Proceedings of Machine Learning Research vol 145:1–36, 2021 2nd Annual Conference on Mathematical and Scientific Machine Learning

Deep Neural Networks Are Effective At Learning High-Dimensional Hilbert-Valued Functions From Limited Data

Ben Adcock BEN_ADCOCK@SFU.CA
Department of Mathematics, Simon Fraser University, Canada

Simone Brugiapaglia SIMONE.BRUGIAPAGLIA@CONCORDIA.CA
Department of Mathematics and Statistics, Concordia University

Nick Dexter NICHOLAS_DEXTER@SFU.CA
Department of Mathematics, Simon Fraser University, Canada

Sebastian Moraga SMORAGAS@SFU.CA
Department of Mathematics, Simon Fraser University, Canada

Editors: Joan Bruna, Jan S Hesthaven, Lenka Zdeborova

Abstract

The accurate approximation of scalar-valued functions from sample points is a key task in mathematical modelling and computational science. Recently, machine learning techniques based on Deep Neural Networks (DNNs) have begun to emerge as promising tools for function approximation in scientific computing problems, with some impressive results achieved on problems where the dimension of the underlying data or problem domain is large. In this work, we broaden this perspective by focusing on the approximation of functions that are *Hilbert-valued*, i.e. they take values in a separable, but typically infinite-dimensional, Hilbert space. This problem arises in many science and engineering problems, in particular those involving the solution of parametric Partial Differential Equations (PDEs). Such problems are challenging for three reasons. First, pointwise samples are expensive to acquire. Second, the domain of the function is usually high dimensional, and third, the range lies in a Hilbert space. Our contributions are twofold. First, we present a novel result on DNN training for holomorphic functions with so-called *hidden anisotropy*. This result introduces a DNN training procedure and a full theoretical analysis with explicit guarantees on the error and sample complexity. This error bound is explicit in the three key errors occurred in the approximation procedure: the best approximation error, the measurement error and the physical discretization error. Our result shows that there exists a procedure (albeit a non-standard one) for learning Hilbert-valued functions via DNNs that performs as well as, but no better than current best-in-class schemes. It therefore gives a benchmark lower bound for how well methods DNN training can perform on such problems. Second, we examine whether better performance can be achieved in practice through different types of architectures and training. We provide preliminary numerical results illustrating the practical performance of DNNs on Hilbert-valued functions arising as solutions to parametric PDEs. We consider different parameters, modify the DNN architecture to achieve better and competitive results and compare these to current best-in-class schemes.

Keywords: deep neural networks, deep learning, high-dimensional approximation, parametric PDEs, Hilbert-valued functions, polynomial approximations, anisotropy

1. Introduction

Driven by their success in many historically-challenging machine learning problems, Deep Neural Networks (DNNs) and Deep Learning (DL) are beginning to be applied successfully to challenging tasks in computational science and engineering. Such tasks are often characterized by the high dimensionality of their data or problem. Examples include inverse problems in imaging Adcock and Hansen (2021); Ongie et al. (2020), molecular dynamics simulations Faber et al. (2017), protein structure prediction Jumper et al. (2020), discovery of unknown dynamical systems Lagergren et al. (2020), Partial Differential Equations (PDEs) Berg and Nyström (2018) and, as discussed below, parameterized PDEs for Uncertainty Quantification (UQ).

The application of DL to such problems is supported by a rapidly-growing theory on the approximation properties of DNNs (see, e.g., Yarotsky (2017); Bach (2017); Petersen and Voigtlaender (2018); Beck et al. (2019); Grohs et al. (2019)). This has become an extremely active area, with many new results produced within the last several years. Generalizing the classical universal approximation theorem Cybenko (1989); Hornik et al. (1989); Leshno et al. (1993), recent works have shown approximation results for DNNs in terms of their depth Liang and Srikant (2016); Lu et al. (2020); Yarotsky (2018), for functions in Sobolev spaces Gühring et al. (2020), Hölder spaces Shen et al. (2020) and Barron spaces E et al. (2019), for bandlimited functions Montanelli et al. (2019) and holomorphic functions E and Q (2018); Opschoor et al. (2019), as well as for tasks in scientific computing such as approximation of high-dimensional functions Schwab and Zech (2019); Li et al. (2019) and PDEs Grohs et al. (2018); Berner et al. (2020), dimensionality reduction Zhang et al. (2019), and methods for DEs Lu et al. (2017); E and Yu (2018). Other works have established connections between DNNs and classical methods of approximation such as polynomials Schwab and Zech (2019); Daws and Webster (2019b), splines Unser (2019), sparse grids Montanelli and Du (2019) and finite elements Opschoor et al. (2019).

The above list represents only a selection of the many recent results in this area. However, it is notable that these results generally fall into the category of *existence theory*: namely, they assert the existence of a DNN with desirable approximation properties, but not a constructive means to compute such a network (we note in passing several exceptions Dereventsov et al. (2019); Fokina and Oseledets (2019); Daws and Webster (2019a), although these generally lack theoretical guarantees on trainability). As discussed in Geist et al. (2020) and shown in Adcock and Dexter (2021), there can often be a substantial gap between theoretical existence results and practical performance of DNNs when trained using standard tools.

Motivated by such a performance gap, this paper considers the following three key issues:

(1) The vast majority of previous work considers only scalar-valued function approximation.

(2) Existence theory says little about whether such a DNN can be obtained by training and the number of samples of the function needed to do so.

(3) Many applications in computational science are relatively *data starved*. Hence it is critical to understand the *sample complexity* DNN approximation: namely, how much data is needed to train an accurate DNN for a given function.

Specifically, the focus of this paper is on learning *high-dimensional*, *Hilbert-valued* functions from *limited datasets* using DNNs. We next describe the motivations for considering this problem.

1.1. Motivations

An important task in UQ involves constructing a surrogate model of a physical system that depends on a set of parameters $\boldsymbol{y} \in \mathbb{R}^d$. The physical system is typically modelled via a PDE (or system of PDEs) in terms of the relevant spatial and temporal variables. In other words, its solution is a function $u = u(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} \in \mathbb{R}^n$, $n = 1, 2, 3, 4$, denotes the physical variables (space and time) and $\boldsymbol{y}$ denotes the parameters. The function u is the solution to a PDE system

$$\mathcal{D}_{\boldsymbol{x}}(u, \boldsymbol{y}) = 0, \tag{1}$$

where $\mathcal{D}_{\boldsymbol{x}}(\cdot, \boldsymbol{y})$ is a differential operator in the physical variable $\boldsymbol{x}$ that depends on the parametric variable $\boldsymbol{y}$. The objective in surrogate model construction is to understand the parametric dependence of the solution u. Since the PDE problem can often be posed (in weak form) in a separable Hilbert space $\mathcal{V}$, this is equivalent to approximating the *Hilbert-valued* function

$$\boldsymbol{y} \in \mathbb{R}^d \mapsto u(\boldsymbol{y}) \in \mathcal{V}, \tag{2}$$

(we suppress the $\boldsymbol{x}$ dependence for ease of notation). This problem have several key features that motivate this work (see, e.g. Cohen and DeVore (2015); Gunzburger et al. (2014) for more details):

(i) The input dimension d is typically high. Indeed, the more parameters, the better the model for the physical system. Typically, d may range from ten to over a hundred. Moreover, in some situations, one may also consider functions with infinitely-many parameters, i.e. $d = \infty$.

(ii) The output $u(\boldsymbol{y})$ takes values in a Hilbert space $\mathcal{V}$. In some applications one may only wish to approximate some scalar *Quantity-of-Interest (QoI)* of u (e.g. the spatial mean $f(\boldsymbol{y}) = \int_\Omega u(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x}$ over the physical domain $\Omega \subseteq \mathbb{R}^n$). However, other applications call for approximating the whole solution $u(\boldsymbol{y})$ (from which one can obviously approximate any number of QoI's).

(iii) Computing samples is expensive. For each value of $\boldsymbol{y}$, evaluating $u(\boldsymbol{y})$ requires solving the PDE (1) via a computationally-intensive numerical simulation. In practice, generating the samples may take a time ranging from minutes to days or even weeks. Furthermore, since the PDE is never solved exactly, this process always commits an error – the *measurement error* as we term it.

To summarize, surrogate model construction involves approximating a high-dimensional, Hilbert-valued function from limited data. Such a task is clearly impossible without further assumptions. Fortunately, u is often *smooth*. To be precise, under certain conditions on the problem (1) one can show that u is a *holomorphic* function of the parameters $\boldsymbol{y}$ Cohen et al. (2010, 2011); Chkifa et al. (2015) (see also Cohen and DeVore (2015) for an in-depth review). This opens the door for approximating u efficiently from limited samples, even when d is large, and thereby lessening the *curse of dimensionality*. We shall exploit this assumption throughout.

Even with this assumption, though, there is another hurdle to overcome; namely the infinite dimensionality of $\mathcal{V}$. The usual way to address this is to introduce a finite-dimensional *discretization* $\mathcal{V}_h$, where h is a discretization parameter, and compute an approximation taking values in $\mathcal{V}_h$. Typically, since u is the solution of a PDE, one takes $\mathcal{V}_h$ to be a finite element discretization. However, regardless of how $\mathcal{V}_h$ is chosen, it is important that the *discretization error* (the effect of replacing $\mathcal{V}$ by $\mathcal{V}_h$) be quantified explicitly in the overall error bound. We also address this matter.

1.2. Contributions

In this work we study the approximation of a holomorphic, Hilbert-valued function $f : \mathcal{U} \to \mathcal{V}$ (we now switch notation from u to f since the problem we consider does not necessarily need to arise

as the solution of a parametric PDE) from m noisy sample values

$$d_i = f(\boldsymbol{y}_i) + n_i, \quad \forall i = 1, \dots, m.$$

We assume throughout that $\mathcal{U} = [-1, 1]^d$ is the unit hypercube and the $\boldsymbol{y}_i$ are drawn identically and independently from the uniform measure on $\mathcal{U}$. Note that this choice of sampling is not only typical in practice, it is critical in allowing for theoretical sample complexity estimates that scale efficiently with the dimension d. The choice of the hypercube implies that the parameters $y_1, \dots, y_d$ (where $\boldsymbol{y} = (y_i)_{i=1}^d$) are independent and vary in finite intervals (which, up to rescaling, can be taken to be equal to $[-1, 1]$). Both assumptions are also standard in practice.

The values n_i are the *measurement errors*. They constitute the errors involved in computing f, whether it be via a numerical PDE solve or some other unspecified process. Throughout, we assume that $\mathcal{V}$ is a separable Hilbert space and that it is discretized via a finite-dimensional subspace $\mathcal{V}_h \subseteq \mathcal{V}$. We make the additional assumption that the samples d_i are elements of $\mathcal{V}_h$, i.e.

$$d_i \in \mathcal{V}_h, \quad \forall i = 1, \dots, m,$$

and we seek to compute an approximation $\tilde{f} : \mathcal{U} \to \mathcal{V}_h$ to f taking values in $\mathcal{V}_h$.

Our first main contribution is a novel theoretical result, Theorem 5. It shows that there exists a DNN architecture (of a given size and depth depending on m and d) and a training procedure (i.e. a loss function) such that any minimizer of the corresponding loss function minimization problem approximates f up to an explicit error bound. This error bound splits into three key terms that fully describe the effects of all main errors involved in the approximation process:

(a) an *approximation error* that is exponentially-small in $m^{1/(2d)}$, up to log factors;

(b) a *measurement error* that is proportional to the ℓ^2-norm of the measurement noise $(n_i)_{i=1}^m$;

(c) a *physical discretization error* that is proportional to the best approximation error of f in $\mathcal{V}_h$.

We term this result a *practical existence theorem* for Hilbert-valued function approximation. Going beyond standard existence theory, as discussed above, it shows not only the existence of a DNN with desirable approximation properties, but both a means of obtaining it via training and an estimate on the sample complexity via the exponentially-decaying approximation error (a).

A key facet of this work is the assumption on f. While we assume f is holomorphic, we do not assume any further information on it. In particular, f may have *anisotropic* dependence on the parameters $y_1, \dots, y_d$ – i.e. it may vary more rapidly in some directions than in others – and such anisotropy may be *hidden* – i.e. it is not used to construct the approximation. This *hidden anisotropy* assumption is highly relevant in practice, yet substantially harder to tackle than scenarios where such behaviour is known *a priori*. We formalize exactly what we mean by hidden anisotropy in §3.

As we note below, current best-in-class methods for holomorphic, Hilbert-valued function approximation are based on multivariate orthogonal polynomials and also achieve exponential rates of convergence in $m^{1/(2d)}$. Through (a) we show that the obtained DNN achieves the same rate of convergence, up to a constant. Hence this work shows the existence of a DNN training procedure that can perform as well as current best-in-class methods.

Further, we also categorize all the errors in the DNN training procedure, including the measurement error (b) and physical discretization error (c). The latter is ubiquitous in simulations (due to the need to work with $\mathcal{V}_h$ instead of $\mathcal{V}$), but often not included in theoretical analyses. Our work thus sheds light on understanding how to tune the method parameters (m, h, and so forth) to balance the various errors, thus leading to optimal practical performance. It also raises the potential for the use of multilevel or multifidelity schemes, where m and h are simultaneously refined.

Note that we establish Theorem 5 by carefully re-interpreting a polynomial-based approximation based on compressed sensing as a DNN training procedure in which all the weights and biases are fixed, except those in the final layer. We do not claim that this training procedure is practically advantageous to use – indeed, since it mimics a polynomial approximation by construction, it is unlikely to offer better performance than the latter. The purpose of Theorem 5 is to show that there are provably good ways to set up and train DNNs for holomorphic function approximation, even if these are not standard procedures. It highlights the potential to achieve better performance in practice with trained DNNs, by modifying the training setup and DNN architectures suitably. Further, since polynomial-based methods are strongly tied to the underlying smoothness of the functions being approximated, while DNNs are not, it suggests DNNs can be useful flexible tools, with the potential of achieving good performance across a range of different function classes.

Nonetheless, with this gap between theory and practice in mind, we end this paper by presenting initial numerical experiments showing the effectiveness of trained DNNs for parametric PDE problems. We also compare with best-in-class compressed sensing-based polynomial approximation schemes, with a focus on the sample complexity of both approaches. Theorem 5 shows that there exists a DNN architecture and training procedure that performs at least as well as such schemes in this regard, but, as noted, this setup is neither standard, nor expected to yield any better performance in practice. Following Adcock and Dexter (2021), in our experiments we use standard DNN architectures coupled with standard loss functions and training algorithms. We show that with proper architecture and hyperparameter selection we are able to achieve competitive results. In particular, we can use smaller architectures than those suggested by Theorem 5 and the standard ℓ^2-loss functions, provided we train all the weights and biases of the DNN. Further, we show that we can actually outperform state-of-the-art polynomial methods for the problem considered by switching from the ReLU (as used in Theorem 5) to smoother activations functions. All in all, these preliminary results demonstrate the promise of the DNN approach for parametric PDEs in terms of sample complexity. This complements recent results shown in Geist et al. (2020), which showed favourable performance of DNNs with respect to the dimension d.

1.3. Related work

There are various ways to approximate the solution map of a parametric PDE, including *reduced basis methods* Hesthaven et al. (2015) and *polynomial chaos expansions* Xiu and Karniadakis (2002). In this paper, we focus on comparing DNN performance against the latter. Polynomial expansions are well-suited to smooth function approximation, with the so-called best s-term polynomial approximation offering exponential rates of convergence in $s^{1/d}$ for holomorphic functions (see §3). For a function with isotropic or known anisotropic behaviour in its variables, such an approximation can be computed in a number of ways, including interpolation or least squares Cohen and Migliorati (2018). The situation becomes more challenging when the anisotropy is hidden. Adaptive interpolation or least-squares schemes Chkifa et al. (2013, 2014); Cohen and Migliorati (2018); Gittelson (2013); Migliorati (2015) may work in practice, though they generally lack theoretical guarantees.

In the last five years, techniques based on compressed sensing have emerged as viable tools for this problem (see, e.g., Adcock et al. (2017, 2019); Chkifa et al. (2018); Doostan and Owhadi (2011); Hampton and Doostan (2015) and references therein). Theoretical guarantees show that such techniques provide quasi-best s-term polynomial approximations, with favourable sample complexities Adcock (2018); Adcock et al. (2017); Chkifa et al. (2018). However, standard com-

pressed sensing only allows for recovery of real or complex sparse vectors, e.g., scalar QoI's of the solution map (2), and therefore does not allow for recovery guarantees for Hilbert-valued functions. The *Simultaneous Compressed Sensing* (SCS) method Dexter et al. (2019) enables fully discrete approximation of the solution map by combining spatial discretization, e.g., finite elements, with joint-sparse vector recovery techniques modified for recovery in $\mathcal{V}_h$. The SCS framework also extends theoretical recovery guarantees from compressed sensing to the Hilbert-valued setting, thereby inheriting the same quasi-best s-term approximation rates and sample complexity estimates. Because of its favourable behaviour and theoretical guarantees, we refer to SCS as the current best-in-class method, and seek to match (or beat) this performance with a DNN procedure.

