,\dots,K$. In general this problem is ill-posed (e.g. [7,20]). We consider additive Gaussian observation noise which regularizes the Bayesian inverse problem [7], i.e., we assume noisy observation data δ_i, $i=1,\dots,K$, such that

$$\delta = O(\mathcal{S}(u))+\eta,$$

where η is assumed to be an additive, centered, independent K-dimensional multivariate Gaussian random variable with nondegenerate covariance matrix Σ. We use the notation $\delta=(\delta_1,\dots,\delta_K)$ and $O=(O_1,\dots,O_K)$. The Bayesian estimate of a quantity of interest $\phi: X\to\mathbb{R}$, which is assumed to be continuous, is given by $\mathbb{E}^{\pi_\delta}[\phi(u)]$. Here, π_δ denotes the posterior probability conditional on given data δ.

In practice the computation of an approximation to this Bayesian estimate could be costly, in particular when the forward map may involve a solver of a partial differential equation. For the repeated efficient approximate numerical evaluation of the data-to-QoI map $\delta\mapsto\mathbb{E}^{\pi_\delta}[\phi(u)]$, surrogate maps of DNN type can lead to large computational savings. These maps can be evaluated

rapidly in comparison to the pointwise, approximate evaluation of this map using, e.g., MCMC algorithms requiring one PDE solve for each proposal.

With the presently developed, explicit, DNN emulators, we are able to explicitly construct a surrogate for the data-to-QoI map by means of a ReLU DNN. The exponential error bound on the generalization error will imply that few *parallel evaluations of the forward map, at synthetic data points,* are required. This is due to the additive Gaussian noise η having a strong regularizing effect. This has been pointed out and was exploited in [18]. There in [18, Corollary 4.7] it is shown that for any $r > 0$, the map

$$[-r, r]^K \ni \delta \mapsto f(\delta) := \mathbb{E}^{\pi_\delta}[\phi(u)] \tag{5.1}$$

is analytic. This allows quite general forward maps $\mathcal{S}$, which are not required to be smooth, see [18, Section 3] for a few examples. Thus, the map $[-r, r]^K \ni \delta \mapsto f(\delta)$ satisfies the assumptions of Proposition 3.5 and Algorithm 4.1 constructs a ReLU DNN with size and depth as in Proposition 4.1, if we are able to approximate the map f by some $\check{f}$ sufficiently well. In this example the modulus of the considered mapping f depends exponentially on the smallest eigenvalue of the covariance matrix Σ of the additive noise η. This term consequentially appears on the right-hand side of our error estimates.

6 Numerical Experiments

We consider the solution q to an elliptic partial differential equation given a realization of an uncertain diffusion coefficient function u as forward operator. Specifically, on a bounded Lipschitz domain $D \subset \mathbb{R}^d$ let

$$-\nabla \cdot (u \nabla q) = g \quad q|_{\partial D} = 0, \tag{6.1}$$

where $g \in L^2(D)$ is assumed known and where $u \in \{\tilde{u} \in L^\infty(D) : \operatorname{ess\,inf}_{x \in D} \tilde{u}(x) > 0\}$. For every such coefficient function u, the solution $q \in H^1_0(D) =: \mathcal{X}$ exists and is unique by the Lax–Milgram lemma. The forward coefficient-to-solution map $\mathcal{S}$ is defined by $\mathcal{S}(u) := q$. The observation operators O_i, $i = 1, \ldots, K$, are taken as linear functionals in $H^{-1}(D) := (H^1_0(D))^*$. We construct an uncertain diffusion coefficient

$$u(\boldsymbol{y}) = \bar{u} + \sum_{j=1}^{s} y_j \psi_j,$$

where $s \in \mathbb{N}$, $\boldsymbol{y} \in [-1/2, 1/2]^s$ and $\operatorname{ess\,inf}_{x \in D}\{\bar{u}(x)\} > \sum_{j=1}^{s} \|\psi_j\|_{L^\infty(D)}/2$. We assume that $\psi_j \in L^\infty(D)$, $j = 1, \ldots, s$. We model the Bayesian prior as product probability measure μ on $[-1/2, 1/2]^s$ by $\mu(\mathrm{d}\boldsymbol{y}) = \bigotimes_{j=1}^{s} \mathrm{d}y_j$. The prior π_0 is chosen as the pushforward measure of the random coefficient $\boldsymbol{y} \mapsto u(\boldsymbol{y})$ on $X = \{\bar{u} + \sum_{i=1}^{s} y_j \psi_j : \boldsymbol{y} \in [-1/2, 1/2]^s\}$, i.e., $\pi_0(A) := \mu(u^{-1}(A))$ for every measurable set $A \subset X$, where $u^{-1}(A)$ denotes the pre-image under the mapping $u : [-1/2, 1/2]^s \to X$. Let $\varphi \in H^{-1}(D)$. Then the QoI ϕ is given by