Recently, a number of works have applied DNNs to parametric PDEs. See Cyr et al. (2020); Dal Santo et al. (2020); Geist et al. (2020); Khoo et al. (2020); Laakmann and Petersen (2020) and references therein. These works generally lack theoretical analysis. On the theoretical side, Opschoor et al. (2019) provides an existence theorem for parametric PDEs with holomorphic solution maps. The work Kutyniok et al. (2020) considers the reduced basis approach to parametric PDEs, showing the existence of a DNN whose size depends on the intrinsic low-dimensionality of the solution manifold. Neither result addresses whether such a DNN can be trained, nor the sample complexity in doing so. Our work in particular complements Opschoor et al. (2019) by showing that DNNs admitting the same approximation error can be obtained as solutions of certain training problems. Furthermore, our result also makes all errors committed by the process explicit, including the aforementioned measurement and physical discretization errors, which have typically not been addressed in previous works.

2. Learning Hilbert-valued functions via DNNs

We first require some notation. Throughout $d \in \mathbb{N}$ denotes the dimension of the input space. We write $\mathbb{N}_0^d := \{\boldsymbol{\nu} = (\nu_k)_{k=1}^d : \nu_k \in \mathbb{N}_0\}$ for the set of nonnegative integer multi-indices and $\mathbf{0}$ and $\mathbf{1}$ for the multi-indices consisting of all zeros and all ones respectively. The inequality $\boldsymbol{\mu} \leq \boldsymbol{\nu}$ between two multi-indices is understood componentwise.

For $1 \leq p \leq \infty$, we write $\|\cdot\|_p$ for the usual vector ℓ^p-norm and for the induced matrix ℓ^p-norm. Moreover, for $1 \leq p, q < \infty$ we define the matrix $\ell^{p,q}$-norm as $\|\boldsymbol{G}\|_{p,q}^q := \sum_{j=1}^n (\sum_{i=1}^m |G_{ij}|^p)^{q/p}$. We also use the notation $A \lesssim B$ to mean that there exists a numerical constant $c > 0$ independent of A and B such that $A \leq cB$, and likewise for $A \gtrsim B$.

2.1. Setup

We now describe the setup in detail. We write $\boldsymbol{y} = (y_1, \dots, y_d)$ for the variables and let $\mathcal{U} = [-1, 1]^d$. We consider the uniform probability measure on $\mathcal{U}$, i.e.

$$\mathrm{d}\varrho(\boldsymbol{y}) = 2^{-d}\,\mathrm{d}\boldsymbol{y}, \quad \forall \boldsymbol{y} \in \mathcal{U}, \tag{3}$$

and write $L_\varrho^p(\mathcal{U})$ for the corresponding weighted Lebesgue spaces of scalar-valued functions over $\mathcal{U}$ and $\|\cdot\|_{L_\varrho^p(\mathcal{U})}$ for their norms.

Throughout, we let $\mathcal{V}$ be a separable Hilbert space over the field $\mathbb{R}$ (we could also consider the field $\mathbb{C}$ with few additional difficulties), with inner product $\langle \cdot, \cdot \rangle_\mathcal{V}$ and corresponding norm

$$\|v\|_\mathcal{V} := \sqrt{\langle v, v \rangle_\mathcal{V}}.$$

We let $\mathcal{V}^N$ be the vector space of Hilbert-valued vectors of length N, i.e. $\boldsymbol{\nu} = (\nu_i)_{i=1}^N$ where $\nu_i \in \mathcal{V}$, $i = 1, \dots, N$. More generally, let $\Lambda \subseteq \mathbb{N}_0^d$ denote a (possibly infinite) multi-index set. We write $\boldsymbol{v} = (v_{\boldsymbol{\nu}})_{\boldsymbol{\nu}\in\Lambda}$ for a sequence with $\mathcal{V}$-valued entries, $v_{\boldsymbol{\nu}} \in \mathcal{V}$. For $1 \leq p \leq \infty$, we define the space $\ell^p(\Lambda; \mathcal{V})$ as the set of those sequences $\boldsymbol{v} = (v_{\boldsymbol{\nu}})_{\boldsymbol{\nu}\in\Lambda}$ for which $\|\boldsymbol{v}\|_{\mathcal{V},p} < \infty$, where

$$\|\boldsymbol{v}\|_{\mathcal{V},p} := \begin{cases} \left(\sum_{\boldsymbol{\nu}\in\Lambda} \|v_{\boldsymbol{\nu}}\|_{\mathcal{V}}^p\right)^{1/p} & 1 \leq p < \infty \\ \sup_{\boldsymbol{\nu}\in\Lambda} \|v_{\boldsymbol{\nu}}\|_{\mathcal{V}} & p = \infty \end{cases}.$$

When $p = 2$ this is a Hilbert space with inner product

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle_{\mathcal{V},2} = \sum_{\boldsymbol{\nu}\in\Lambda} \langle u_{\boldsymbol{\nu}}, v_{\boldsymbol{\nu}}\rangle_{\mathcal{V}}.$$

Next, we define the weighted (Lebesgue-)Bochner space $L^p_\varrho(\mathcal{U}; \mathcal{V})$ as the space consisting of (equivalence classes of) strongly ϱ-measurable functions $f : \mathcal{U} \to \mathcal{V}$ for which $\|f\|_{L^p_\varrho(\mathcal{U};\mathcal{V})} < \infty$, where

$$\|f\|_{L^p_\varrho(\mathcal{U};\mathcal{V})} := \begin{cases} \left(\int_{\mathcal{U}} \|f(\boldsymbol{y})\|_{\mathcal{V}}^p \,\mathrm{d}\varrho(\boldsymbol{y})\right)^{1/p} & 1 \leq p < \infty \\ \operatorname{ess\,sup}_{\boldsymbol{y}\in\mathcal{U}} \|f(\boldsymbol{y})\|_{\mathcal{V}} & p = \infty \end{cases}. \tag{4}$$

In general, we cannot work directly in the space $\mathcal{V}$, since it is usually infinite dimensional. Hence, we consider a finite-dimensional discretization

$$\mathcal{V}_h \subseteq \mathcal{V}. \tag{5}$$

Here $h > 0$ denotes a discretization parameter, e.g. the mesh size in the case of a finite element discretization. In the context of finite elements, assuming (5) corresponds to considering so-called conforming discretizations. We let $\{\varphi_k\}_{k=1}^K$ be a (not necessarily orthonormal) basis of $\mathcal{V}_h$, where $K = K(h) = \dim(\mathcal{V}_h)$. We write $\mathcal{P}_h : \mathcal{V} \to \mathcal{V}_h$ for the orthogonal projection onto $\mathcal{V}_h$ and for $f \in L^2_\varrho(\mathcal{U}; \mathcal{V})$ we let $\mathcal{P}_h f \in L^2_\varrho(\mathcal{U}; \mathcal{V}_h)$ be the function defined almost everywhere as

$$(\mathcal{P}_h f)(\boldsymbol{y}) = \mathcal{P}_h(f(\boldsymbol{y})), \quad \forall \boldsymbol{y} \in \mathcal{U}.$$

2.2. DNNs

Let $f : \mathcal{U} \to \mathcal{V}$ and write its projection in terms of the basis $\{\varphi_k\}_{k=1}^K$ for $\mathcal{V}_h$ as

$$f \approx (\mathcal{P}_h f)(\boldsymbol{y}) = \sum_{k=1}^K c_k(\boldsymbol{y})\varphi_k.$$

Notice that the *coefficients* c_k are scalar-valued functions of $\boldsymbol{y}$, i.e. $c_k : \mathcal{U} \to \mathbb{R}$. Our objective is to approximate the coefficients with a DNN. We consider standard feedforward DNN architectures of the form $\Phi : \mathbb{R}^d \to \mathbb{R}^K$, where

$$\Phi(\boldsymbol{y}) = \mathcal{A}_{L+1}(\sigma(\mathcal{A}_L(\sigma(\cdots\sigma(\mathcal{A}_0(\boldsymbol{x}))\cdots)))). \tag{6}$$

Here $\mathcal{A}_l : \mathbb{R}^{N_l} \to \mathbb{R}^{N_{l+1}}$, $l = 0, \dots, L+1$ are affine maps and σ is the activation function, which we assume acts componentwise, i.e. $\sigma(\boldsymbol{y}) := (\sigma(y_i))_{i=1}^d$ for $\boldsymbol{y} = (y_i)_{i=1}^d$. The values $\{N_l\}_{l=1}^{L+1}$ are the widths of the hidden layers. By definition $N_0 = d$ and $N_{L+2} = K$. Given (6), we write

$$f_\Phi(\boldsymbol{y}) = \sum_{k=1}^K (\Phi(\boldsymbol{y}))_k \varphi_k, \tag{7}$$

for the resulting approximation to f. We also write $\mathcal{N}$ for a class of DNNs of the form (6). We term L the *depth*, and denote this as $\mathrm{depth}(\mathcal{N})$. We write $\mathrm{param}(\mathcal{N})$ for the number of trainable parameters in $\mathcal{N}$ (i.e. the number of weights and biases that parameterize $\mathcal{N}$). We also write $\mathrm{size}(\mathcal{N})$ for the size of the DNNs in $\mathcal{N}$. This is equal to the total number of nonzero weights and biases.

2.3. Problem statement

Let $f : \mathcal{U} \to \mathcal{V}$ be a Hilbert-valued function and consider m sample points $\boldsymbol{y}_1, \dots, \boldsymbol{y}_m$ drawn independently from the uniform measure (3). We assume noisy evaluations of f of the form

$$d_i = f(\boldsymbol{y}_i) + n_i \in \mathcal{V}_h, \quad \forall i = 1, \dots, m, \tag{8}$$

where $n_i \in \mathcal{V}$ is the ith noise term. It is important to note that the samples d_i are elements of the finite-dimensional space $\mathcal{V}_h$. Hence, the term n_i encompasses the error involved in approximating $f(\boldsymbol{y}_i) \in \mathcal{V}$ by an element of $\mathcal{V}_h$. This takes into account the fact that these measurements are computed by some unspecified routine (e.g. a PDE solver in the case of parametric PDEs) which returns a value in the finite-dimensional Hilbert space $\mathcal{V}_h$ (e.g. a finite element space). We do not specify precisely how this approximation is performed, nor how large this error is, but merely aim to show an error bound that scales linearly with respect to the n_i. A particular case is when $d_i = \mathcal{P}_h(f(\boldsymbol{y}_i))$, but in what follows it is not necessary to assume this, as the numerical computation that leads to d_i may not involve computing the projection $\mathcal{P}_h$. We remark in passing that one can easily extend this analysis to the case where the space $\mathcal{V}_h$ used for constructing the approximation f_Φ to f differs from that used to compute the evaluation of f.

Since any algorithm for learning a DNN needs to take finite inputs, we assume the evaluations (8) are provided to us via the basis $\{\varphi_k\}$. To be precise, we assume we have access to the data

$$\{d_{ik}\}_{i,k=1}^{m,k}, \quad \text{where } f(\boldsymbol{y}_i) + n_i = \sum_{i=1}^{K} d_{ik}\varphi_k. \tag{9}$$

With this in hand, the problem is as follows:

Problem 1 *Use the data (9) to learn a DNN $\hat{\Phi} : \mathbb{R}^d \to \mathbb{R}^K$ from the class $\mathcal{N}$, and therefore an approximation $f_{\hat{\Phi}}$ to f of the form (7).*

3. Holomorphy, best s-term polynomial approximation and hidden anisotropy

3.1. Holomorphy

We start by recalling the definition of holomorphy and of holomorphic extension for Hilbert-valued functions. The definition of holomorphy employed here is based on the notion of Gateaux partial derivative, although other equivalent definitions are possible (see (Hervé, 2011, Chapter 2)). The holomorphic extension assumption has played a crucial role in the context of parametric PDEs since the seminal work Cohen et al. (2011); see also Cohen and DeVore (2015) and references therein.

Definition 1 (Holomorphy) *Let $\mathcal{O} \subseteq \mathbb{C}^d$ be an open set and $\mathcal{V}$ be a separable Hilbert space. A function $f : \mathcal{O} \to \mathcal{V}$ is holomorphic in $\mathcal{O}$ if and only if it is holomorphic with respect to each variable in $\mathcal{O}$. Equivalently, if and only if the following limit exists for any $z \in \mathcal{O}$ and any $j \in [d]$:*

$$\lim_{\substack{h \in \mathbb{C} \\ h \to 0}} \frac{f(z + h\boldsymbol{e}_j) - f(z)}{h} \in \mathcal{V}, \quad \textit{where } \boldsymbol{e}_j = (\delta_{ij})_{i=1}^d.$$

Definition 2 (Holomorphic extension) *Let $\mathcal{V}$ be a Hilbert space. The function $f : \mathcal{U} \to \mathcal{V}$ is holomorphic in $\mathcal{U} \subseteq \mathcal{O} \subseteq \mathbb{C}^d$ if it has a holomorphic extension to $\mathcal{O}$, i.e. there is a $\widetilde{f} : \mathcal{O} \to \mathcal{V}$ that is holomorphic in $\mathcal{O}$ with $\widetilde{f}|_{\mathcal{U}} = f$. In this case, we also define $\|f\|_{L^\infty(\mathcal{O};\mathcal{V})} := \|\widetilde{f}\|_{L^\infty(\mathcal{O};\mathcal{V})}$.*

We are interested in approximating Hilbert-valued functions $f : \mathcal{U} \to \mathcal{V}$ that admit a holomorphic extension to suitable open neighborhoods of $\mathcal{U}$. Specifically, regions defined by *Bernstein (poly)ellipses*. When $d = 1$ the Bernstein ellipse of parameter $\rho > 1$ is defined by $\mathcal{E}_\rho = \left\{\frac{1}{2}(z + z^{-1}) : z \in \mathbb{C}, 1 \leq |z| \leq \rho\right\} \subset \mathbb{C}$. This is an ellipse with ± 1 as its foci and major and minor semi-axis lengths given by $\frac{1}{2}(\rho \pm \rho^{-1})$. For $d \geq 1$, given $\boldsymbol{\rho} = (\rho_j)_{j=1}^d \in \mathbb{R}^d$ with $\boldsymbol{\rho} > \mathbf{1}$, we define the Bernstein polyellipse as

$$\mathcal{E}_{\boldsymbol{\rho}} = \mathcal{E}_{\rho_1} \times \cdots \times \mathcal{E}_{\rho_d} \subset \mathbb{C}^d.$$

Further, we denote the class of unit-norm, Hilbert-valued functions that are holomorphic in $\mathcal{E}_{\boldsymbol{\rho}}$ as

$$\mathcal{B}(\boldsymbol{\rho}) = \left\{f : \mathcal{U} \to \mathcal{V}, f \text{ holomorphic in } \mathcal{E}_{\boldsymbol{\rho}},\ \|f\|_{L^\infty(\mathcal{E}_{\boldsymbol{\rho}};\mathcal{V})} \leq 1\right\}.$$

Note that the parameter $\boldsymbol{\rho}$ dictates the smoothness of f in the different coordinate directions. The larger ρ_j, the smoother f is in the jth variable y_j. We say that f has *anisotropic* dependence on the variables $\boldsymbol{y}$ if the ρ_j are potentially nonequal and that the anisotropy is *hidden* if the ρ_j's are unknown.

3.2. Polynomial approximation of holomorphic functions

The approximation theory of functions that are holomorphic in Bernstein ellipses is a classical topic, especially in $d = 1$ dimensions. However, polynomial approximations have also proved to be extremely effective tools in $d \gg 1$ dimensions as well.

Let $\{\Psi_\nu\}_{\nu \in \mathbb{N}_0}$ be the univariate orthonormal Legendre polynomial basis of $L^2_\varrho([-1, 1])$ (see, for example, Szegö (1975)). Let $f \in \mathcal{B}(\rho)$ for some $\rho > 1$ and consider its expansion

$$f = \sum_{\nu \in \mathbb{N}_0} c_\nu \Psi_\nu, \qquad c_\nu := \int_{-1}^{1} f(y)\Psi_\nu(y)2^{-1}\,\mathrm{d}y \in \mathcal{V},$$

where the coefficients $\{c_\nu\}_{\nu \in \mathbb{N}_0}$ are elements of $\mathcal{V}$. It is well-known that the truncated expansion $f_s = \sum_{\nu=0}^{s-1} c_\nu \Psi_\nu$ converges to f exponentially-fast with error $\mathcal{O}(\sqrt{s}\rho^{-s})$ in $L^2_\varrho([-1, 1]; \mathcal{V})$. [1]

The situation becomes more complicated in $d \geq 2$ dimensions. Let $\{\Psi_{\boldsymbol{\nu}}\}_{\boldsymbol{\nu} \in \mathbb{N}_0^d}$ be the tensor Legendre polynomial basis, defined by $\Psi_{\boldsymbol{\nu}} = \Psi_{\nu_1} \otimes \cdots \otimes \Psi_{\nu_d}$ for $\boldsymbol{\nu} = (\nu_k)_{k=1}^d \in \mathbb{N}_0^d$. Then, once more, we can then write $f \in \mathcal{B}(\boldsymbol{\rho})$ as

$$f = \sum_{\boldsymbol{\nu} \in \mathbb{N}_0^d} c_{\boldsymbol{\nu}} \Psi_{\boldsymbol{\nu}}, \qquad c_{\boldsymbol{\nu}} := \int_{\mathcal{U}} f(\boldsymbol{y})\Psi_{\boldsymbol{\nu}}(\boldsymbol{y})2^{-d}\,\mathrm{d}\boldsymbol{y} \in \mathcal{V}. \tag{10}$$

One aims to construct a polynomial approximation by selecting s terms from this expansion, i.e.

$$f \approx f_S = \sum_{\boldsymbol{\nu} \in S} c_{\boldsymbol{\nu}} \Psi_{\boldsymbol{\nu}},$$

1. In the scalar-valued case, i.e. $\mathcal{V} = \mathbb{R}$, this is due to classical estimates on the Legendre coefficients (see, e.g., (Davis, 1975, Theorem 12.4.7)). A simple adaptation of this argument leads to the same result in the Hilbert-valued case.

where $S \subset \mathbb{N}_0^d$ is a multi-index set with $|S| = s$. Unfortunately, standard, isotropic choices for S such as the *(isotropic) total degree* set $S = \{\boldsymbol{\nu} = (\nu_k)_{k=1}^d : \nu_1 + \cdots + \nu_d \leq n\}$ generally lead to less favourable convergence rates in terms of s. In particular, when the dependence of f on its variables $\boldsymbol{y}$ is anisotropic, the index set S may include many indices that correspond to small-norm coefficients $c_{\boldsymbol{\nu}}$. This motivates an alternative approach based on *best s-term approximation*, in which a polynomial approximation is formed by selecting the indices in (10) corresponding to the largest s of the coefficient norms $\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}}$; see DeVore (1998); Cohen et al. (2010).

Best s-term approximation is particularly well suited to approximating holomorphic functions with anisotropic dependence. If $f \in \mathcal{B}(\boldsymbol{\rho})$ then it is known that, for any $\epsilon > 0$,

$$\|f - f_s\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq \exp\left(-\frac{1}{d+1}\left(\frac{sd!\prod_{j=1}^d \log(\rho_j)}{1+\epsilon}\right)^{1/d}\right), \qquad \forall s \geq \bar{s} = \bar{s}(d,\epsilon,\boldsymbol{\rho}), \tag{11}$$

where f_s is the best s-term polynomial approximation to f (see Theorem 10). This shows that the best s-term approximation error converges exponentially-fast in $s^{1/d}$ without any prior knowledge of the ρ_j's. In particular, as the right-hand side of (11) depends on the product of the logarithms of the ρ_j's, it is completely independent of their ordering. We note that, although a closed formula for $\bar{s}(d,\epsilon,\boldsymbol{\rho})$ in (11) is not available, an inspection of the proof of Theorem 10 (see Appendix B.3) and of (Opschoor et al., 2019, Theorem 3.5) reveals a constructive characterization of this quantity.

It is notable that estimates such as (11) say nothing about how to actually construct the approximation f_s from sample values. Nevertheless, we consider the rate (11) as the benchmark against which we compare the developed DNN procedure.

Remark 3 *There are various results on the convergence rate of best s-term polynomial approximation of a holomorphic function. Algebraic rates of convergence can be found in, for instance, Cohen et al. (2011); Cohen and DeVore (2015). These are attractive since they also hold when $d = \infty$, thus theoretically permitting the approximation of functions of infinitely-many variables. However, in finite dimensions the constants in these error bounds may be large Tran et al. (2017). The bound (11) is based on (Cohen and DeVore, 2015, Sec. 3.9) and Opschoor et al. (2019). However, the scaling $1/(d+1)$ is not sharp. For quasi-optimal error bounds, see Beck et al. (2014, 2012); Tran et al. (2017). The results of Tran et al. (2017) are asymptotically sharp as $s \to \infty$. The challenge, however, for our purposes is that they do not generally lead to an approximation in a so-called lower set, unlike (11). This is an important component of our subsequent analysis (See §B.3).*

3.3. Hidden anisotropy

Motivated by this discussion, we now define the following class of functions:

Definition 4 *Let $d \geq 1$, $\gamma > 0$ and $\epsilon > 0$. Then $\mathcal{HA} = \mathcal{HA}(\gamma,\epsilon,d)$ is the set of Hilbert-valued functions $f : \mathcal{U} \to \mathcal{V}$ that have a holomorphic extension to a Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}}$ and satisfy $\|f\|_{L^\infty(\mathcal{E}_{\boldsymbol{\rho}};\mathcal{V})} \leq 1$, and where the parameters $\boldsymbol{\rho} = (\rho_j)_{j=1}^d$ satisfy*

$$\frac{1}{d+1}\left(\frac{d!\prod_{j=1}^d \log(\rho_j)}{1+\epsilon}\right)^{1/d} \geq \gamma. \tag{12}$$

Observe that, thanks to (11), for this class of functions we have

$$\|f - f_s\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq \exp(-\gamma s^{1/d}), \quad \forall f \in \mathcal{HA}(\gamma, \epsilon, d),\ \forall s \geq \bar{s}(d, \epsilon, \boldsymbol{\rho}). \tag{13}$$

It is this rate we seek to obtain with the DNN approximation, with some polynomial scaling between m and s. Specifically, we consider the following problem:

Problem 2 *Devise a DNN architecture and training procedure that solves Problem 1 and for which the error decays exponentially fast for all $f \in \mathcal{HA}(\gamma, \epsilon, d)$.*

4. Main result

In this section we present our main result that resolves Problem 2. Its proof can be found in Appendix B.4. Given $m \geq 1$, $0 < \varepsilon < 1$ and $d \geq 1$, we first define $\mathcal{L} = \mathcal{L}(m, d, \varepsilon)$ as

$$\mathcal{L}(m, d, \varepsilon) := c_0 \cdot \log(2m) \cdot \left(\log(2m) \cdot \min\{\log(2m) + d, \log(2m) \cdot \log(2d)\} + \log(\varepsilon^{-1})\right), \tag{14}$$

where $c_0 > 0$ is a universal constant, and

$$\widetilde{m} = \widetilde{m}(m, d, \varepsilon) := m/\mathcal{L}. \tag{15}$$

Theorem 5 *There are universal constants $c_0, c_1, c_2, c_3 > 0$ such that the following holds. Let $d \geq 1$, $0 < \epsilon, \varepsilon < 1$, $\gamma > 0$, $m \geq 1$ and $\widetilde{m}$ be as in (15). Let $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ be drawn independently from the uniform measure (3) on $\mathcal{U}$. Then there is (a) a class of neural networks $\mathcal{N}$ with ReLU activation functions, $N_0 = d$, $N_{L+2} = K$,*

$$\operatorname{depth}(\mathcal{N}) \leq c_1 \cdot (1 + d\log(d)) \cdot (1 + \log(\widetilde{m})) \cdot \left((\widetilde{m}/2^d)^{1/2} + \log(\Delta) + \gamma \widetilde{m}^{1/(2d)}\right),$$

$$\operatorname{size}(\mathcal{N}) \leq c_2 \cdot d \left(d(\widetilde{m}/2^d) + \left((\widetilde{m}/2^d)^{1/2} + d \cdot \Delta\right) \cdot \left(\log(\widetilde{m}) + \log(\Delta) + \gamma \widetilde{m}^{1/(2d)}\right)\right) + K \cdot \Delta,$$

and $\operatorname{param}(\mathcal{N}) \leq K \cdot \Delta$, where

$$\Delta := \min\left\{2\widetilde{m}^{3/2} 2^{d/2}, \mathrm{e}^2 (\widetilde{m}/2^d)^{1+\log(d)/(2\log(2))}, \frac{\widetilde{m}^{1/2} \left(\log(\widetilde{m}) + (d+1)\log(2)\right)^{d-1}}{2^{d/2-1}(d-1)!}\right\},$$

(with the convention that $0! = 1$); (b) a regularization function $\mathcal{J} : \mathcal{N} \to [0, \infty)$ equivalent to a certain norm of the trainable parameters; and (c) a choice of regularization parameter λ involving only $\widetilde{m}$ and d; such that the following holds with probability at least $1 - \varepsilon$. For all $f \in \mathcal{HA}(\gamma, \epsilon, d)$ with noisy evaluations $d_i = f(\boldsymbol{y}_i) + n_i \in \mathcal{V}_h$ as in (8), every minimizer $\hat{\Phi}$ of the training problem

$$\min_{\Phi \in \mathcal{N}} \sqrt{\frac{1}{m} \sum_{i=1}^{m} \|f_\Phi(\boldsymbol{y}_i) - d_i\|_{\mathcal{V}}^2} + \lambda \mathcal{J}(\Phi), \tag{16}$$

satisfies

$$\|f - f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq c_3 \left(E_1 + E_2 + E_3\right), \tag{17}$$

for $\widetilde{m} \geq 2^d \bar{s}^2$, where $\bar{s} = \bar{s}(d, \epsilon, \boldsymbol{\rho})$ is as in (13), f_Φ is as in (7) and, for $\boldsymbol{e} = \frac{1}{\sqrt{m}}(n_i)_{i=1}^m$,

$$E_1 = \exp(-\gamma \widetilde{m}^{1/(2d)}/\sqrt{2}), \quad E_2 = \|\boldsymbol{e}\|_{\mathcal{V},2}, \quad E_3 = \|f - \mathcal{P}_h(f)\|_{L^\infty(\mathcal{U};\mathcal{V})}. \tag{18}$$

This result asserts that there is a DNN architecture and training procedure with an explicit error bound comprising three terms. First, the *approximation error* E_1. This quantifies how well f is approximated by a DNN in terms of the number of samples m. It is exponentially small in $m^{1/(2d)}$, up to log terms. Furthermore, the log term $\mathcal{L} = \mathcal{L}(m, d, \varepsilon)$ effectively behaves roughly like $\log^3(m)\log(d)$, i.e. polylogarithmic in m but only logarithmic in d. The second term is the *measurement error* $E_2 = \|\boldsymbol{e}\|_{\mathcal{V},2}$. This is, as discussed, the error in the pointwise evaluations of f at the points $\boldsymbol{y}_i$. Note that we do not assume $\boldsymbol{e}$ stems from any random distribution – the term E_2 implies that the noise can be adversarial, which is generally suitable in problems such as parametric PDEs, where the noise stems from the (deterministic) error incurred in the numerical PDE solve. The third term is the *physical discretization error* E_3. Recall that we cannot work directly in the infinite-dimensional space $\mathcal{V}$. This term accounts for the error induced by working in $\mathcal{V}_h$ instead. Since the projection $\mathcal{P}_h(f)(\boldsymbol{y})$ is the best approximation to $f(\boldsymbol{y}) \in \mathcal{V}$ from $\mathcal{V}_h$ in the $\mathcal{V}$-norm, the term E_3 is optimal, up to a minor switch from the $L^2_\varrho(\mathcal{U};\mathcal{V})$-norm on the left-hand side of (17) to the $L^\infty(\mathcal{U};\mathcal{V})$-norm on its right-hand side.

It is notable that the approximation error E_1 achieves a similar exponential rate of decay as the best s-term polynomial approximation error bound (13), except with s replaced by $\sqrt{\widetilde{m}}$ and an additional factor of $1/\sqrt{2}$. Note that the former translates into $\widetilde{m} \propto s^2$, which is precisely the quadratic sample complexity that arises when computing s-sparse polynomial approximations from uniformly-distributed samples via compressed sensing Chkifa et al. (2018); Adcock et al. (2017). Hence, the presence of the term E_1 implies that the DNN procedure performs as well as current best-in-class polynomial approximation schemes based on compressed sensing.

As mentioned, the proof of Theorem 5 is in fact based on re-interpreting such a polynomial-based compressed sensing procedure as a DNN training problem. This is done using a DNN to approximate the Legendre polynomial basis (using an approach due to Opschoor et al. (2019)), and therefore results in a class of DNN architectures $\mathcal{N}$ in which only the weights in the final layer are trained, the remainder being fixed. As shown in step 2 of the proof of Theorem 5, the regularization $\mathcal{J}$ is a certain weighted $\ell^{2,1}$-norm over the weight matrix in the final layer. Note that the training problem may also seem unusual, since it involves fitting in the continuous $\|\cdot\|_{\mathcal{V}}$-norm. However, it can be reformulated as a discrete norm. Let $\boldsymbol{G} = (\langle \varphi_j, \varphi_k \rangle_{\mathcal{V}})_{j,k=1}^K$ be the Gram matrix of the basis $\{\varphi_k\}_{k=1}^K$ of $\mathcal{V}_h$. Then (16) is equivalent to the problem $\min_{\Phi \in \mathcal{N}} \sqrt{\frac{1}{m}\sum_{i=1}^m \|\Phi(\boldsymbol{y}_i) - \boldsymbol{d}_i\|_{\boldsymbol{G},2}^2} + \lambda \mathcal{J}(\Phi)$, where $\boldsymbol{d}_i = (d_{ik})_{k=1}^K \in \mathbb{R}^K$ and $\|\boldsymbol{c}\|_{\boldsymbol{G},2} = \sqrt{\langle \boldsymbol{G}\boldsymbol{c}, \boldsymbol{c} \rangle}$, which involves a standard (weighted) ℓ^2-norm over $\mathbb{R}^K$. However, we also note that the loss function in (16) is is rather different from the standard ℓ^2-loss used commonly in DL, since the data fidelity term is not squared. The reason for this, as discussed in §B.2, is to ensure property (c). If the loss term were squared, the optimal choice regularization parameter λ would depend on the unknown error terms E_1, E_2 and E_3. It is rather remarkable that forgoing the squaring leads to such a desirable property for λ. We are not aware of any existing results in DNN training where the regularization parameter can be set optimally in a function-independent way.

5. Numerical exploration

As commented previously, since Theorem 5 emulates an existing polynomial-based method with a DNN, the resulting training procedure is not expected to give better results in practice. We now present preliminary experiments on the efficiency of training DNNs for approximating solutions to

parametric PDEs from limited data. A more comprehensive study of the efficacy of DL techniques for problems in computational UQ will be presented in an upcoming work. Unlike in Theorem 5, we consider fully-connected DNNs trained using standard (unregularized) loss functions. In addition, differing from previous works – including Geist et al. (2020), which focused on the scaling of accuracy with the dimension d in the presence of large amounts of training data – in this study we investigate the trade-off between accuracy and the number of samples m. We also include a comparison to the best-in-class SCS scheme Dexter et al. (2019) (see §1.3 and §A.4).

5.1. Setup

We approximate a parametric elliptic PDE defined over the spatial domain $\Omega = (0,1)^2$ and parametric domain $\mathcal{U} = [-1,1]^d$ with the uniform probability measure (3). Specifically, given $g \in L^2(\Omega)$ we seek a function $u : \Omega \times \mathcal{U} \to \mathbb{R}$ satisfying

$$-\nabla \cdot (a(\boldsymbol{x},\boldsymbol{y})\nabla u(\boldsymbol{x},\boldsymbol{y})) = g(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \Omega,\ \boldsymbol{y} \in \mathcal{U}, \quad u(\boldsymbol{x},\boldsymbol{y}) = 0,\ \forall \boldsymbol{x} \in \partial\Omega,\ \boldsymbol{y} \in \mathcal{U}. \tag{19}$$

We choose the function g to be independent of $\boldsymbol{y}$ for simplicity. We use a finite element method for spatial discretization, based on a triangulation with $K = 1089$ nodes and meshsize $h = \sqrt{2}/32$. In this study we consider only the error in approximating the parametric PDE, therefore we use the same finite element discretization in generating all of the sample training and testing data, and also in discretization and solution of the SCS problem. See §A.1 for further details.

The training data for the DNNs and for the SCS method is generated by solving (19) at a set of uniform random points $\{\boldsymbol{y}_i\}_{i=1}^{m_{\max}} \subset \mathcal{U}$, yielding a set of solutions $\{u_h(\cdot,\boldsymbol{y}_i)\}_{i=1}^{m_{\max}}$, where $u_h(\cdot,\boldsymbol{y}) \in \mathcal{V}_h$ is the computed solution of (19). Here we examine both the efficiency of DL techniques on fixed subsets of training data of size $m \leq m_{\max}$ and the effect of increasing m up to $m_{\max}$ on the accuracy of the approximations. We also examine the average performance of both methods, running multiple trials training the DNNs and solving the SCS problem given different sets of uniform random points generated with the trial number as the seed. The testing data is generated in the same way by evaluating (19) at a set of points $\{\boldsymbol{y}_i\}_{i=1}^{m_{\text{test}}}$. However instead of using random points to test, we use a deterministic high-order sparse grid stochastic collocation method to generate the set of testing points and data $\{u_h(\cdot,\boldsymbol{y}_i)\}_{i=1}^{m_{\text{test}}}$ (see Nobile et al. (2008)). The sparse grid method is selected due to the superior convergence over standard Monte Carlo integration in evaluating the global testing error metrics, chosen here to be the Bochner norms $L^2_\varrho(\mathcal{U};L^2(\Omega))$ and $L^2_\varrho(\mathcal{U};H^1_0(\Omega))$ (see (4)). See §A.2 for further details.