the composition of φ and $\mathcal{S}$, i.e., $\phi = \varphi \circ \mathcal{S}$. According to [7, Theorem 14], the posterior expectation in (5.1) is given by

$$f(\delta) = \frac{1}{Z} \int_X \phi(u) \exp\left(-\frac{(\delta - O(\mathcal{S}(u)))^\top \Sigma^{-1} (\delta - O(\mathcal{S}(u)))}{2} \right) \pi_0(\mathrm{d}u) \quad (6.2)$$

and

$$Z = \int_X \exp\left(-\frac{(\delta - O(\mathcal{S}(u)))^\top \Sigma^{-1} (\delta - O(\mathcal{S}(u)))}{2} \right) \pi_0(\mathrm{d}u). \quad (6.3)$$

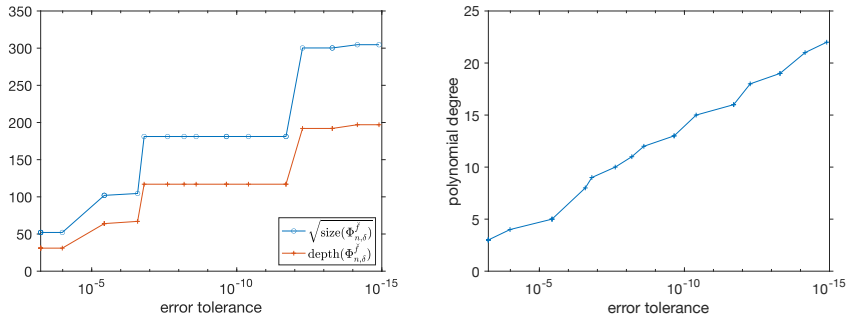
In our computations, we consider the univariate case and one or two parameters, i.e., $d = 1$, $D = (-1, 1)$, and $s = 1, 2$. Also the data space dimension is chosen to be one, i.e., $K = 1$. For given $u(y)$, the solution $q(y) = \mathcal{S}(u(y))$ is approximated by the Finite Element Method (FEM) with 65 degrees of freedom using a hat functions basis and the integrals in (6.2) and in (6.3) are efficiently approximated by a product Gauss–Legendre quadrature with 40 quadrature nodes in each parameter direction. This allows our numerical tests to focus on the algorithmic realization of ReLU DNNs to approximate the data-to-QoI map $[-1/2, 1/2]^s \ni \delta \mapsto f(\delta)$.

In our setup, we choose $\bar{u} = 1.5$, $\psi_j(x) = 2j^{-2}\cos(j\pi x)$. The observation functional is taken to be $\mathcal{O}(v) := \int_{-1}^{0} v(x)\mathrm{d}x$ and the goal functional is taken as $\varphi(v) := \int_0^1 v(x)\mathrm{d}x$ for every $v \in H_0^1(D)$. The right-hand side in (6.1) is taken as $g = 1$. We ran Algorithm 4.1 for given error tolerances and observed the resulting size of the constructed DNN. Specifically, we ran an implementation of Algorithm 4.1 for the data-to-QoI map $\delta \to f(\delta)$ given in (6.2).

In Figures 6.1 and 6.2, we visualize the output of Algorithm 4.1 with the choice $n_0 = 3$. The size of the DNN that is constructed in Algorithm 4.1, the depth, and the achieved accuracy are plotted in a semi-logarithmic scale. In the figures the error should impact the size with the square of the logarithm according to Proposition 3.5, which is why the square root of the size of the DNN is plotted. As described in Algorithm 4.1 line 8, the error is estimated by the maximum of the absolute value of the difference of the DNN approximation and the data-to-QoI map evaluated in Clenshaw–Curtis points of one degree less than used as training points. Note that Clenshaw–Curtis points are not nested. Because we cannot evaluate f exactly, the error was computed with respect to the previously described, quadrature based numerical approximation, corresponding to $\breve{f}$ in the notation of the algorithm. We tested this setting for additive Gaussian noise with variance $\sigma^2 = 0.04$, see (5.1). The step-like behavior of the size and depth of the constructed DNN that is visible in Figures 6.1 and 6.2 is a consequence of the employed construction of DNNs approximating Chebyšev polynomials, see Appendix A. There, for given polynomial degree $p \in \mathbb{N}$ and $k \in \mathbb{N}$ such that $2^{k-1} < p \leq 2^k$, DNNs approximating univariate Chebyšev polynomials up to degree 2^k are constructed, see also Remark A.1.

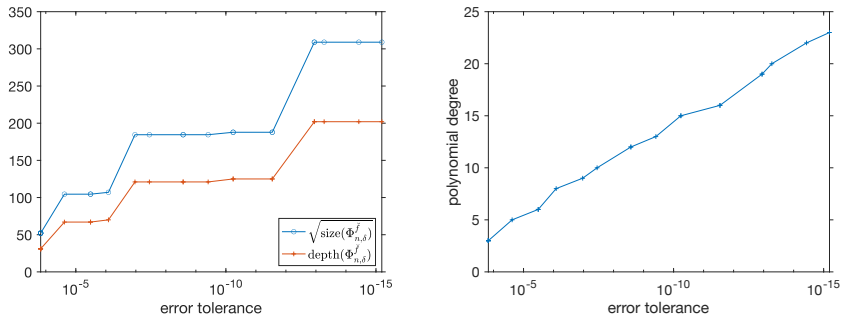
We remark that for large values of s the Gauss–Legendre quadrature that is used to compute the training data is no longer feasible as the number of quadrature points would depend exponentially on s. However, for these cases

high-dimensional quadratures such as quasi-Monte Carlo or Smolyak are applicable and the convergence is of high order and independent of the dimension s [9,10,19,40].



(a) (b)

Fig. 6.1: Accuracy of the DNNs constructed by Algorithm 4.1 vs. (a) size and depth and (b) polynomial degree n. The data-to-QoI map $\delta \mapsto f(\delta)$ from (6.2) is approximated by constructed DNNs for $s = 1$ and $\sigma^2 = 0.04$. The number of evaluations of $\breve{f}$ in Algorithm 4.1 is bounded by $(n+1)^2$, see Proposition 4.1, which is the amount of training data that is used.



(a) (b)

Fig. 6.2: Accuracy of the DNNs constructed by Algorithm 4.1 vs. (a) size and depth and (b) polynomial degree n. The data-to-QoI map $\delta \mapsto f(\delta)$ from (6.2) is approximated by constructed DNNs for $s = 2$ and $\sigma^2 = 0.04$. The number of evaluations of $\breve{f}$ in Algorithm 4.1 is bounded by $(n+1)^2$, see Proposition 4.1, which is the amount of training data that is used.

7 Conclusions

We proved approximation results resp. expressive power bounds for DNNs approximating multivariate, analytic functions, and introduced a novel, deterministic, constructive approach for generating ReLU DNN surrogates. Unlike the (constructive) proofs of DNN emulation in [30], the presently proposed argument and algorithm depends in an essential way on the use of multivariate Chebyšev interpolants of the map f which are to be emulated by DNNs. On the one hand, while retaining exponential convergence for multivariate holomorphic functions, Chebyšev polynomials are well-known to have favorable algorithmic and stability properties. In particular, through a close link with discrete Fourier transformation, Chebyšev interpolants of multivariate functions $f : [-1,1]^K \to \mathbb{R}$ can be conveniently computed through function *collocation on $(n+1)^K$-point tensorized Clenshaw–Curtis grids*.

The good (logarithmic w.r.to the polynomial degree n) stability of interpolation in Clenshaw–Curtis points also allowed us to perform an error analysis for interpolation based on *"noisy", corrupted function evaluations*. We developed bounds for the DNN expression error of the type $C\exp(-bM^{1/(K+1)})$ in terms of the network size M, for some constants $C > 0$ independent of M and $b > 0$ independent of K and M, under the provision that exact function evaluations of f in Clenshaw–Curtis points are used. The constructed DNN is stable with respect to errors in the training data. If the function values of f in the Clenshaw–Curtis points are approximated with a possibly corrupted, numerical approximation $\breve{f}$ of f, then the additional term in the L^∞-error is bounded by $C(\log(\log(1/\varepsilon)))^K \|f - \breve{f}\|_{L^\infty([-1,1]^K)}$, where ε denotes the error of polynomial interpolation (for which it holds $n \propto \log(1/\varepsilon)$ in Proposition 3.5). The additional error in $W^{1,\infty}$-norm is bounded by $C(\log(1/\varepsilon))^2(\log(\log(1/\varepsilon)))^K \|f - \breve{f}\|_{L^\infty([-1,1]^K)}$.