The DNN strategy is based on §2.2. For $\boldsymbol{y} \in \mathcal{U}$, we define the Finite Element (FE) and DNN approximations by

$$u_h(\boldsymbol{y}) = \sum_{k=1}^{K} c_k(\boldsymbol{y})\varphi_k \quad \text{and} \quad u_{\Phi,h}(\boldsymbol{y}) = \sum_{k=1}^{K} (\Phi(\boldsymbol{y}))_k \varphi_k,\ \Phi : \mathbb{R}^d \to \mathbb{R}^K, \tag{20}$$

respectively. We consider several types of fully-connected DNN architectures with input dimension $N_0 = d$, output dimension $N_{L+2} = K$, and constant hidden-layer widths, i.e. $N_1 = N_2 = \ldots = N_{L+1} = M$ for some $M \in \mathbb{N}$. We follow the convention from Adcock and Dexter (2021) of referring to a DNN with activation function σ, L hidden layers and M nodes per hidden layer as a σ $L \times M$ DNN. In this study we focus on either the $\tanh$, $\rho(x) = \tanh(x)$, the ReLU,

$\rho(x) = \max\{x, 0\}$, or the Leaky-ReLU, $\rho(x) = \max\{x, 0.2x\}$, activation functions.[2] For details on the training approach, see §A.3.

5.2. Effective architectures and loss functions and efficiency of training

We first study the performance of the DL approach based on two different loss functions. The first loss function is the well-known mean square error loss, given in terms of the coefficients (20) as

$$\text{MSE}(\boldsymbol{y}) := \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}(c_k(\boldsymbol{y}_i) - (\Phi(\boldsymbol{y}_i))_k)^2. \tag{21}$$

On the other hand, motivated by the recent work (Geist et al., 2020) we also consider the squared $\mathcal{V}$-norm loss function, which incorporates information about the complexity of the domain in the training procedure. This is defined by

$$\text{MVNSE}(\boldsymbol{y}) := \frac{1}{m}\sum_{i=1}^{m}\|u_h(\boldsymbol{y}_i) - u_{\Phi,h}(\boldsymbol{y}_i)\|_{\mathcal{V}}^2. \tag{22}$$

In Figure 1, we study the efficacy of minimizing either (21) or (22) given solution data from problem (19) with a simple affine coefficient given by

$$a(\boldsymbol{x}, \boldsymbol{y}) = 3 + x_1y_1 + x_2y_2. \tag{23}$$

We make several observations. First, when comparing the training time, identical networks trained with loss (22) take longer to complete 50,000 epochs of training than those trained with loss (21), and the overall time scales poorly with increasing training set size m and increasing number of FE basis elements K for both loss functions. Second, DNN architectures trained with loss function (22) underperform identical architectures trained with the MSE loss in both testing metrics. However, it is interesting that the difference is largest in the case of the $L^2_\varrho(\mathcal{U}; L^2(\Omega))$ testing error, where the best-performing $\tanh$, ReLU, and Leaky-ReLU 5×50 DNNs trained with the MSE loss achieved order 10^{-3} error. Third, we note the non-monotonic decrease in training error for both loss functions, and stagnation in error of some of the larger networks. Figure 2 displays a visualization comparing solutions output by the DNN at a fixed $\boldsymbol{y}$ both early in the training and with a much improved result after achieving order 10^{-6} error in the MSE loss.

5.3. Sample complexity of training

In this section we run a large-scale study of the average testing errors and training times over a range of 10 trials, with increasing number of training samples m up to $m_{\max} = 675$, and testing with a sparse grid quadrature rule with $m_{\text{test}} = 1861$ testing points. For these experiments, we consider a modification of the example from Nobile et al. (2008) of a diffusion coefficient with

2. As noted in Geist et al. (2020), the Leaky-ReLU can help avoid the occurrence of 'dead neurons' in training. Further, (Geist et al., 2020, Remark 3.3) implies that theoretical guarantees such as Theorem 5 also hold if the ReLU is replaced by the Leaky-ReLU, up to a minor change in the architecture of the family of DNNs $\mathcal{N}$.

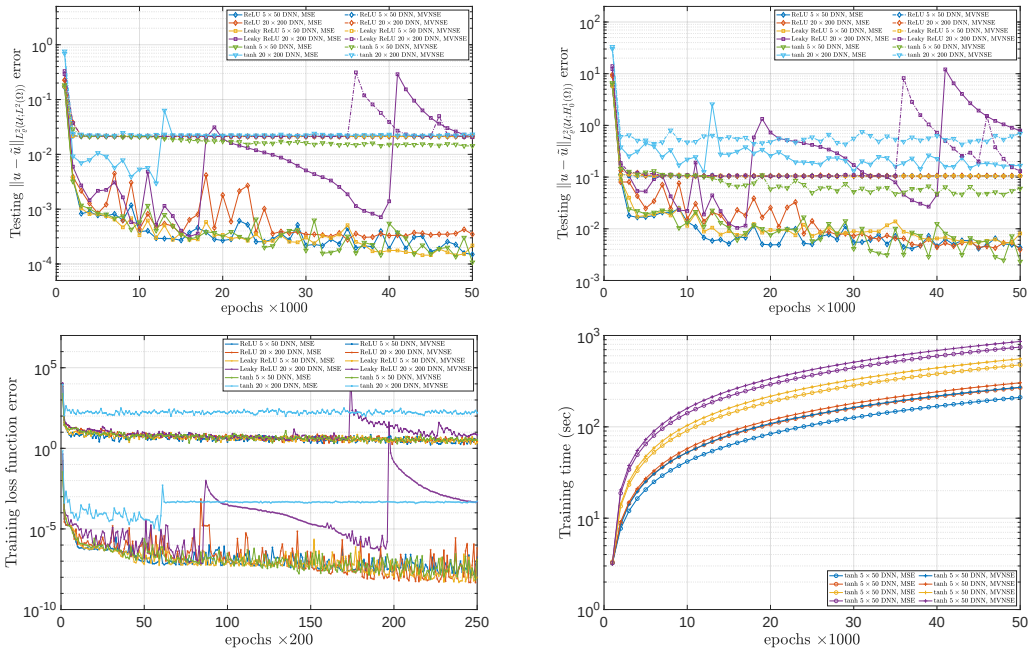

Figure 1: Comparison of **(top-left)** testing error in $L^2_\varrho(\mathcal{U};L^2(\Omega))$, **(top-right)** testing error in $L^2_\varrho(\mathcal{U};H^1_0(\Omega))$, and **(bottom-left)** training error for a variety of DNNs trained with MSE loss function (21) and MVNSE loss (22) on $m = 400$ samples, and **(bottom-right)** training time of a $\tanh$ 5×50 DNN with both loss functions and a range of samples.

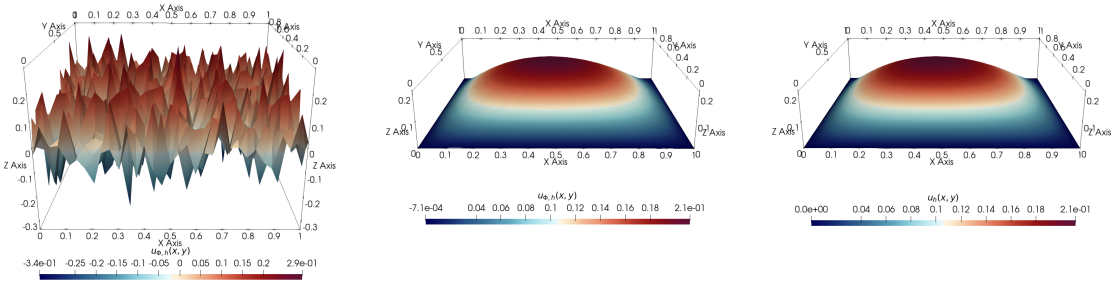

Figure 2: Prediction for $u_h(\boldsymbol{x},\boldsymbol{y})$ from a $\tanh$ 5×50 DNN at $\boldsymbol{y} = [0.995184, 0]^\top$ **(left)** after 2 epochs of Adam (MSE 6.4255) and **(middle)** after training for 2045 epochs (MSE $4.879 \cdot 10^{-7}$), and **(right)** the reference FE solution. At this $\boldsymbol{y}$, $\|u_h(\boldsymbol{y}) - u_{\Phi,h}(\boldsymbol{y})\|_{L^2(\Omega)} = 8.417 \cdot 10^{-4}$ and $\|u_h(\boldsymbol{y}) - u_{\Phi,h}(\boldsymbol{y})\|_{H^1(\Omega)} = 2.315 \cdot 10^{-2}$.

one-dimensional (layered) spatial dependence given by

$$a(\boldsymbol{x},\boldsymbol{y}) = \exp\left(1 + y_1\left(\frac{\sqrt{\pi}\beta}{2}\right)^{1/2} + \sum_{i=2}^{d} \zeta_i\, \vartheta_i(\boldsymbol{x})\, y_i\right) \tag{24}$$

$$\zeta_i := (\sqrt{\pi}\beta)^{1/2} \exp\left(\frac{-\left(\lfloor \frac{i}{2} \rfloor \pi\beta\right)^2}{8}\right), \quad \vartheta_i(\boldsymbol{x}) := \begin{cases} \sin\left(\lfloor \frac{i}{2} \rfloor \pi x_1/\beta_p\right), & \text{if } i \text{ is even,} \\ \cos\left(\lfloor \frac{i}{2} \rfloor \pi x_1/\beta_p\right), & \text{if } i \text{ is odd.} \end{cases}.$$

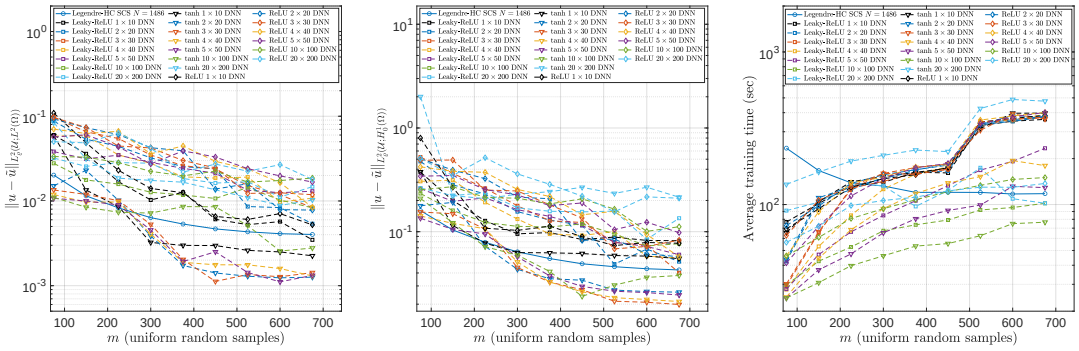

Figure 3: Comparison of average testing errors in **(left)** $L^2_\varrho(\mathcal{U}; L^2(\Omega))$ and **(middle)** $L^2_\varrho(\mathcal{U}; H^1_0(\Omega))$, and **(right)** average training times of the SCS method and various DNN architectures in solving problem (19) with coefficient (24) in $d = 30$ dimension.

Here we let $\beta_c = 1/8$, and $\beta_p = \max\{1, 2\beta_c\}$, $\beta = \beta_c/\beta_p$. We consider this problem with parameter dimension $d = 30$.

Figure 3 displays the result solving problem (19) with coefficient (24) with SCS and a variety of DNN architectures. For the SCS method, we use the Legendre basis and hyperbolic cross of order $p = 6$ with cardinality $N = 1486$. The DNN architectures have depth parameter L and number of hidden layer nodes M chosen so that the ratio $L/M = 0.1$. Due to the non-monotonic decrease in error during training, in this work we employ checkpointing to ensure the parameters achieving the lowest loss are saved and later reloaded for testing, as described in Section A.3. In testing we observe competitive performance using DNNs with the ReLU and Leaky-ReLU activation function. We also observe superior performance over SCS and ReLU and Leaky-ReLU DNNs using DNNs with $\tanh$ activation function, with such networks achieving testing errors on average approximately 3.2 times lower in the $L^2_\varrho(\mathcal{U}; L^2(\Omega))$ error and 2.15 times lower in the $L^2_\varrho(\mathcal{U}; H^1_0(\Omega))$ error than the SCS method given 675 training samples. We also include a comparison of the average training time of the SCS and DNN approaches. While the average times are overall quite similar between the SCS method and training the DNNs, we note that the DNNs are trained on a GPU with accelerated matrix-vector product operations. Therefore this comparison does not provide a good estimate of computational complexity. We leave a study of the computational efficiency of DL techniques for such problems to a future work.

6. Conclusion

In this paper, we first established a novel theoretical result, Theorem 5, asserting the existence of a DNN architecture and training procedure that performs as well as current best-in-class schemes. While this theorem does not explain the success of standard architectures and training, it does highlight the potential of DNNs for holomorphic function approximation. The preliminary numerical results shown above also indicate this promise. The architecture and training differ from that described in Theorem 5, since the networks are much shallower and the loss function simpler. Yet, through a suitable choice of architecture and optimizer, we are not only able to match such performance of current best-in-class schemes using a simpler setup, but also outperform it in this example.

These preliminary results indicate practical promise of the DNN approach, as well as the need for further improvements to the theory so as to address more realistic training scenarios. Results of a more comprehensive study will be reported in a future work.

References

B. Adcock. Infinite-dimensional compressed sensing and function interpolation. *Found. Comput. Math.*, 18(3):661–701, 2018.

B. Adcock and N. Dexter. The gap between theory and practice in function approximation with deep neural networks. *SIAM J. Math. Data Sci. (to appear)*, 2021.

B. Adcock and A. C. Hansen. *Compressive Imaging: Structure, Sampling, Learning*. Cambridge University Press (in press), www.compressiveimagingbook.com, Cambridge, 2021.

B. Adcock, S. Brugiapaglia, and C. G. Webster. Compressed sensing approaches for polynomial approximation of high-dimensional functions. In Holger Boche, Giuseppe Caire, Robert Calderbank, Maximilian März, Gitta Kutyniok, and Rudolf Mathar, editors, *Compressed Sensing and its Applications: Second International MATHEON Conference 2015*, Applied and Numerical Harmonic Analysis, pages 93–124. Birkhäuser, Cham, 2017.

B. Adcock, A. Bao, and S. Brugiapaglia. Correcting for unknown errors in sparse high-dimensional function approximation. *Numer. Math.*, 142(3):667–711, 2019.

S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. The FEniCS Project Version 1.5. *Archive of Numerical Software*, 3 (100), 2015.

F. Bach. Breaking the Curse of Dimensionality with Convex Neural Networks. *J. Mach. Learn. Res.*, 18(19):1–53, 2017.

C. Beck, A. Jentzen, and B. Kuckuck. Full error analysis for the training of deep neural networks. *arXiv:1910.00121*, 2019.

J. Beck, R. Tempone, F. Nobile, and L. Tamellni. On the optimal polynomial approximation of stochastic PDEs by Galerkin and collocation methods. *Math. Models Methods Appl. Sci.*, 22(9): 1250023, 2012.

J. Beck, F. Nobile, L. Tamellini, and R. Tempone. Convergence of quasi-optimal stochastic galerkin methods for a class of pdes with random coefficients. *Comput. Math. Appl.*, 67(4):732–751, 2014.

J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28 – 41, 2018.

J. Berner, P. Grohs, and A. Jentzen. Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations. *SIAM J. Math. Data Sci.*, 2(3):631–657, 2020.

A. Chernov and D. Dũng. New explicit-in-dimension estimates for the cardinality of high-dimensional hyperbolic crosses and approximation of functions having mixed smoothness. *J. Complexity*, 32:92–121, 2016.

A. Chkifa, A. Cohen, R. DeVore, and C. Schwab. Sparse adaptive Taylor approximation algorithms for parametric and stochastic elliptic PDEs. *Modél. Math. Anal. Numér.*, 47(1):253–280, 2013.

A. Chkifa, A. Cohen, and C. Schwab. High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs. *Found. Comput. Math.*, 14(4):601–633, 2014.

A. Chkifa, A. Cohen, and C. Schwab. Breaking the curse of dimensionality in sparse polynomial approximation of parametric PDEs. *J. Math. Pures Appl.*, 103(2):400–428, 2015.

A. Chkifa, N. Dexter, H. Tran, and C. G. Webster. Polynomial approximation via compressed sensing of high-dimensional functions on lower sets. *Math. Comp.*, 87(311):1415–1450, 2018. doi: https://doi.org/10.1090/mcom/3272.

A. Cohen and R. A. DeVore. Approximation of high-dimensional parametric PDEs. *Acta Numer.*, 24:1–159, 2015. doi: 10.1017/S0962492915000033.

A. Cohen and G. Migliorati. Multivariate approximation in downward closed polynomial spaces. In Josef Dick, Frances Y. Kuo, and Henryk Woźniakowski, editors, *Contemporary Computational Mathematics – A Celebration of the 80th Birthday of Ian Sloan*, pages 233–282. Springer, Cham, 2018.

A. Cohen, R. A. DeVore, and C. Schwab. Convergence rates of best N-term Galerkin approximations for a class of elliptic sPDEs. *Foundations of Computational Mathematics*, 10(6):615–646, 2010.