The presently proposed DNN construction does not require loss function minimization, and is not prone to the difficulties encountered in [1] with DNN training via optimization techniques. Our procedure reproduces the polynomial best approximation rates up to logarithmic factors. Furthermore, note that when larger function classes are considered, which can still be approximated with some rate by DNNs, *any algorithm based on point evaluations of functions at best only realizes a significantly smaller rate*. This has been proved in [16]. However, for analytic functions, continuously differentiable functions, or Lipschitz continuous functions, this is not the case as we prove in the present paper for analytic functions. As pointed out in Section 1.2, the proofs are also applicable to Lipschitz continuous functions. Moreover, our methodology extends to any method that is based on constructive polynomial approximation, with the polynomials subsequently replaced by DNNs that emulate them. Potential candidates are least squares (LSQ) and compressed sensing (CS) techniques. See, for example, [1] for studies of CS in the context of DNN approximations. In LSQ polynomial regression, guarantees hold only in the root mean squared sense. Polynomial fitting via CS only affords guarantees with *high probability*, whereas the presently developed bounds hold with certainty.

The use of DNN emulations of Chebyšev polynomials also resolves numerical stability issues of DNNs approximating polynomials recently highlighted in [1, p. 650 bottom].

In this paper, we applied the constructive algorithm to an example in the context of Bayesian PDE inversion, where under the assumption of additive Gaussian observation noise the data-to-quantity of interest map is analytic, cf. [18]. Again, the algorithm is applicable much more generally, e.g. to any Lipschitz continuous map.

A Constructive ReLU DNN Approximation of T_n

We present an emulation of univariate Chebyšev polynomials $T_n(x)$ of arbitrary degrees by ReLU DNNs, which will be developed in [28] and which follows closely the construction in [29] for univariate monomials. Specifically, we construct a DNN that approximates the Chebyšev polynomials of the first kind, denoted by $\{T_\ell\}_{\ell\in\mathbb{N}_0}$. As was derived for DNNs with the RePU activation function $\sigma_r(x) = (\max\{x, 0\})^r$ for $r \in \mathbb{N}$ satisfying $r \geq 2$ in [36], they can be approximated efficiently also by ReLU DNNs by exploiting the three term recursion

$$\forall m, n \in \mathbb{N}_0 : \quad T_{m+n} = 2T_m T_n - T_{|m-n|}, \qquad T_0(x) = 1, \quad T_1(x) = x, \quad \forall x \in \mathbb{R}. \tag{A.1}$$

This recursion is specific to the T_n and follows from the addition rule for cosines.

To construct the DNNs that approximate all Chebyšev polynomials of degree $1, \dots, n$ on $\hat{I} := (-1, 1)$, we first construct inductively, for all $k \in \mathbb{N}$, DNNs $\{\Psi_\delta^k\}_{\delta\in(0,1)}$ with input dimension one and output dimension $2^{k-1} + 2$ with the following properties: denoting all components of the output, except for the first one, by $\tilde{T}_{\ell,\delta} := (\Psi_\delta^k)_{2+\ell-2^{k-1}}$ for $\ell \in \{2^{k-1}, \dots, 2^k\}$, it holds that

$$\begin{aligned} \Psi_\delta^k(x) &= \big(x, \tilde{T}_{2^{k-1},\delta}(x), \dots, \tilde{T}_{2^k,\delta}(x)\big), \quad x \in \hat{I}, \\ \Big\|T_\ell(x) - \tilde{T}_{\ell,\delta}(x)\Big\|_{W^{1,\infty}(\hat{I})} &\leq \delta, \quad \ell \in \{2^{k-1}, \dots, 2^k\}. \end{aligned} \tag{A.2}$$

We only provide the DNN constructions, for proofs of the error bound and the estimates on the network depth and size in Lemma 3.1 we refer to [28]. For brevity, we will denote the weights of a DNN layer by a matrix, similarly to the notation used in [28]. We say that $A \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ *are the weights of layer* $\ell \in \{1, \dots, L+1\}$ if $A_{ji} = A_{i,j}^\ell$ in the notation of Section 3.1.

Induction basis. Let $\delta \in (0, 1)$ be arbitrary and define $L_1 := \mathrm{depth}(\tilde{\prod}_{\delta/4,1}^2)$. Also, define the matrix $A := [1, 1]^\top \in \mathbb{R}^{2\times 1}$ and the vector $b' := [-1] \in \mathbb{R}^1$, and let A_i, b_i, $i = 1, \dots, L_1 + 1$ denote the weights and biases of $\tilde{\prod}_{\delta/4,1}^2$ as in Proposition 3.2. Then we define

$$\Psi_\delta^1 := \left(\Phi_{1,L_1}^{\mathrm{Id}}, \Phi_{1,L_1}^{\mathrm{Id}}, \Phi\right),$$

where the weights and biases of Φ are $A_1 A, A_2, \dots, A_{L_1}, 2A_{L_1+1}$ resp. $b_1, \dots, b_{L_1}, 2b_{L_1+1} + b'$. It follows that $(\Psi_\delta^1(x))_1 = x$, $\tilde{T}_{1,\delta}(x) := (\Psi_\delta^1(x))_2 = x = T_1(x)$ and $\tilde{T}_{2,\delta}(x) := (\Psi_\delta^1(x))_3 = 2(\tilde{\prod}_{\delta/4,1}^2)(x, x) - 1$ for all $x \in \hat{I} := (-1, 1)$.

Induction hypothesis (IH). For all $\delta \in (0, 1)$ and $k \in \mathbb{N}$, let $\theta := 2^{-2k-4}\delta$, and assume that there exists a DNN Ψ_θ^k which satisfies Equation (A.2) with θ instead of δ.

Induction step. For δ and k as in (IH), we show that (A.2) holds with δ as in (IH) and with $k + 1$ instead of k. We define, for $\Phi^{1,k}$ and $\Phi_\delta^{2,k}$ introduced below,

$$\Psi_\delta^{k+1} := \Phi_\delta^{2,k} \circ \Phi^{1,k} \circ \Psi_\theta^k. \tag{A.3}$$

For a sketch of the network structure, see Figure A.1. The DNN $\Phi^{1,k}$ of depth 0 implements the linear map

$$\mathbb{R}^{2^{k-1}+2} \to \mathbb{R}^{2^{k+1}+2} : (z_1, \dots, z_{2^{k-1}+2}) \mapsto (z_1, z_{2^{k-1}+2}, z_2, z_3, z_3, z_3, z_3, z_4, z_4, z_4, z_4, z_5, \\ \dots, z_{2^{k-1}+1}, z_{2^{k-1}+2}, z_{2^{k-1}+2}, z_{2^{k-1}+2}).$$

Denoting its weights and biases by $A^{1,k}, b^{1,k}$, it holds that $b^{1,k} = 0$ and

$$(A^{1,k})_{m,i} = \begin{cases} 1 & \text{if } m = 1, i = 1, \\ 1 & \text{if } m = 2, i = 2^{k-1} + 2, \\ 1 & \text{if } m \in \{3, \dots, 2^{k+1} + 2\}, i = \lceil \frac{m+5}{4} \rceil, \\ 0 & \text{else.} \end{cases}$$

With $L_\theta := \operatorname{depth}(\tilde{\prod}^2_{\theta,2})$ we define

$$\Phi^{2,k}_\delta := \Phi \circ \left(\Phi^{\mathrm{Id}}_{2,L_\theta}, \tilde{\prod}^2_{\theta,2}, \dots, \tilde{\prod}^2_{\theta,2} \right)_{\mathrm{d}},$$

containing 2^k $\tilde{\prod}^2_{\theta,2}$-networks, with Φ denoting the depth 0 network with weights and biases $A^{2,k} \in \mathbb{R}^{(2^k+2)\times(2^k+2)}$ and $b^{2,k} \in \mathbb{R}^{2^k+2}$ defined as

$$(A^{2,k})_{m,i} := \begin{cases} 1 & \text{if } m = i \le 2, \\ 2 & \text{if } m = i \ge 3, \\ -1 & \text{if } m \ge 3 \text{ is odd}, i = 1, \\ 0 & \text{else,} \end{cases} \qquad (b^{2,k})_m = \begin{cases} -1 & \text{if } m \ge 3 \text{ is even,} \\ 0 & \text{else.} \end{cases}$$

The network Ψ^{k+1}_δ defined in Equation (A.3) realizes

$$(\Psi^{k+1}_\delta(x))_1 = x, \qquad \text{for } x \in \hat{I}, \tag{A.4}$$

$$(\Psi^{k+1}_\delta(x))_2 = \tilde{T}_{2^k,\theta}(x), \qquad \text{for } x \in \hat{I}, \tag{A.5}$$

$$(\Psi^{k+1}_\delta(x))_{\ell+2-2^k} = 2\tilde{\prod}^2_{\theta,2}\left(\tilde{T}_{\lceil \ell/2 \rceil,\theta}(x), \tilde{T}_{\lfloor \ell/2 \rfloor,\theta}(x)\right) - x^{\lceil \ell/2 \rceil - \lfloor \ell/2 \rfloor}, \\ \text{for } x \in \hat{I} \text{ and } \ell \in \{2^k + 1, \dots, 2^{k+1}\}, \tag{A.6}$$

where $x^{\lceil \ell/2 \rceil - \lfloor \ell/2 \rfloor} = x = T_1(x)$ for odd ℓ and $x^{\lceil \ell/2 \rceil - \lfloor \ell/2 \rfloor} = 1 = T_0(x)$ for even ℓ. For $\ell \in \{2^k + 1, \dots, 2^{k+1}\}$ and $x \in \hat{I}$, the right-hand side of (A.6) will be denoted by

$$\tilde{T}_{\ell,\delta}(x) := (\Psi^{k+1}_\delta(x))_{\ell+2-2^k}.$$

This finishes the construction of Ψ^{k+1}_δ.