A. Cohen, R. DeVore, and C. Schwab. Analytic regularity and polynomial approximation of parametric and stochastic elliptic PDE's. *Analysis and Applications*, 9(01):11–47, 2011.

G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signals Systems*, 2(4):303–314, 1989.

E. C. Cyr, M. A. Gulian, R. G. Patel, M. Perego, and N. A. Trask. Robust training and initialization of deep neural networks: An adaptive basis viewpoint. In Jianfeng Lu and Rachel Ward, editors, *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*, pages 512–536, Princeton University, Princeton, NJ, USA, 2020. PMLR.

D. Dũng. Private communication, 2020.

N. Dal Santo, S. Deparis, and L. Pegolotti. Data driven approximation of parametrized PDEs by reduced basis and neural networks. *J. Comput. Phys.*, 416:109550, 2020.

P. J. Davis. *Interpolation and approximation*. Courier Corporation, 1975.

J. Daws and C. G. Webster. A Polynomial-Based Approach for Architectural Design and Learning with Deep Neural Networks. *arXiv:1905.10457*, 2019a.

J. Daws and C. G. Webster. Analysis of Deep Neural Networks with Quasi-optimal polynomial approximation rates. *arXiv:1912.02302*, 107(1):1–13, 2019b.

A. Dereventsov, A. Petrosyan, and C. G. Webster. Greedy Shallow Networks: A New Approach for Constructing and Training Neural Networks. *arXiv:1905.10409*, 2019.

R. A. DeVore. Nonlinear approximation. *Acta Numer.*, 7:51–150, 1998.

N. Dexter, H. Tran, and C. Webster. A mixed ℓ_1 regularization approach for sparse simultaneous approximation of parameterized PDEs. *ESAIM Math. Model. Numer. Anal.*, 53:2025–2045, 2019.

A. Doostan and H. Owhadi. A non-adapted sparse approximation of PDEs with stochastic inputs. *J. Comput. Phys.*, 230(8):3015–3034, 2011.

W. E and Wang. Q. Exponential Convergence of the Deep Neural Network Approximation for Analytic Functions. *arXiv:1807.00297*, 2018.

W. E and B. Yu. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Commun. Math. Stat.*, 6(1):1–14, 2018.

W. E, C. Ma, and L. Wu. Barron Spaces and the Compositional Function Spaces for Neural Network Models. *arXiv:1906.08039*, 2019.

F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld. Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error. *Journal of Chemical Theory and Computation*, 13(11):5255–5264, 2017.

Daria Fokina and Ivan Oseledets. Growing axons: greedy learning of neural networks with application to function approximation. *arXiv:1910.12686*, pages 1–17, 2019.

M. Geist, P. Petersen, M. Raslan, R. Schneider, and G. Kutyniok. Numerical solution of the parametric diffusion equation by deep neural networks. *arXiv:2004.12131*, 2020.

C. J. Gittelson. An adaptive stochastic Galerkin method for random elliptic operators. *Math. Comp.*, 82(283):1515–1541, 2013.

P. Grohs, F. Hornung, A. Jentzen, and P. Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *arXiv:1809.02362*, 2018.

P. Grohs, D. Perekrestenko, D. Elbrächter, and H. Bölcskei. Deep neural network approximation theory. *arXiv:1901.02220*, 2019.

I. Gühring, G. Kutyniok, and P. Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *Analysis and Applications*, 18(05):803–859, 2020. doi: 10.1142/S0219530519410021.

M. D. Gunzburger, C. G. Webster, and G. Zhang. Stochastic finite element methods for partial differential equations with random input data. *Acta Numer.*, 23:521–650, 2014.

E. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for ℓ_1-minimization: methodology and convergence. *SIAM J. Optim.*, 19(3):1107–1130, 2008.

J. Hampton and A. Doostan. Compressive sampling of polynomial chaos expansions: convergence analysis and sampling strategies. *J. Comput. Phys.*, 280:363–386, 2015.

M. Hervé. *Analyticity in infinite dimensional spaces*, volume 10. Walter de Gruyter, 2011.

J. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer Briefs in Mathematics. Springer, 2015.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 08936080.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, K. Tunyasuvunakool, O. Ronneberger, R. Bates, A. Zidek, A. Bridgland, C. Meyer, S. A. A. Kohl, A. Potapenko, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, M. Steinegger, M. Pacholska, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. High Accuracy Protein Structure Prediction Using Deep Learning. *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*, 2020.

Y. Khoo, J. Lu, and L. Ying. Solving parametric PDE problems with artificial neural networks. *European J. Appl. Math. (in press)*, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

T. Kühn, W. Sickel, and T. Ullrich. Approximation of mixed order Sobolev functions on the d-torus: asymptotics, preasymptotics, and d-dependence. *Constr. Approx.*, 42:353–398, 2015.

G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric PDEs. *arXiv:1904.00377*, 2020.

F. Laakmann and P. Petersen. Efficient approximation of solutions of parametric linear transport equations by ReLU DNNs. *arXiv:2001.11441*, 2020.

J. Lagergren, J. T. Nardini, G. M. Lavigne, E. M. Rutter, and K. B. Flores. Learning partial differential equations for biological transport models from noisy spatiotemporal data. *Proc. R. Soc. A*, 476(2234):1–21, 2020.

M. Leshno, V. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. doi: https://doi.org/10.1016/S0893-6080(05)80131-5.

B. Li, S. Tang, and H. Yu. Better approximations of high dimensional smooth functions by deep neural networks with rectified power units. *1903.05858*, 2019.

S. Liang and R. Srikant. Why deep neural networks for function approximation? *arXiv:1610.04161*, 2016.

A. Logg and G. N. Wells. DOLFIN: Automated Finite Element Computing. *ACM Transactions on Mathematical Software*, 37(2), 2010. doi: 10.1145/1731022.1731030.

J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep Network Approximation for Smooth Functions. *arXiv:2001.03040*, pages 1–33, 2020.

Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond finite layer neural networks: bridging deep architectures and numerical differential equations. *arXiv:1710.10121*, 2017.

G. Migliorati. Adaptive polynomial approximation by means of random discrete least squares. In Assyr Abdulle, Simone Deparis, Daniel Kressner, Fabio Nobile, and Marco Picasso, editors, *Numerical Mathematics and Advanced Applications - ENUMATH 2013*, pages 547–554, Cham, 2015. Springer.

H. Montanelli and Qiang. Du. New error bounds for deep relu networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1):78–92, 2019. doi: 10.1137/18M1189336.

H. Montanelli, H. Yang, and Q. Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *1903.00735*, 2019.

F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345, 2008.

G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett. Deep Learning Techniques for Inverse Problems in Imaging. *arXiv:2005.06001*, 2020.

J. A. A. Opschoor, Ch. Schwab, and J. Zech. Exponential ReLU DNN expression of holomorphic maps in high dimension. *SAM Research Report*, 2019-35(35), 2019. doi: https://doi.org/10.1142/S0219530518500203.

P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Netw.*, 108:296–330, 2018.

C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Anal. Appl.*, 17(01):19–55, 2019.

Z. Shen, H. Yang, and S. Zhang. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, 28(5):1768–1811, 2020.

M. Stoyanov. User manual: Tasmanian sparse grids. Technical Report ORNL/TM-2015/596, Oak Ridge National Laboratory, One Bethel Valley Road, Oak Ridge, TN, 2015.

M. Stoyanov, D. Lebrun-Grandie, J. Burkardt, and D. Munster. Tasmanian, 9 2013. URL https://github.com/ORNL/Tasmanian.

G. Szegö. *Orthogonal Polynomials*. American Mathematical Society, Providence, RI, 1975.

H. Tran, C. G. Webster, and G. Zhang. Analysis of quasi-optimal polynomial approximations for parameterized PDEs with deterministic and stochastic coefficients. *Numer. Math.*, pages 1–43, 2017.

M. Unser. A representer theorem for deep neural networks. *J. Mach. Learn. Res.*, 20:1–28, 2019.

D. Xiu and G. E. Karniadakis. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(2):619–644, 2002.

D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Netw.*, 94:103–114, 2017.

D. Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. *arXiv:1802.03620*, 2018.

W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM J. Imaging Sci.*, 1(1):143–168, 2008.

G. Zhang, J. Zhang, and J. Hinkle. Learning nonlinear level sets for dimensionality reduction in function approximation. In *Advances in Neural Information Processing Systems 32*, pages 13199–13208. Curran Associates, Inc., 2019.

Appendix A. Further details on the experiments

A.1. Finite element discretization

For the spatial discretization we rely on the finite element method as implemented by Dolfin (Logg and Wells, 2010), and accessed through the python FEniCS project (Alnæs et al., 2015). We generate a regular triangulation $\mathcal{T}_h$ of $\overline{\Omega}$ composed of triangles T of equal diameter $h_T = h$. We consider a conforming discretization, meaning a finite dimensional subspace $\mathcal{V}_h \subset \mathcal{V} := H_0^1(\Omega)$, where $\mathcal{V}_h$ is chosen as the usual Lagrange Finite Elements (FE) of order $k = 1$. We rely on the Dolfin `UnitSquareMesh` method to generate a mesh with 33 nodes per side, corresponding to a finite element triangulation with $K = 1089$ nodes, 2048 elements and meshsize $h = \sqrt{2}/32$. See (Dexter et al., 2019) for further implementation details.

A.2. Testing error

As discussed, rather than using random points to evaluate the test error, we use a high-order stochastic collocation reference solution (see Nobile et al. (2008)) based on a deterministically-generated quadrature rule. Specifically, we apply a Smolyak sparse-grid quadrature rule based on Clenshaw-Curtis points to generate the testing data with the TASMANIAN software package Stoyanov et al. (2013); Stoyanov (2015). In testing, we choose the level ℓ of the sparse grid rule such that $m_{\text{test}} \gg m_{\max}$. The testing error is recorded in the Bochner norms $L^2_\varrho(\mathcal{U}; L^2(\Omega))$ and $L^2_\varrho(\mathcal{U}; H^1_0(\Omega))$ (see (4)) and approximated as the square root of the result of the quadrature formulas

$$\sum_{i=1}^{m_{\text{test}}} \|u_h(\boldsymbol{y}_i) - \tilde{u}_h(\boldsymbol{y}_i)\|^2_{L^2(\Omega)} w_i \quad \text{and} \quad \sum_{i=1}^{m_{\text{test}}} \|u_h(\boldsymbol{y}_i) - \tilde{u}_h(\boldsymbol{y}_i)\|^2_{H^1_0(\Omega)} w_i,$$

respectively, where w_i are the quadrature weights associated with the sparse grid rule and $\tilde{u}_h$ are the approximations obtained with either the DNNs or SCS. When testing multiple trials, we report the average of the errors over all of the trials.

A.3. DNN training

We follow a similar training methodology to that of Adcock and Dexter (2021). All weights and biases of the DNN are initialized as normal random variables with mean 0 and variance 0.01, with the same seed 0 for each network. We perform calculations in single precision, using the Adam optimizer Kingma and Ba (2014) with an exponentially-decaying learning rate and training for 50,000 epochs or until a stopping tolerance of 5×10^{-7} is met. We also employ checkpointing, saving the DNN weights and biases once the error has decreased to $1/16$th the error of the previous checkpoint, and keeping the configuration which provided the lowest training error. Due to the large size of the training data, we use a batch size of $\min\{m, 256\}$ for each training set of size m. As discussed, the loss function is chosen either as the MSE loss (21) or the MVNSE loss (22).

A.4. The SCS method

The SCS method uses multivariate orthonormal Legendre polynomial approximation as described in §3.2 and Appendix B.1, with the hyperbolic cross index set $\Lambda = \Lambda_{s-1}^{\mathrm{HC}}$ as in (26). Given the measurement matrix $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ from (27) and measurement vector $\boldsymbol{b} = \frac{1}{\sqrt{m}}(u_h(\boldsymbol{y}_i))_{i=1}^m \in \mathcal{V}_h^m$, we solve the LASSO problem

$$\min_{\boldsymbol{z}\in\mathcal{V}_h^N} \lambda\|\boldsymbol{z}\|_{\mathcal{V},1} + \|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_{\mathcal{V},2}^2, \tag{25}$$

using a combination of Bregman iterations Yin et al. (2008) and fixed point continuation for ISTA Hale et al. (2008). The full implementation details, including choice of parameters for the solvers and value of λ, can be found in Dexter et al. (2019).

Appendix B. Proof of Theorem 5

B.1. Formulation as a vector recovery problem

Following §3.2, we first consider approximating f using the tensor Legendre polynomial basis $\{\Psi_{\boldsymbol{\nu}}\}_{\boldsymbol{\nu}\in\mathbb{N}_0^d}$. Write

$$f = \sum_{\boldsymbol{\nu}\in\mathbb{N}_0^d} c_{\boldsymbol{\nu}}\Psi_{\boldsymbol{\nu}}, \qquad c_{\boldsymbol{\nu}} = \int_{\mathcal{U}} f(\boldsymbol{y})\Psi_{\boldsymbol{\nu}}(\boldsymbol{y})2^{-d}\,\mathrm{d}\boldsymbol{y} \in \mathcal{V},$$

and let $\boldsymbol{c} = (c_{\boldsymbol{\nu}})_{\boldsymbol{\nu}\in\mathbb{N}_0^d} \in \ell^2(\mathbb{N}_0^d;\mathcal{V})$ be the infinite vector of coefficients of f. Fix $s \in \mathbb{N}$ and let

$$\Lambda = \Lambda_{s-1}^{\mathrm{HC}} = \left\{\boldsymbol{\nu} = (\nu_k)_{k=1}^d \in \mathbb{N}_0^d : \prod_{k=1}^d (\nu_k + 1) \leq s\right\} \subset \mathbb{N}_0^d, \tag{26}$$

be the hyperbolic cross index set of index $s-1$. Let $N = |\Lambda|$ and $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_N$ be an indexing of the multi-indices in Λ and define the normalized measurement matrix.

$$\boldsymbol{A} = \left(\frac{\Psi_{\boldsymbol{\nu}_j}(\boldsymbol{y}_i)}{\sqrt{m}}\right)_{i,j=1}^{m,N} \in \mathbb{R}^{m\times N}. \tag{27}$$

We also define the the normalized measurement and error vectors

$$\boldsymbol{b} = \frac{1}{\sqrt{m}} \left(f(\boldsymbol{y}_i) + n_i\right)_{i=1}^m \in \mathcal{V}_h^m, \quad \text{and} \quad \boldsymbol{e} = \frac{1}{\sqrt{m}} (n_i)_{i=1}^m \in \mathcal{V}^m.$$

Now define

$$f_\Lambda = \sum_{\boldsymbol{\nu} \in \Lambda} c_{\boldsymbol{\nu}} \Psi_{\boldsymbol{\nu}}, \tag{28}$$

as the truncated expansion of f based on the index set Λ and

$$\boldsymbol{c}_\Lambda = (c_{\boldsymbol{\nu}_j})_{j=1}^N \in \mathcal{V}^N,$$

as the finite vector of coefficients of f with indices in Λ. Then we have

$$\boldsymbol{A}\boldsymbol{c}_\Lambda = \frac{1}{\sqrt{m}} \left(f_\Lambda(\boldsymbol{y}_i)\right)_{i=1}^m = \frac{1}{\sqrt{m}} (f(\boldsymbol{y}_i))_{i=1}^m - \frac{1}{\sqrt{m}} \left(f(\boldsymbol{y}_i) - f_\Lambda(\boldsymbol{y}_i)\right)_{i=1}^m,$$

and therefore

$$\boldsymbol{A}\boldsymbol{c}_\Lambda + \boldsymbol{e} + \boldsymbol{e}' = \boldsymbol{b}, \tag{29}$$

where

$$\boldsymbol{e}' = \frac{1}{\sqrt{m}} \left(f(\boldsymbol{y}_i) - f_\Lambda(\boldsymbol{y}_i)\right)_{i=1}^m.$$

Hence, the recovery of the coefficients $\boldsymbol{c}_\Lambda$ of f is equivalent to solving the noisy linear system of equations (29).

B.2. Hilbert-valued compressed sensing

To solve this, we consider techniques from *Hilbert-valued* compressed sensing Dexter et al. (2019). In classical (scalar-valued) compressed sensing, one aims to solve (29) by finding a solution with minimal ℓ^1-norm. In Hilbert-valued compressed sensing, we use the $\ell^1(\Lambda; \mathcal{V})$-norm instead. Both classical and Hilbert-valued compressed sensing are usually formulated as either a quadratically-constrained basis pursuit or LASSO problem. However, we instead consider the *Square Root LASSO (SR-LASSO)* problem

$$\min_{\boldsymbol{z} \in \mathcal{V}_h^N} \lambda \|\boldsymbol{z}\|_{\mathcal{V},1} + \|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_{\mathcal{V},2}, \tag{30}$$

where $\lambda > 0$ is a parameter. This is based on ideas of Adcock et al. (2019). While the other approaches are more common, they have the undesirable feature that the optimal choices for their parameters (in the sense that they give the best error bounds) depend on the magnitude of the terms $\boldsymbol{e}$ and $\boldsymbol{e}'$. These terms are unknown (in particular, they typically depend on f). Hence, they would not give rise to a result such as Theorem 5, where the parameter choice is independent of f. Fortunately, as shown in Adcock et al. (2019), the SR-LASSO (30) has such a desirable property.