Next, we construct $\Phi^{\mathrm{Cheb},n}_\delta$ as in Lemma 3.1.

If $n = 1$, for all $\delta \in (0,1)$ we define $\Phi^{\mathrm{Cheb},n}_\delta := ((A, b))$, where $A := [1] \in \mathbb{R}^{1\times 1}$ and $b := [0] \in \mathbb{R}^1$.

If $n \ge 2$, let $k := \lceil \log_2(n) \rceil$. See Figure A.2 for a sketch of the DNN construction below. We use the networks $\{\Psi^j_\delta\}_{\delta\in(0,1), j\in\{1,\dots,k\}}$ constructed above and take $\{\ell_j\}_{j=1}^k \subset \mathbb{N}$ such that $\operatorname{depth}\left(\Psi^k_\delta\right) + 1 = \operatorname{depth}\left(\Psi^j_\delta\right) + \ell_j$ for $j = 1, \dots, k$, and thus $\ell_j \le \max_{j=1}^k \operatorname{depth}\left(\Psi^j_\delta\right) = \operatorname{depth}\left(\Psi^k_\delta\right)$. We define

$$\Phi^{\mathrm{Cheb},n}_\delta := \Phi^{3,n} \circ \left(\Psi^1_\delta \circ \Phi^{\mathrm{Id}}_{1,\ell_1}, \dots, \Psi^k_\delta \circ \Phi^{\mathrm{Id}}_{1,\ell_k}\right),$$

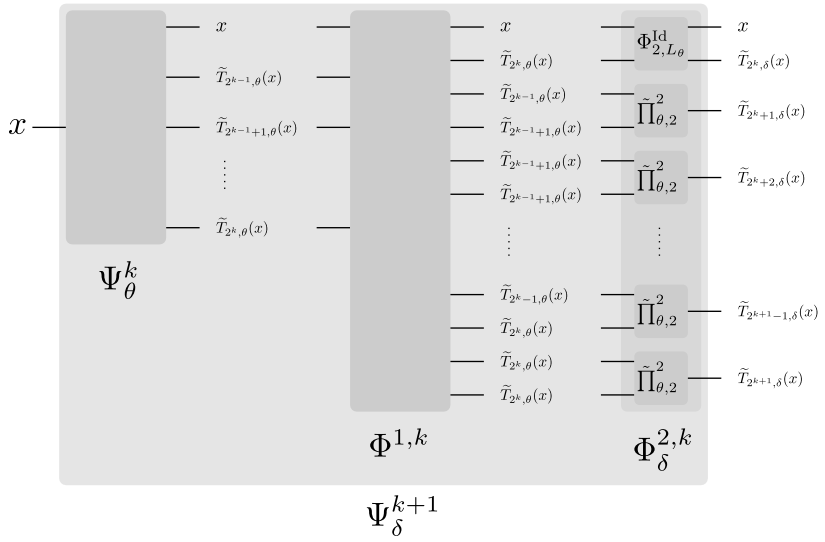


Fig. A.1: Sketch of Ψ_δ^{k+1} for some $k \in \mathbb{N}$ and $\delta \in (0,1)$, inductively constructed from Ψ_θ^k with $\theta = 2^{-2k-4}\delta$. The subnetwork $\Phi^{1,k}$ realizes a linear map, correctly coupling the output of Ψ_θ^k to the input of $\Phi_\delta^{2,k}$. The subnetwork $\Phi_\delta^{2,k}$ acts as the identity on the first two inputs, and as an approximate multiplication from Proposition 3.2 on pairs of the remaining inputs. Output are the input x and approximations of Chebyšev polynomials of degree $2^k, \dots, 2^{k+1}$, with accuracy δ.

where the DNN $\Phi^{3,n}$ of depth 0 emulates the linear map $\mathbb{R}^{2^k+2k-1} \to \mathbb{R}^n$ satisfying

$$\Phi^{3,n}(z_1,\dots,z_{2^k+2k-1})_1 = z_2, \qquad \Phi^{3,n}(z_1,\dots,z_{2^k+2k-1})_2 = z_3,$$
$$\text{and for all } \ell = 3,\dots,n, \text{ with } j := \lceil \log_2(\ell) \rceil : \qquad \Phi^{3,n}(z_1,\dots,z_{2^k+2k-1})_\ell = z_{\ell+2j-1}.$$

The realization satisfies for all $\ell = 1,\dots,n$

$$(\Phi_\delta^{\mathrm{Cheb},n}(x))_\ell = \tilde{T}_{\ell,\delta}(x), \qquad x \in \hat{I},\ \ell \in \{1,\dots,n\}.$$

Remark A.1 The subnetwork Ψ_δ^k of $\Phi_\delta^{\mathrm{Cheb},n}$ approximates all univariate Chebyšev polynomials of degree up to 2^k, also when $n < 2^k$. This causes the "step-like" behavior in Figures 6.1 and 6.2. This step-wise growth of the network size can easily be prevented by removing from Ψ_δ^k the product networks $\tilde{\prod}_{\theta,2}^2$ that compute $\tilde{T}_{\ell,\delta}(x)$ for $\ell > n$, and modifying $\Phi^{3,n}$ accordingly.

References

1. B. Adcock and N. Dexter. The gap between theory and practice in function approximation with deep neural networks. *SIAM J. Math. Data Sci.*, 3(2):624–655, 2021.
2. S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. Solving inverse problems using data-driven models. *Acta Numer.*, 28:1–174, 2019.

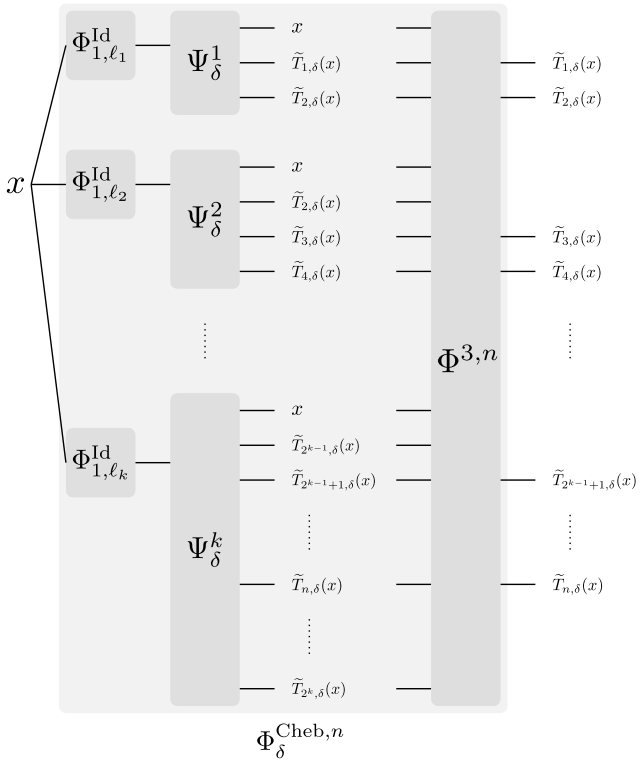


Fig. A.2: Sketch of $\Phi^{\mathrm{Cheb},n}_\delta$ for some $n \in \mathbb{N}$ and $\delta \in (0,1)$, constructed from identity networks and the previously described $\Psi^1_\delta, \dots, \Psi^k_\delta$, with $k := \lceil \log_2(n) \rceil$. The subnetwork $\Phi^{3,n}$ realizes a linear map that selects the desired approximations of univariate Chebyšev polynomials from the outputs of the preceding layer.

3. H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM J. Math. Data Sci.*, 1(1):8–45, 2019.
4. N. Boull, Y. Nakatsukasa, and A. Townsend. Rational neural networks, 2020. Accepted for publication in *34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.*
5. P. Cheridito, A. Jentzen, and F. Rossmannek. Non-convergence of stochastic gradient descent in the training of deep neural networks. *J. Complexity*, 64:Paper No. 101540, 10, 2021.
6. A. Cohen, C. Schwab, and J. Zech. Shape holomorphy of the stationary Navier-Stokes Equations. *SIAM J. Math. Analysis*, 50(2):1720–1752, 2018.
7. M. Dashti and A. M. Stuart. The Bayesian approach to inverse problems. In *Handbook of uncertainty quantification. Vol. 1, 2, 3*, pages 311–428. Springer, Cham, 2017.
8. J. Daws and C. Webster. Analysis of deep neural networks with quasi-optimal polynomial approximation rates, 2019. ArXiv: 1912.02302.
9. J. Dick, R. N. Gantner, Q. T. L. Gia, and C. Schwab. Multilevel higher-order quasi-Monte Carlo Bayesian estimation. *Math. Models Methods Appl. Sci.*, 27(5):953–995, 2017.
10. J. Dick, R. N. Gantner, Q. T. L. Gia, and C. Schwab. Higher order quasi-Monte Carlo integration for Bayesian PDE inversion. *Comput. Math. Appl.*, 77(1):144–172, 2019.