The theory of Hilbert-valued compressed sensing has been developed in Dexter et al. (2019). We recall several key definitions. The *support* of a Hilbert-valued vector $\boldsymbol{x} = (x_i)_{i=1}^N \in \mathcal{V}^N$ is

$$\operatorname{supp}(\boldsymbol{x}) = \{i : \|x_i\|_{\mathcal{V}} \neq 0\} \subseteq \{1, \dots, N\}.$$

A Hilbert-valued vector is *s-sparse* if

$$\|\boldsymbol{x}\|_{\mathcal{V},0} := |\operatorname{supp}(\boldsymbol{x})| \leq s.$$

We write $\Sigma_s \subseteq \mathcal{V}^N$ for the set of s-sparse Hilbert-valued vectors. For nonsparse vectors, we also define the ℓ^p-norm *best s-term approximation error* as

$$\sigma_s(\boldsymbol{x})_{\mathcal{V},p} = \inf_{\boldsymbol{z}\in\mathcal{V}^N} \{\|\boldsymbol{x}-\boldsymbol{z}\|_{\mathcal{V},p} : \boldsymbol{z} \in \Sigma_s\}, \quad \boldsymbol{x} \in \mathcal{V}^N.$$

We now focus on properties of the matrix $\boldsymbol{A}$ that ensure recovery of approximately sparse vectors via (30). To this end, we now define some additional notation. Given a set $S \subseteq \{1,\dots,N\}$ we write $P_S : \mathcal{V}^N \to \mathcal{V}^N$ for the projection that, given a vector $\boldsymbol{x} \in \mathcal{V}^N$, produces a vector $P_S(\boldsymbol{x})$ whose ith entry is equal to x_i if $i \in S$ and zero otherwise.

Definition 6 *The matrix $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ satisfies the robust Null Space Property (rNSP) of order $1 \leq s \leq N$ over $\mathcal{V}^N$ with constants $0 < \rho < 1$ and $\tau > 0$ if*

$$\|P_S(\boldsymbol{x})\|_{\mathcal{V},2} \leq \frac{\rho\|P_{S^c}(\boldsymbol{x})\|_{\mathcal{V},1}}{\sqrt{s}} + \tau\|\boldsymbol{A}\boldsymbol{x}\|_{\mathcal{V},2}, \quad \forall \boldsymbol{x} \in \mathcal{V}^N,$$

for any $S \subseteq [N]$ with $|S| \leq s$.

See (Dexter et al., 2019, Defn. 4.1). In (Dexter et al., 2019, Prop. 4.2) the authors show that the rNSP is sufficient to provide an error bound for minimizers of the quadratically-constrained basis pursuit problem. We need an analogous result for the SR-LASSO problem (30). The following is a straightforward extension of (Adcock and Hansen, 2021, Thm. 6.4)[3] from the scalar to the Hilbert-valued case.

Lemma 7 *Suppose that $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ has the rNSP of order $1 \leq s \leq N$ with constants $0 < \rho < 1$ and $\tau > 0$. Let $\boldsymbol{x} \in \mathcal{V}^N$, $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} \in \mathcal{V}^m$ and*

$$\lambda \leq \frac{C_1}{C_2\sqrt{s}},$$

where $C_1 = \frac{(3\rho+1)(\rho+1)}{2(1-\rho)}$ and $C_2 = \frac{(3\rho+5)\tau}{2(1-\rho)}$. Then every minimizer $\hat{\boldsymbol{x}} \in \mathcal{V}^N$ of the Hilbert-valued SR-LASSO problem

$$\min_{\boldsymbol{z}\in\mathcal{V}^N} \lambda\|\boldsymbol{z}\|_{\mathcal{V},1} + \|\boldsymbol{A}\boldsymbol{z}-\boldsymbol{b}\|_{\mathcal{V},2},$$

satisfies

$$\|\hat{\boldsymbol{x}}-\boldsymbol{x}\|_{\mathcal{V},2} \leq 2C_1\frac{\sigma_s(\boldsymbol{x})_{\mathcal{V},1}}{\sqrt{s}} + \left(\frac{C_1}{\sqrt{s}\lambda} + C_2\right)\|\boldsymbol{e}\|_{\mathcal{V},2}.$$

Now we recall that a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ satisfies the *Restricted Isometry Property (RIP)* of order $1 \leq s \leq N$ with constant $0 < \delta_s < 1$ if δ_s is the smallest constant δ for which

$$(1-\delta)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta)\|\boldsymbol{x}\|_2^2, \quad \forall \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{x} \text{ } s\text{-sparse}.$$

The following result connects the RIP and rNSP. It is based on a combination of (Dexter et al., 2019, Prop 4.3) (which applies to the Hilbert-valued case) and (Adcock and Hansen, 2021, Lem. 5.17) (which gives explicit values for ρ and τ):

3. Chapter available online at www.compressiveimagingbook.com.

Lemma 8 *Suppose that $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ satisfies the RIP of order $2s$ with constant $\delta_{2s} < \sqrt{2}-1$. Then $\boldsymbol{A}$ satisfies the rNSP of order s over $\mathcal{V}^N$ with constants ρ and τ given by*

$$\rho = \frac{\sqrt{2}\delta_{2s}}{1-\delta_{2s}}, \qquad \tau = \frac{\sqrt{1+\delta_{2s}}}{1-\delta_{2s}}.$$

With this in mind, we end this section with a result asserting that conditions under which the matrix defined in (27) satisfies the RIP:

Lemma 9 *Let $\{\Psi_{\boldsymbol{\nu}}\}_{\boldsymbol{\nu}\in\mathbb{N}_0^d}$ be the orthonormal tensor Legendre polynomial basis of $L^2_{\varrho}(\mathcal{U})$ and $\boldsymbol{y}_1,\dots,\boldsymbol{y}_m$ be drawn independently from the uniform measure on $\mathcal{U}$. Let $0<\delta,\varepsilon<1$ and suppose that*

$$m \geq c\cdot 2^d\cdot s^2\cdot\log(2s)\cdot\left(\log(2s)\cdot\min\{\log(s)+d,\log(2d)\cdot\log(2s)\}+\log(\varepsilon^{-1})\right),$$

where $c>0$ is a universal constant. Then, with probability at least $1-\varepsilon$, the matrix $\boldsymbol{A}$ defined in satisfies the RIP of order s with constant $\delta_s \leq 1/4$.

Note that the choice of $1/4$ here is arbitrary. Any value less than $\sqrt{2}-1\approx 0.41$ will suffice.

Proof Theorem 2.2 of Chkifa et al. (2018) implies that any matrix of this form satisfies the RIP of order s with constant $\delta_s \leq 1/4$, provided (after simplifying)

$$m \gtrsim \Theta^2\cdot s\cdot\log(2\Theta^2 s)\cdot\max\left\{\log(2\Theta^2 s\log(2\Theta^2 s))\cdot\log(2N),\log(2\varepsilon^{-1}\log(2\Theta^2 s))\right\}, \quad (31)$$

where $N=|\Lambda|$ and Θ is defined by

$$\Theta = \max_{\boldsymbol{\nu}\in\Lambda}\|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}.$$

The Legendre polynomials attain their maxima at $\boldsymbol{y}=\boldsymbol{1}$ (see, for example, Szegö (1975)) and, due to the normalization with respect to the uniform measure, satisfy $|\Psi_{\boldsymbol{\nu}}(\boldsymbol{1})| = \prod_{k=1}^d\sqrt{2\nu_k+1}$. Hence, since Λ is the hyperbolic cross of index $s-1$ (recall (26)), we have

$$\Theta^2 \leq \max\left\{\prod_{k=1}^d(2\nu_k+1) : \prod_{k=1}^d(\nu_k+1)\leq s\right\} \leq 2^d s.$$

Furthermore, it was also show in (Chkifa et al., 2018, Lem. 3.5) that $\Theta^2 \leq s^{\log(3)/\log(2)}$. Substituting this into the above expression and using the fact that

$$\log(2\Theta^2 s) \leq \log(2)+(\log(3)/\log(2)+1)\log(s) \lesssim \log(2s) \leq 2s.$$

now shows that (31) is implied by

$$m \gtrsim 2^d\cdot s^2\cdot\log(2s)\cdot\left(\log(2s)\cdot\log(2N)+\log(\varepsilon^{-1})\right). \quad (32)$$

Furthermore, it can be shown that

$$N \leq \min\left\{2s^3 4^d, \mathrm{e}^2 s^{2+\log(d)/\log(2)}, \frac{s(\log(s)+d\log(2))^{d-1}}{(d-1)!}\right\}. \quad (33)$$

The first and third bounds are due to Theorems 3.7 and 3.5 of Chernov and Dũng (2016) respectively with values $s = d$, $a = 1$ and $T = s$ (note that there is a small typo in the statement of Theorem 3.5 of Chernov and Dũng (2016): the denominator should read $\ln T - s\ln(a - 1/2) + s - 1$ Dũng (2020)). The second bound is due to (Kühn et al., 2015, Thm. 4.9) (note that, although (Kühn et al., 2015, Thm. 4.9) has the condition $s \leq 2^d$ among its assumptions, an inspection of the proof reveals that this assumption is not necessary). In particular,

$$\begin{aligned}\log(2N) &\leq \min\left\{\log(8s^3) + d\log(4), (2 + \log(d)/\log(2))\log(s) + 2\right\} \\ &\lesssim \min\{\log(s) + d, \log(2d)\cdot\log(2s)\}.\end{aligned}$$

Substituting this into (32) now gives the result. ∎

B.3. Polynomial approximation error bounds for holomorphic functions

In order to establish exponential rates of convergence, we need the following result regarding estimates for the error of the best s-term polynomial approximation of a holomorphic function f. For this next result, we recall that a set $S \subseteq \mathbb{N}_0^d$ is *lower* if whenever $\boldsymbol{\nu} \in S$ then $\boldsymbol{\mu} \in S$ for all multi-indices $\boldsymbol{\mu}$ satisfying $\boldsymbol{\mu} \leq \boldsymbol{\nu}$.

Theorem 10 *Let $d \in \mathbb{N}$, $f : \mathcal{U} \to \mathcal{V}$ be holomorphic in a Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}}$ for some $\boldsymbol{\rho} > \mathbf{1}$. Then for every $\epsilon > 0$ there exists $\bar{s} = \bar{s}(d, \epsilon, \boldsymbol{\rho})$ such that, for every $s \geq \bar{s}$, there is a lower set $S \subset \mathbb{N}_0^d$ of size $|S| \leq s$ for which*

$$\begin{aligned}\|f - f_S\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq \|f - f_S\|_{L^\infty(\mathcal{U};\mathcal{V})} &\leq \sum_{\boldsymbol{\nu}\notin S} \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}} \\ &\leq \|f\|_{L^\infty(\mathcal{E}_{\boldsymbol{\rho}};\mathcal{V})} \exp\left(-\frac{1}{d+1}\left(\frac{sd!\prod_{j=1}^d \log(\rho_j)}{1+\epsilon}\right)^{1/d}\right).\end{aligned}$$

Furthermore, the same bound also applies to the coefficient error $\|\boldsymbol{c} - \boldsymbol{c}_S\|_{1,\mathcal{V}}$, where $\boldsymbol{c} = (c_{\boldsymbol{\nu}})_{\boldsymbol{\nu}\in\mathbb{N}_0^d}$ is the sequence of coefficients of f as in (10) and $\boldsymbol{c}_S$ is the infinite sequence with $\boldsymbol{\nu}$th entry equal to $c_{\boldsymbol{\nu}}$ if $\boldsymbol{\nu} \in S$ and zero otherwise.

Note that this result immediately implies (11). In fact, it is stronger, since it asserts an L^∞-norm bound, and shows that this can be achieved by a set that is also lower.

Proof The proof is mainly based on (Opschoor et al., 2019, Theorem 3.5). We start by proving the theorem in the scalar-valued case, i.e. for $\mathcal{V} = \mathbb{R}$. Note that in this case the coefficients $c_{\boldsymbol{\nu}} \in \mathbb{R}$. Inspecting the proof of (Opschoor et al., 2019, Theorem 3.5) we see that, for any given $\tau \in (0, 1)$, choosing the multi-index set as

$$S = S_\tau := \left\{\boldsymbol{\nu} \in \mathbb{N}_0^d : \boldsymbol{\rho}^{-\boldsymbol{\nu}} \geq \tau\right\},$$

where $\boldsymbol{\rho}^{-\boldsymbol{\nu}} := \prod_{j=1}^d \rho_j^{-\nu_j}$, leads to the upper bound

$$\sum_{\boldsymbol{\nu}\notin S_\tau} \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}|c_{\boldsymbol{\nu}}| \leq C\,\|f\|_{L^\infty(\mathcal{E}_{\boldsymbol{\rho}})}\,\tau\,(1 + \log(1/\tau))^{2d}, \tag{34}$$

where $C = C(d, \boldsymbol{\rho}) > 0$ is a constant depending only on d and $\boldsymbol{\rho}$. (Specifically, using the notation of (Opschoor et al., 2019, Theorem 3.5), (34) is obtained by letting $\varepsilon = \tau$ and $k = 0$.)

The next step is to convert the upper bound (34) into an exponential best s-term decay rate. Let us consider the right-hand side of (34). For any $\alpha \in (0,1)$, we have

$$\lim_{\tau \to 0^+} \frac{\tau(1+\log(1/\tau))^{2d}}{\tau^{\alpha}} = 0.$$

Therefore, for any $\alpha \in (0,1)$, we choose the largest $\bar{\tau} = \bar{\tau}(\alpha, d, \boldsymbol{\rho}) \in (0,1)$ such that

$$\tau(1+\log(1/\tau))^{2d} \leq \frac{\tau^{\alpha}}{C \exp\left(\alpha \sum_{j=1}^{d} \log(\rho_j)\right)}, \quad \forall 0 < \tau \leq \bar{\tau}.$$

Combining the above inequality with (34) yields

$$\sum_{\boldsymbol{\nu} \notin S_{\tau}} \|\Psi_{\boldsymbol{\nu}}\|_{L^{\infty}(\mathcal{U})} |c_{\boldsymbol{\nu}}| \leq \frac{\tau^{\alpha}}{\exp\left(\alpha \sum_{j=1}^{d} \log(\rho_j)\right)}, \quad \forall 0 < \tau \leq \bar{\tau}. \tag{35}$$

Now, following (Adcock and Dexter, 2021, Theorem 5.2), we establish a direct link between the parameter $\tau \in (0,1)$ and the sparsity $s \in \mathbb{N}$. Indeed, for any $s \geq 2$ there exists only one value $\tau = \tau(s) \in (0,1)$ such that

$$s = \prod_{j=1}^{d} \left(\frac{\log(1/\tau)}{\log(\rho_j)} + 1 \right), \tag{36}$$

and that

$$\frac{s}{(d+1)^d} \leq |S_{\tau(s)}| \leq s.$$

Observing that (36) defines a monotone decreasing relation between $\tau(s)$ and s, we can find the minimum value $\bar{s} = \bar{s}(\alpha, d, \boldsymbol{\rho}) \in \mathbb{N}$ with $\bar{s} \geq 2$, such that for every $s \geq \bar{s}$, we have $\tau(s) \leq \bar{\tau}$. In this way, we have

$$\sum_{\boldsymbol{\nu} \notin S_{\tau(s)}} \|\Psi_{\boldsymbol{\nu}}\|_{L^{\infty}(\mathcal{U})} |c_{\boldsymbol{\nu}}| \leq \frac{\tau(s)^{\alpha}}{\exp\left(\alpha \sum_{j=1}^{d} \log(\rho_j)\right)}, \quad \forall s \geq \bar{s}.$$

Now, we observe that the cardinality of S_{τ} can be explicitly bounded as in (Opschoor et al., 2019, Lemma 3.3) using a volumetric argument (note that S_{τ} corresponds to Λ_{ε} in Opschoor et al. (2019)). This cardinality bound can be written as

$$\tau \leq \exp\left(\sum_{j=1}^{d} \log(\rho_j) - \left(|S_{\tau}| d! \prod_{j=1}^{d} \log(\rho_j) \right)^{\frac{1}{d}} \right), \quad \forall \tau \in (0,1).$$

Combining the above inequalities and recalling that $|S_{\tau(s)}| \geq s/(d+1)^d$, we obtain that, for any $s \geq \bar{s}$,

$$\begin{aligned} \sum_{\boldsymbol{\nu} \notin S_{\tau(s)}} \|\Psi_{\boldsymbol{\nu}}\|_{L^{\infty}(\mathcal{U})} |c_{\boldsymbol{\nu}}| &\leq \exp\left(-\alpha \left(|S_{\tau(s)}| d! \prod_{j=1}^{d} \log(\rho_j) \right)^{\frac{1}{d}} \right) \\ &\leq \exp\left(-\frac{\alpha}{d+1} \left(s d! \prod_{j=1}^{d} \log(\rho_j) \right)^{\frac{1}{d}} \right). \end{aligned}$$

Finally, we let $\alpha = 1/(1+\epsilon)^{1/d}$ and observe that $\|f - f_S\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq \|f - f_S\|_{L^\infty(\mathcal{U};\mathcal{V})} \leq \sum_{\boldsymbol{\nu}\notin S} \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}|c_{\boldsymbol{\nu}}|$. This concludes the proof in the case $\mathcal{V} = \mathbb{R}$.