11. D. Dng and V. K. Nguyen. Deep ReLU neural networks in high-dimensional approximation. *Neural Networks*, 142:619–635, 2021.
12. W. E and Q. Wang. Exponential convergence of the deep neural network approximation for analytic functions. *Sci. China Math.*, 61(10):1733–1740, 2018.
13. H. Ehlich and K. Zeller. Auswertung der Normen von Interpolationsoperatoren. *Math. Ann.*, 164:105–112, 1966.
14. D. Elbrächter, P. Grohs, A. Jentzen, and C. Schwab. DNN Expression Rate Analysis of High-dimensional PDEs: Application to Option Pricing. *Constructive Approximation.* published online May 6th, 2021.
15. M. Gaß, K. Glau, M. Mahlstedt, and M. Mair. Chebyshev interpolation for parametric option pricing. *Finance and Stochastics*, 22(3):701–731, 2018.
16. P. Grohs and F. Voigtlaender. Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces. Technical report, 2021. ArXiv 2104.02746.
17. F. Henríquez and C. Schwab. Shape holomorphy of the Calderón projector for the Laplacian in $\mathbb{R}^2$. *Integral Equations Operator Theory*, 93(4):Paper No. 43, 40, 2021.
18. L. Herrmann, C. Schwab, and J. Zech. Deep neural network expression of posterior expectations in Bayesian PDE inversion. *Inverse Problems*, 36(12):125011, 2020.
19. L. Herrmann and Ch. Schwab. Multilevel quasi-Monte Carlo uncertainty quantification for advection-diffusion-reaction. In *Monte Carlo and quasi-Monte Carlo methods*, volume 324 of *Springer Proc. Math. Stat.*, pages 31–67. Springer, Cham, [2020] ©2020.
20. B. Hosseini and N. Nigam. Well-posed Bayesian inverse problems: priors with exponential tails. *SIAM/ASA J. Uncertain. Quantif.*, 5(1):436–465, 2017.
21. C. Jerez-Hanckes, C. Schwab, and J. Zech. Electromagnetic Wave Scattering by Random Surfaces: Shape Holomorphy. *Math. Mod. Meth. Appl. Sci.*, 27(12):2229–2259, 2017.
22. B. Li, S. Tang, and H. Yu. Better approximations of high dimensional smooth functions by deep neural networks with rectified power units. *Communications in Computational Physics*, 27(2):379–411, 2019.
23. S. Liang and R. Srikant. Why deep neural networks for function approximation? In *Proc. of ICLR 2017*, pages 1 – 17, 2017. ArXiv:1610.04161.
24. L. Lu, P. Jin, and G. E. Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, 2020. arXiv: 1910.03193.
25. K. O. Lye, S. Mishra, and D. Ray. Deep learning observables in computational fluid dynamics. *J. Comput. Phys.*, 410:109339, 26, 2020.
26. H. N. Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, 1(1):61–80, Feb 1993.
27. H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8(1):164–177, 1996.
28. J. A. A. Opschoor. *In preparation.* PhD thesis, Dissertation, ETH Zürich, 202x.
29. J. A. A. Opschoor, P. C. Petersen, and C. Schwab. Deep ReLU networks and high-order finite element methods. *Analysis and Applications*, 18(05):715–770, 2020.
30. J. A. A. Opschoor, C. Schwab, and J. Zech. Exponential ReLU DNN expression of holomorphic maps in high dimension. *Constructive Approximation.* published online April 23rd, 2021.
31. P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Netw.*, 108:296 – 330, 2018.
32. 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. *J. Comput. Phys.*, 378:686–707, 2019.
33. T. J. Rivlin. *The Chebyshev polynomials.* Wiley-Interscience [John Wiley & Sons], New York-London-Sydney, 1974. Pure and Applied Mathematics.
34. D. Rolnick and M. Tegmark. The power of deeper networks for expressing natural functions. In *International Conference on Learning Representations*, 2018.
35. C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Analysis and Applications, Singapore*, 17(1):19–55, 2019.

36. S. Tang, B. Li, and H. Yu. ChebNet: Efficient and stable constructions of deep neural networks with rectified power units using Chebyshev approximations. Technical report, 2019. ArXiv: 1911.05467.
37. L. N. Trefethen. *Approximation theory and approximation practice*. Society for Industrial and Applied Mathematics, Philadelphia, extended edition, 2019.
38. L. Yang, X. Meng, and G. E. Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425:109913, Jan 2021.
39. D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Netw.*, 94:103–114, 2017.
40. J. Zech and C. Schwab. Convergence rates of high dimensional Smolyak quadrature. *ESAIM Math. Model. Numer. Anal.*, 54(4):1259–1307, 2020.