This proof can be generalized to the Hilbert-valued case by replacing coefficients' magnitude $|c_{\boldsymbol{\nu}}|$ with coefficients' norm $\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}}$. With this modification, the analogous of (34) holds. Indeed, the proof of (34) given in (Opschoor et al., 2019, Theorem 3.5) relies on coefficient bounds for Legendre polynomials that hold in the Hilbert-valued case as well (see, e.g., (Cohen and DeVore, 2015, Lemma 3.15). The rest of the argument is identical to the scalar-valued case. We also observe that the exponential bound holds for $\|\boldsymbol{c} - \boldsymbol{c}_S\|_{1,\mathcal{V}}$ because $\|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})} \geq 1$ for every $\boldsymbol{\nu} \in \mathbb{N}_0^d$. This completes the proof. ■

B.4. Proof of Theorem 5

We are now ready to prove the main result. Our strategy is based on reformulating the above compressed sensing formulation as a DNN training problem. We first require the following result (Opschoor et al., 2019, Prop. 2.13):

Proposition 11 *There exists a universal constant $c > 0$ such that the following holds. For every finite subset $\Lambda \subset \mathbb{N}_0^d$ and every $0 < \delta < 1$ there exists a ReLU neural network $\Phi_{\Lambda,\delta} : \mathbb{R}^d \to \mathbb{R}^{|\Lambda|}$ such that, if $\Phi_{\Lambda,\delta} = (\Phi_{\boldsymbol{\nu},\delta})_{\boldsymbol{\nu}\in\Lambda}$, then*

$$\|\Psi_{\boldsymbol{\nu}} - \Phi_{\boldsymbol{\nu},\delta}\|_{L^\infty(\mathcal{U})} \leq \delta, \quad \forall \boldsymbol{\nu} \in \Lambda.$$

The depth and size of this network satisfy

$$\begin{aligned}\operatorname{depth}(\Phi) &\leq c\cdot(1+d\log(d))\cdot(1+\log(m(\Lambda)))\cdot(m(\Lambda)+\log(\delta^{-1})),\\ \operatorname{size}(\Phi) &\leq c\cdot(d^2 m(\Lambda)^2 + dm(\Lambda)\log(\delta^{-1})) + d^2\cdot|\Lambda|\cdot(1+\log(m(\Lambda))+\log(\delta^{-1})),\end{aligned}$$

where $m(\Lambda) = \max_{\boldsymbol{\nu}\in\Lambda}\|\boldsymbol{\nu}\|_1 = \max_{\boldsymbol{\nu}\in\Lambda}\sum_{j=1}^d \nu_j$.

The general idea of the proof is to use Proposition 11 to approximate the matrix-vector multiplication $\boldsymbol{A}\boldsymbol{z}$ – which is a polynomial evaluated at the sample points $\boldsymbol{y}_i$ – as a neural network Φ evaluated at the sample points, or equivalently, to approximate the matrix $\boldsymbol{A}$, which is built from polynomials, as a matrix $\boldsymbol{A}'$ built from DNNs. We then use compressed sensing results applied to $\boldsymbol{A}'$ to establish an error bound. Since this process commits an error, we first require the following result, which shows that the rNSP is robust to small matrix perturbations:

Lemma 12 *Suppose that $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ has the rNSP of order $1 \leq s \leq N$ over $\mathcal{V}^N$ with constants $0 < \rho < 1$ and $\tau > 0$. Let $\boldsymbol{A}' \in \mathbb{R}^{m\times N}$ be such that $\|\boldsymbol{A} - \boldsymbol{A}'\|_2 \leq \delta$, where*

$$0 \leq \delta < \frac{1-\rho}{\tau(\sqrt{s}+1)}.$$

Then $\boldsymbol{A}'$ has the rNSP of order s over $\mathcal{V}^N$ with constants $0 < \rho' < 1$ and $\tau' > 0$ given by

$$\rho' = \frac{\rho + \tau\delta\sqrt{s}}{1-\tau\delta}, \quad \tau' = \frac{\tau}{1-\tau\delta}.$$

This is a straightforward extension of (Adcock and Hansen, 2021, Lemma 8.5) from the scalar-valued case to the Hilbert-valued case. We give the proof for completeness:

Proof Let $\boldsymbol{z} \in \mathcal{V}^N$ and a set $S \in [N]$ such that $|S| \leq s$. Then, since A has the rNSP,

$$\begin{aligned}\|P_S(\boldsymbol{z})\|_{\mathcal{V},2} &\leq \frac{\rho}{\sqrt{s}}\|P_{S^c}(\boldsymbol{z})\|_{\mathcal{V},1} + \tau\|\boldsymbol{A}\boldsymbol{z}\|_{\mathcal{V},2} \\ &\leq \frac{\rho}{\sqrt{s}}\|P_{S^c}(\boldsymbol{z})\|_{\mathcal{V},1} + \tau\|\boldsymbol{A}'\boldsymbol{z}\|_{\mathcal{V},2} + \tau\delta\|\boldsymbol{z}\|_{\mathcal{V},2} \\ &\leq \frac{\rho}{\sqrt{s}}\|P_{S^c}(\boldsymbol{z})\|_{\mathcal{V},1} + \tau\|\boldsymbol{A}'\boldsymbol{z}\|_{\mathcal{V},2} + \tau\delta\|P_S(\boldsymbol{z})\|_{\mathcal{V},2} + \tau\delta\|P_{S^c}(\boldsymbol{z})\|_{\mathcal{V},1}.\end{aligned}$$

Rearranging, we get

$$(1-\tau\delta)\|P_S(\boldsymbol{z})\|_{\mathcal{V},2} \leq \frac{\rho+\tau\delta\sqrt{s}}{\sqrt{s}}\|P_{S^c}(\boldsymbol{z})\|_{\mathcal{V},1} + \tau\|\boldsymbol{A}'\boldsymbol{z}\|_{\mathcal{V},2}.$$

The result now follows immediately from the assumptions on δ. ■

Proof (of Theorem 5) We divide the proof into several steps:

Step 1: We first define the class of DNNs $\mathcal{N}$. Let $\widetilde{m}$ be as in (15) and define

$$s := \left\lceil \left(\widetilde{m}/2^d\right)^{1/2} \right\rceil. \tag{37}$$

Now let $\Lambda = \Lambda_{s-1}^{\mathrm{HC}}$ and $\Phi_{\Lambda,\delta}$ be as in Proposition 11. We will choose δ in Step 5. Letting $N = |\Lambda|$, we define the class $\mathcal{N}$ as

$$\mathcal{N} = \left\{ \Phi : \mathbb{R}^d \to \mathbb{R}^K : \Phi(\boldsymbol{y}) = \boldsymbol{Z}^\top \Phi_{\Lambda,\delta}(\boldsymbol{y}),\ \boldsymbol{Z} \in \mathbb{R}^{N\times K} \right\}.$$

Note that $\boldsymbol{Z}$ is the matrix of trainable parameters (it is the weight matrix in the output layer). We will establish the size, depth and number of trainable parameters of this class (part (a) of Theorem 5) in Step 6.

Step 2: We next show that (16) can be reinterpreted as a SR-LASSO minimization problem. Analogously to (27), we define the matrix $\boldsymbol{A}'$ as

$$\boldsymbol{A}' = \left(\frac{\Phi_{\boldsymbol{\nu}_j,\delta}(\boldsymbol{y}_i)}{\sqrt{m}} \right)_{i,j=1}^{m,N} \in \mathbb{R}^{m\times N},$$

where $\Phi_{\boldsymbol{\nu},\delta}$ is the $\boldsymbol{\nu}$th component of $\Phi_{\Lambda,\delta}$. Let $\Phi = \boldsymbol{Z}^\top \Phi_{\Lambda,\delta} \in \mathcal{N}$ and f_Φ be as in (7). Then

$$f_\Phi(\boldsymbol{y}_i) = \sum_{k=1}^K \left(\Phi(\boldsymbol{y}_i)\right)_k \varphi_k = \sqrt{m}\sum_{k=1}^K (\boldsymbol{A}'\boldsymbol{Z})_{ik}\varphi_k = \sqrt{m}\left(\boldsymbol{A}'\boldsymbol{z}\right)_i, \tag{38}$$

where $\boldsymbol{z} = (z_{\boldsymbol{\nu}_j})_{j=1}^N \in \mathcal{V}_h^N$ is the Hilbert-valued vector with $\boldsymbol{\nu}_j$th component $z_{\boldsymbol{\nu}_j} = \sum_{k=1}^K Z_{jk}\varphi_k$. Hence

$$\left(\frac{1}{\sqrt{m}} f_\Phi(\boldsymbol{y}_i) \right)_{i=1}^m = \boldsymbol{A}'\boldsymbol{z}.$$

We now define the regularization function $\mathcal{J}$ as

$$\mathcal{J}(\Phi) = \|\boldsymbol{z}\|_{\mathcal{V},1},$$ [4]

We verify that $\mathcal{J}$ is equivalent to a norm over the trainable parameters $\boldsymbol{Z}$ (part (b) of Theorem 5). Notice that

$$\|\boldsymbol{z}\|_{\mathcal{V},1} = \sum_{j=1}^{N} \|z_{\boldsymbol{\nu}_j}\|_{\mathcal{V}} = \sum_{j=1}^{N} \|\sum_{k=1}^{K} Z_{jk}\varphi_k\|_{\mathcal{V}} = \sum_{j=1}^{N} \|\boldsymbol{Z}^\top \boldsymbol{e}_j\|_{\boldsymbol{G},2} = \|\boldsymbol{G}^{1/2}\boldsymbol{Z}^\top\|_{2,1}$$

where $\boldsymbol{e}_j \in \mathbb{R}^N$ is the jth coordinate vector, $\boldsymbol{G}^{1/2}$ is the unique positive definite square root of $\boldsymbol{G}$ (defined in §4, along with the norm $\|\cdot\|_{\boldsymbol{G},2}$) and $\|\cdot\|_{2,1}$ is the matrix $\ell^{2,1}$-norm, defined in §2. Hence, $\mathcal{J}$ is equivalent to a norm over the trainable parameters.

Using this and (38), we now see that (16) can be expressed as the Hilbert-valued SR-LASSO problem

$$\min_{\boldsymbol{z}\in\mathcal{V}_h^N} \lambda\|\boldsymbol{z}\|_{\mathcal{V},1} + \|\boldsymbol{A}'\boldsymbol{z} - \boldsymbol{b}\|_{\mathcal{V},2}, \tag{39}$$

where $\boldsymbol{b} = \frac{1}{\sqrt{m}}(d_i)_{i=1}^m \in \mathcal{V}_h^m$. We will choose the value of λ in Step 4(ii). In particular, any minimizer $\hat{\boldsymbol{c}} = (\hat{c}_{\boldsymbol{\nu}})_{\boldsymbol{\nu}\in\Lambda}$ of (39) yields a minimizer $\hat{\Phi}$ of (16), given by

$$\hat{\Phi} = \widehat{\boldsymbol{C}}^\top \Phi_{\Lambda,\delta}(\boldsymbol{y}),$$

where $\widehat{\boldsymbol{C}} = \left(\widehat{C}_{jk}\right)_{j,k=1}^{N,K}$ is such that

$$\hat{c}_{\boldsymbol{\nu}_j} = \sum_{k=1}^{K} \widehat{C}_{jk}\varphi_k, \quad \forall j = 1,\dots,N,$$

and vice versa. This completes Step 2.

Step 3: In the third step, we show that the matrix $\boldsymbol{A}'$ has the rNSP of order s over $\mathcal{V}_h^N$ with probability at least $1-\varepsilon$. Since $\widetilde{m} \geq 2^d\bar{s}^2 \geq 2^d$ by assumption, we have

$$s \leq 2(\widetilde{m}/2^d)^{1/2}, \tag{40}$$

and therefore $s \leq 2(m/(2^d\mathcal{L}))^{1/2} \leq m$, since $\mathcal{L} \geq 1$ for a suitable choice of the universal constant c_0. Hence, recalling (14) and (15), we see that

$$m = \widetilde{m}\cdot\mathcal{L}(m,d,\varepsilon) \geq \widetilde{m}\cdot\mathcal{L}(s,d,\varepsilon) \geq 2^d\cdot(s^2/4)\cdot\mathcal{L}(s,d,\varepsilon),$$

and therefore

$$m \gtrsim 2^d\cdot s^2\cdot\log(2s)\cdot\left(\log(2s)\cdot\min\{\log(2s)+d, \log(2s)\cdot\log(2d)\} + \log(\varepsilon^{-1})\right).$$

4. Note that a DNN Φ is not, in general, uniquely defined by its parameters. In this case, this expression may fail to define a well-defined map $\mathcal{J}$. However, this is not a problem since we can define $\mathcal{J}(\Phi)$ in this case as the infimum of $\|\boldsymbol{z}\|_{\mathcal{V},1}$ over all parameters $\boldsymbol{Z}$ that yield the same DNN Φ.

Lemma 9 (after replacing s by $2s$) now implies that the matrix $\boldsymbol{A}$ satisfies the RIP of order $2s$ with constant $\delta_{2s} \leq 1/4$ and Lemma 8 implies that $\boldsymbol{A}$ satisfies the rNSP of order s over $\mathcal{V}_h^N$ with constants

$$\rho = \sqrt{2}/3, \quad \tau = 2\sqrt{5}/3, \tag{41}$$

with probability at least $1-\varepsilon$. To show that $\boldsymbol{A}'$ satisfies the rNSP we use Lemma 12. We first bound $\|\boldsymbol{A} - \boldsymbol{A}'\|_2$. Let $\boldsymbol{z} \in \mathbb{R}^N$. Then we observe that

$$\begin{aligned}
\|(\boldsymbol{A}-\boldsymbol{A}')\boldsymbol{z}\|_2^2 &= \frac{1}{m}\sum_{i=1}^m \left| \sum_{\boldsymbol{\nu}\in\Lambda} \left(\Psi_{\boldsymbol{\nu}}(\boldsymbol{y}_i) - \Phi_{\boldsymbol{\nu},\delta}(\boldsymbol{y}_i)\right) z_{\boldsymbol{\nu}} \right|^2 \\
&\leq \left(\sum_{\boldsymbol{\nu}\in\Lambda} \|\Psi_{\boldsymbol{\nu}}(\boldsymbol{y}_i) - \Phi_{\boldsymbol{\nu},\delta}(\boldsymbol{y}_i)\|_{L^\infty(\mathcal{U})} |z_{\boldsymbol{\nu}}| \right)^2 \\
&\leq \sum_{\boldsymbol{\nu}\in\Lambda} \|\Psi_{\boldsymbol{\nu}} - \Phi_{\boldsymbol{\nu},\delta}\|_{L^\infty(\mathcal{U})}^2 \|\boldsymbol{z}\|_2^2 \leq N\delta^2 \|\boldsymbol{z}\|_2^2,
\end{aligned}$$

by definition of $\Phi_{\boldsymbol{\nu},\delta}$. Hence

$$\|\boldsymbol{A}-\boldsymbol{A}'\|_2 \leq \sqrt{N}\delta.$$

Now suppose that δ satisfies

$$\sqrt{N}\delta \leq \frac{9-4\sqrt{2}}{2\sqrt{5}(3+4\sqrt{s})}, \tag{42}$$

(the choice of δ that will be made in Step 5 will ensure this condition). Using Lemma 12 and the values (41) for ρ and τ we now see that $\boldsymbol{A}'$ has the rNSP of order s over $\mathcal{V}_h^N$ with constants

$$\rho' = \frac{3}{4}, \qquad \tau' = \frac{\sqrt{5}(3+4\sqrt{s})}{2(\sqrt{2}+3\sqrt{s})} \leq \frac{3\sqrt{5}}{2}, \tag{43}$$

with probability at least $1-\varepsilon$.

Step 4: We now derive an error bound for $f - f_{\hat{\Phi}}$, where $\hat{\Phi}$ is a minimizer of (16) and $f_{\hat{\Phi}}$ is as in (7). Throughout, we assume the results of the previous steps. In particular, the matrix $\boldsymbol{A}'$ defined in step 2 has the rNSP of order s (step 3). The problem (39), or equivalently (16), involves three discretizations. Truncation of the infinite expansion via the index set Λ, discretization of the space $\mathcal{V}$ via $\mathcal{V}_h$ and replacement of the polynomial functions by DNNs. First, given $\hat{\Phi} = \widehat{\boldsymbol{C}}^\top \Phi_{\Lambda,\delta}$ we let $\hat{\boldsymbol{c}}$ be the corresponding minimizer of (39) and set

$$f_{\hat{\Psi}} = \sum_{\boldsymbol{\nu}\in\Lambda} \hat{c}_{\boldsymbol{\nu}} \Psi_{\boldsymbol{\nu}}.$$

We now proceed as follows:

$$\begin{aligned}
\|f - f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq& \|f - \mathcal{P}_h(f)\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} + \|\mathcal{P}_h(f) - \mathcal{P}_h(f_\Lambda)\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \\
&+ \|\mathcal{P}_h(f_\Lambda) - f_{\hat{\Psi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} + \|f_{\hat{\Psi}} - f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \\
=:& \|f - \mathcal{P}_h(f)\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} + A_1 + A_2 + A_3.
\end{aligned}$$

Here $\mathcal{P}_h$ is as defined in §2.1 and f_Λ is as in (28). We now bound A_1, A_2 and A_3.

Step 4(i): We commence with A_1. This is elementary. Since $\mathcal{P}_h$ is an orthogonal projection we have $\|\mathcal{P}_h(a)\|_{\mathcal{V}} \leq \|a\|_{\mathcal{V}}$ for $a \in \mathcal{V}$ and therefore

$$A_1 = \|\mathcal{P}_h(f) - \mathcal{P}_h(f_\Lambda)\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq \|f - f_\Lambda\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq \|f - f_\Lambda\|_{L^\infty(\mathcal{U};\mathcal{V})}, \tag{44}$$

recalling that we are considering a probability measure $\mathrm{d}\varrho(\boldsymbol{y})$ over $\mathcal{U}$.

Step 4(ii): We next consider A_2. Observe that

$$\mathcal{P}_h(f_\Lambda) - f_{\hat{\Psi}} = \mathcal{P}_h\left(\sum_{\boldsymbol{\nu}\in\Lambda} c_{\boldsymbol{\nu}}\Psi_{\boldsymbol{\nu}}\right) - \sum_{\boldsymbol{\nu}\in\Lambda} \hat{c}_{\boldsymbol{\nu}}\Psi_{\boldsymbol{\nu}} = \sum_{\boldsymbol{\nu}\in\Lambda} \left(\mathcal{P}_h(c_{\boldsymbol{\nu}}) - \hat{c}_{\boldsymbol{\nu}}\right)\Psi_{\boldsymbol{\nu}}.$$

Define the vector $\mathcal{P}_h(\boldsymbol{c}_\Lambda) = (\mathcal{P}_h(c_{\boldsymbol{\nu}}))_{\boldsymbol{\nu}\in\Lambda}$. Then, by orthonormality of the $\Psi_{\boldsymbol{\nu}}$'s, we have

$$\|\mathcal{P}_h(f_\Lambda) - f_{\hat{\Psi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} = \|\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \hat{\boldsymbol{c}}\|_{\mathcal{V},2}.$$

To bound this term, apply Lemma 7 to the problem (39). This gives

$$\|\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \hat{\boldsymbol{c}}\|_{\mathcal{V},2} \leq 2C_1' \frac{\sigma_s(\mathcal{P}_h(\boldsymbol{c}_\Lambda))_{\mathcal{V},1}}{\sqrt{s}} + \left(\frac{C_1'}{\sqrt{s}\lambda} + C_2'\right)\|\boldsymbol{A}'\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \boldsymbol{b}\|_{\mathcal{V},2},$$

where $C_1' = \frac{(3\rho'+1)(\rho'+1)}{2(1-\rho')}$ and $C_2 = \frac{(3\rho'+5)\tau'}{2(1-\rho')}$ with ρ' and τ' as in (43). This holds provided $\lambda \leq C_1'/(C_2'\sqrt{s})$. Thus, we now set

$$\lambda = \frac{C_1'}{C_2'\sqrt{s}}. \tag{45}$$

Since s is given via (37) in terms of $\widetilde{m}$ and d, we have shown part (c) of Theorem 5. Notice that $C_1', C_2' \asymp 1$ due to the values of ρ' and τ' given by (43). Hence

$$\|\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \hat{\boldsymbol{c}}\|_{\mathcal{V},2} \lesssim \frac{\sigma_s(\mathcal{P}_h(\boldsymbol{c}_\Lambda))_{\mathcal{V},1}}{\sqrt{s}} + \|\boldsymbol{A}'\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \boldsymbol{b}\|_{\mathcal{V},2}.$$

Consider the first term. Let $\pi : \{1,\dots,N\} \to \Lambda$ be a bijection that gives a nonincreasing rearrangement of the sequence $(\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}})_{\boldsymbol{\nu}\in\Lambda}$. Then

$$\sigma_s(\mathcal{P}_h(\boldsymbol{c}_\Lambda))_{\mathcal{V},1} \leq \sum_{i=s+1}^{N} \|\mathcal{P}_h(c_{\boldsymbol{\nu}_{\pi(i)}})\|_{\mathcal{V}} \leq \sum_{i=s+1}^{N} \|c_{\boldsymbol{\nu}_{\pi(i)}}\|_{\mathcal{V}} = \sigma_s(\boldsymbol{c}_\Lambda)_{\mathcal{V},1} \leq \sigma_s(\boldsymbol{c})_{\mathcal{V},1}.$$

Hence,

$$\|\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \hat{\boldsymbol{c}}\|_{\mathcal{V},2} \lesssim \frac{\sigma_s(\boldsymbol{c})_{\mathcal{V},1}}{\sqrt{s}} + \|\boldsymbol{A}'\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \boldsymbol{b}\|_{\mathcal{V},2}. \tag{46}$$

We now estimate the second term. Let $i = 1,\dots,m$ and write

$$\begin{aligned}\sqrt{m}\left(\boldsymbol{A}'\mathcal{P}_h(\boldsymbol{c}_\Lambda) - \boldsymbol{b}\right)_i &= \sum_{\boldsymbol{\nu}\in\Lambda} \mathcal{P}_h(c_{\boldsymbol{\nu}})\Phi_{\boldsymbol{\nu},\delta}(\boldsymbol{y}_i) - f(\boldsymbol{y}_i) - n_i \\ &= \sum_{\boldsymbol{\nu}\in\Lambda} \mathcal{P}_h(c_{\boldsymbol{\nu}})\left(\Phi_{\boldsymbol{\nu},\delta}(\boldsymbol{y}_i) - \Psi_{\boldsymbol{\nu}}(\boldsymbol{y}_i)\right) + \mathcal{P}_h(f_\Lambda)(\boldsymbol{y}_i) - f(\boldsymbol{y}_i) - n_i.\end{aligned}$$

Then

$$\begin{aligned}\|\sqrt{m}(\boldsymbol{A}'\mathcal{P}_h(\boldsymbol{c}_\Lambda)-\boldsymbol{b})_i\|_{\mathcal{V}} &\\ &\leq \sum_{\boldsymbol{\nu}\in\Lambda}\|\mathcal{P}_h(c_{\boldsymbol{\nu}})\|_{\mathcal{V}}\delta+\|\mathcal{P}_h(f_\Lambda)(\boldsymbol{y}_i)-f(\boldsymbol{y}_i)\|_{\mathcal{V}}+\|n_i\|_{\mathcal{V}}\\ &\leq \delta\sum_{\boldsymbol{\nu}\in\Lambda}\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}}+\|\mathcal{P}_h(f_\Lambda)(\boldsymbol{y}_i)-\mathcal{P}_h(f)(\boldsymbol{y}_i)\|_{\mathcal{V}}+\|\mathcal{P}_h(f)(\boldsymbol{y}_i)-f(\boldsymbol{y}_i)\|_{\mathcal{V}}+\|n_i\|_{\mathcal{V}}\\ &\leq \sqrt{N}\delta\|\boldsymbol{c}_\Lambda\|_{\mathcal{V},2}+\|f-f_\Lambda\|_{L^\infty(\mathcal{U};\mathcal{V})}+\|f-\mathcal{P}_h(f)\|_{L^\infty(\mathcal{U};\mathcal{V})}+\|n_i\|_{\mathcal{V}}.\end{aligned}$$

Observing that $\|\boldsymbol{c}_\Lambda\|_{\mathcal{V},2}\leq\|\boldsymbol{c}\|_{\mathcal{V},2}=\|f\|_{L^2_\varrho(\mathcal{U};\mathcal{V})}\leq\|f\|_{L^\infty(\mathcal{U};\mathcal{V})}\leq 1$ by Parseval's identity and the assumption $f\in\mathcal{HA}(\gamma,\epsilon,d)$, we conclude that

$$\|\boldsymbol{A}'\mathcal{P}_h(\boldsymbol{c}_\Lambda)-\boldsymbol{b}\|_{\mathcal{V},2}\leq\sqrt{N}\delta+\|f-f_\Lambda\|_{L^\infty(\mathcal{U};\mathcal{V})}+\|f-\mathcal{P}_h(f)\|_{L^\infty(\mathcal{U};\mathcal{V})}+\|\boldsymbol{e}\|_{\mathcal{V},2},$$

where we recall from (18) that $\boldsymbol{e}$ is defined by $\boldsymbol{e}=\frac{1}{\sqrt{m}}(n_i)_{i=1}^m$. Using the definitions of E_2 and E_3 from (18) and substituting this into (46) now yields

$$A_2=\|\mathcal{P}_h(\boldsymbol{c}_\Lambda)-\hat{\boldsymbol{c}}\|_{\mathcal{V},2}\lesssim\frac{\sigma_s(\boldsymbol{c})_{\mathcal{V},1}}{\sqrt{s}}+\sqrt{N}\delta+\|f-f_\Lambda\|_{L^\infty(\mathcal{U};\mathcal{V})}+E_2+E_3. \tag{47}$$

Step 4(iii): Finally, we consider A_3. We have

$$\|f_{\hat{\Psi}}-f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})}\leq\sum_{\boldsymbol{\nu}\in\Lambda}\|\Psi_{\boldsymbol{\nu}}-\Phi_{\boldsymbol{\nu},\delta}\|_{L^2_\varrho(\mathcal{U})}\|\hat{c}_{\boldsymbol{\nu}}\|_{\mathcal{V}}\leq\delta\|\hat{\boldsymbol{c}}\|_{\mathcal{V},1}.$$

We now use the fact that $\hat{\boldsymbol{c}}$ is a minimizer of (39) to get

$$\lambda\|\hat{\boldsymbol{c}}\|_{\mathcal{V},1}\leq\lambda\|\boldsymbol{0}\|_{\mathcal{V},1}+\|\boldsymbol{A}'\boldsymbol{0}-\boldsymbol{b}\|_{\mathcal{V},2}=\|\boldsymbol{b}\|_{\mathcal{V},2}.$$

where $\boldsymbol{0}\in\mathcal{V}_h^N$ is the zero vector. Using the definition of $\boldsymbol{b}$ (given after equation (39)) and (45), we deduce that

$$\|\hat{\boldsymbol{c}}\|_{\mathcal{V},1}\lesssim\sqrt{s}\left(\|\boldsymbol{e}\|_{\mathcal{V},2}+\|f\|_{L^\infty(\mathcal{U};\mathcal{V})}\right)\leq\sqrt{s}\left(\|\boldsymbol{e}\|_{\mathcal{V},2}+1\right)=\sqrt{s}(E_2+1).$$

Note that (42) implies that $\delta\sqrt{s}\lesssim 1/\sqrt{N}\lesssim 1$. Hence, we conclude that

$$A_3=\|f_{\hat{\Psi}}-f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})}\lesssim\sqrt{s}\delta+E_2. \tag{48}$$

Combining the estimates (44), (47) and (48) and noticing that $s\leq N$ by definition, we deduce that

$$\|f-f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})}\lesssim\|f-f_\Lambda\|_{L^\infty(\mathcal{U};\mathcal{V})}+\frac{\sigma_s(\boldsymbol{c})_{\mathcal{V},1}}{\sqrt{s}}+\sqrt{N}\delta+E_2+E_3. \tag{49}$$

This concludes Step 4.

Step 5: In this penultimate step we use Theorem 10 to estimate the remaining error terms in (49) and give the exact value of δ. Let S be the set defined in Theorem 10. Since $|S|\leq s$ and $\|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}\geq 1$

(this is due to the fact the $\Psi_{\boldsymbol{\nu}}$ have unit L^2_ϱ-norm with respect to the uniform probability measure over $\mathcal{U}$) we have

$$\sigma_s(\boldsymbol{c})_{\mathcal{V},1} \leq \sum_{\boldsymbol{\nu}\notin S} \|c_{\boldsymbol{\nu}}\|_{\mathcal{V}} \leq \sum_{\boldsymbol{\nu}\notin S} \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}}.$$

Further, since S is lower and $\Lambda = \Lambda^{\mathrm{HC}}_{s-1}$ we also have $S \subseteq \Lambda$. In fact, $\Lambda^{\mathrm{HC}}_{s-1}$ is the union of all lower sets of size at most s. It follows that

$$\|f - f_\Lambda\|_{L^\infty(\mathcal{U};\mathcal{V})} \leq \sum_{\boldsymbol{\nu}\notin \Lambda} \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}} \leq \sum_{\boldsymbol{\nu}\notin S} \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}}.$$

Recall that $\widetilde{m} \geq 2^d \bar{s}^2$ by assumption. Hence (37) and (40) implies that $s \geq \bar{s}$. We now use Theorem 10 and the fact that $f \in \mathcal{HA}(\gamma, \epsilon, d)$ to deduce that

$$\sum_{\boldsymbol{\nu}\notin S} \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}\|c_{\boldsymbol{\nu}}\|_{\mathcal{V}} \leq \exp(-\gamma s^{1/d}).$$

Returning to (49), this therefore gives

$$\|f - f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \lesssim \exp(-\gamma s^{1/d}) + \sqrt{N}\delta + E_2 + E_3 \leq \exp(-\gamma s^{1/d}) + E_2 + E_3,$$

provided

$$\delta \leq N^{-1/2}\exp(-\gamma s^{1/d}). \tag{50}$$

Hence, recalling (42) we now set

$$\delta = \frac{1}{\sqrt{N}} \min\left\{ \frac{9-4\sqrt{2}}{2\sqrt{5}(3+4\sqrt{s})}, \exp(-\gamma s^{1/d}) \right\}, \tag{51}$$

and recall the definitions (37) and (18) of s and E_1 respectively to deduce that

$$\|f - f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \lesssim E_1 + E_2 + E_3,$$

as required.

Step 6: Having now shown the error bound, the final step of the proof involves bounding the size, depth and number of trainable parameters for DNNs in the class $\mathcal{N}$ defined in Step 1. To do this, we notice from (51) that

$$\delta \gtrsim s^{-1/2}N^{-1/2}\exp(-\gamma s^{1/d}),$$

which, after a few steps, implies

$$\log(\delta^{-1}) \lesssim \log(s) + \log(N) + \gamma s^{1/d}.$$

Moreover, from the definition of the hyperbolic cross index set (26) we notice that $m(\Lambda) \leq s$ (recall that $m(\Lambda) = \max_{\nu\in\Lambda} \|\boldsymbol{\nu}\|_1$, as defined in Proposition 11). Then, substituting this into the bounds of the size and depth in Proposition 11, we deduce that

$$\begin{aligned}
&\mathrm{depth}(\mathcal{N}) \lesssim (1 + d\log(d)) \cdot (1 + \log(s)) \cdot (s + \log(s) + \log(N) + \gamma s^{1/d}),\\
&\quad \mathrm{size}(\mathcal{N}) \lesssim d^2 s^2 + ds(\log(s) + \log(N) + \gamma s^{1/d}) + d^2 \cdot N \cdot (1 + \log(s) + \log(N) + \gamma s^{1/d}) + NK.
\end{aligned}$$

We recall that $s \geq 1$. Then after some rearrangements we have

$$\begin{aligned} \mathrm{depth}(\mathcal{N}) &\lesssim (1 + d\log(d)) \cdot (1 + \log(s)) \cdot (s + \log(N) + \gamma s^{1/d}), \\ \mathrm{size}(\mathcal{N}) &\lesssim d^2 s^2 + (ds + d^2 N)(\log(s) + \log(N) + \gamma s^{1/d}) + NK, \end{aligned} \tag{52}$$

Now, it is easy to see from the definition in (37) that (52) becomes

$$\begin{aligned} \mathrm{depth}(\mathcal{N}) &\lesssim (1 + d\log(d)) \cdot (1 + \log(\widetilde{m})) \cdot \left((\widetilde{m}/2^d)^{1/2} + \log(N) + \gamma \widetilde{m}^{1/(2d)} \right), \\ \mathrm{size}(\mathcal{N}) &\lesssim d^2 (\widetilde{m}/2^d) + (d(\widetilde{m}/2^d)^{1/2} + d^2 N) \left(\log(\widetilde{m}) + \log(N) + \gamma \widetilde{m}^{1/(2d)} \right) + NK, \end{aligned}$$

Next we recall that the number of trainable parameters is $\mathrm{param}(\mathcal{N}) = N \cdot K$. To complete the proof, we use (33) and (40) to obtain $N \leq \Delta$, as required. ■

Acknowledgments

BA acknowledges the support of the PIMS CRG "High-dimensional Data Analysis", SFU's Big Data Initiative "Next Big Question" Fund and NSERC through grant R611675. SB acknowledges the support of NSERC, the Faculty of Arts and Science of Concordia University, and the CRM Applied Math Lab. ND acknowledges the support of the PIMS Postdoctoral Fellowship program.