Learning High-Dimensional Hilbert-Valued Functions With Deep Neural Networks From Limited Data

Ben Adcock[1], **Simone Brugiapaglia**[2], **Nick Dexter**[1], **Sebastian Moraga**[1]
[1]Department of Mathematics, Simon Fraser University, Burnaby, BC V5A 1S6, Canada
[2]Department of Mathematics & Statistics, Concordia University, Montréal, QC H3G 1M8, Canada
ben_adcock@sfu.ca, simone.brugiapaglia@concordia.ca, nicholas_dexter@sfu.ca, smoragas@sfu.ca

Abstract

Deep learning (DL) techniques have been successful in many applications, with the most impressive results achieved on problems where the dimension of the underlying data or problem domain is large. In this paper, we describe recent results on both scalar- and Hilbert-valued function approximation via Deep Neural Networks (DNN). This problem arises in many engineering problems, in particular those involving the solution of parametric Partial Differential Equations (PDEs). Such problems are challenging, since point-wise samples are expensive to acquire, and the function is usually high-dimensional. First, we consider a DNN architecture and training procedure for which the resulting DNN is guaranteed to perform as well as current best-in-class schemes for holomorphic, scalar- or Hilbert-valued functions based on polynomial approximations. This result demonstrates the efficacy of DL for this problem, and makes explicit the effect of all sources of error, including discretization error of the underlying Hilbert space and measurement error. Second, we provide several numerical results illustrating the performance of DNNs on both real-valued functions and solutions of parametric PDEs. These results suggest that better approximations can be achieved through careful tuning of the DNN architecture and training algorithm.

1 Introduction

Modern machine learning approaches based on training DNNs on large datasets have achieved impressive results on high-dimensional problems in several important scientific computing applications, including inverse problems in imaging (Adcock and Hansen 2021; Ongie et al. 2020), molecular dynamics simulations (Faber et al. 2017), partial differential equations (PDEs) (Berg and Nyström 2018), and parameterized PDEs for uncertainty quantification (UQ) in engineering (Cyr et al. 2020; Dal Santo, Deparis, and Pegolotti 2020; Geist et al. 2020; Khoo, Lu, and Ying 2020; Laakmann and Petersen 2020). Recent theoretical results on DNN expressivity suggest that such architectures are capable of approximating a wide class of functions by emulating existing approximation schemes, e.g., polynomials, wavelets, or free-knot splines. For smooth functions, exponential rates of convergence have been shown for fully-connected ReLU DNN architectures (Opschoor, Schwab, and Zech 2019).

While these results are impressive, many important questions about the application of DL techniques to problems in scientific computing remain. The work (Adcock and Dexter 2020) raised key issues regarding current methods of training which prevent DNNs from practically achieving high-accuracy approximations on smooth function approximation tasks. It was also demonstrated in (Antun et al. 2020; Gottschling et al. 2020) that DL models trained to solve inverse problems in imaging can fail to recover small structural changes in images, and small perturbations can lead to numerical artifacts. In other words, DNNs can lead to unstable methods unless they are carefully constructed. Hence, in order for these tools to be accepted and applied to critical tasks in scientific computing with strict error tolerances, e.g., UQ problems in mechanical engineering, it is necessary to develop theoretical foundations on stability and convergence of DNNs, as well as an understanding of how to train them to ensure these properties.

In this work we showcase results on learning high-dimensional scalar- and Hilbert-valued functions from limited datasets by DNNs. While most previous studies have considered scalar-valued functions, Hilbert-valued functions arise is various important applications, for example UQ problems involving PDEs. Such problems are typically expressed in terms of a differential operator $\mathcal{D}_{\boldsymbol{x}}$, with $\boldsymbol{x}$ comprising the spatial and temporal variables, and the goal is to find for all $\boldsymbol{y} \in \mathcal{U} \subseteq \mathbb{R}^d$ a solution $u \in \mathcal{V}$, with $\mathcal{V}$ a Hilbert or Banach space, satisfying

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

subject to suitable boundary conditions. Existing techniques for solving such problems achieve a discretization of the infinite-dimensional parameter to solution map $\boldsymbol{y} \mapsto u(\boldsymbol{y}) \in \mathcal{V}$ by combining spatial discretization, e.g., finite element or difference methods, with global polynomial approximation, achieved through Galerkin projection (Babuška, Tempone, and Zouraris 2004), hierarchical interpolation on sparse grids (Nobile, Tempone, and Webster 2008a), discrete least-squares (Chkifa et al. 2015), or compressive sensing-based approaches (Adcock, Brugiapaglia, and Webster 2017; Dexter, Tran, and Webster 2019; Doostan and Owhadi 2011; Hampton and Doostan 2015).

Challenges

The large underlying dimension of problems of the form (1) presents a major challenge in practical applications. The *curse of dimensionality* states that the amount of data and computational effort needed to construct an approximation may grow exponentially with the problem dimension. In many real-world problems arising in scientific computing and UQ the parametric dimension $d \gg 1$, as the parameters are used to model various material properties, forcing terms, and boundary information. In such settings, naïve application of the aforementioned techniques can lead to computationally inefficient schemes or suboptimal approximations with respect to the *sample complexity*, i.e., the number of samples required to construct the approximation.

Previous work

DNNs have recently been studied for these problems in (Kutyniok et al. 2020) where, under assumptions guaranteeing low-dimensionality of the solution manifold, fast, dimension-independent rates of convergence were established by connecting DNNs to reduced basis approximations and the theory of Kolmogorov N-widths. The work (Geist et al. 2020) numerically tested the accuracy of DNNs in approximating solutions to the parametric diffusion equation over a range of dimensions, finding errors scale with the inherent complexity of the problem as the dimension increases, but the observed scaling was not exponential. Other works such as (Cyr et al. 2020; Dal Santo, Deparis, and Pegolotti 2020; Khoo, Lu, and Ying 2020; Laakmann and Petersen 2020) have also demonstrated the potential of the DNN approach for parametric PDEs.

Further, it is hoped that DNNs may overcome some of the limitations of standard polynomial-based methods. These methods require strong smoothness assumptions on the underlying parameterized PDE problem in order to show fast rates of convergence. However, problems typically found in engineering or science applications rarely satisfy all of these assumptions (Smith 2013). On the other hand, DNNs have been shown to achieve optimal rates of approximation for piecewise smooth functions (Petersen and Voigtlaender 2018), suggesting far greater flexibility than the polynomial-based approach. Numerical studies have also demonstrated the efficiency of DNNs in approximating discontinuous (scalar-valued) functions (Adcock and Dexter 2020), whereas polynomial-based methods fail to converge.

This work

The purpose of this work is to describe recent results (Adcock and Dexter 2020; Adcock et al. 2020) on DNN approximation of scalar- and Hilbert-valued functions and their application to parametric PDEs. Although DNNs have shown some promise for discontinuous functions, in this work we focus on the smooth case, where a direct comparison with polynomial-based methods is valid. In particular, we focus on the following question: *to what extent (both theoretically and empirically) can DNN approaches outperform current best-in-class polynomial-based methods?*

2 Preliminaries

We first require some notation. For $d \in \mathbb{N}$ 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. The inequality $\boldsymbol{\mu} \leq \boldsymbol{\nu}$ is understood componentwise. We write $\mathbf{0}$ and $\mathbf{1}$ for the multi-indices consisting of all zeros and all ones respectively. 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 $A \gtrsim B$.

Setup

Throughout, we let $\mathcal{V}$ be a separable Hilbert space over the field $\mathbb{R}$, 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. $\boldsymbol{u} = (u_i)_{i=1}^N$ where $u_i \in \mathcal{V}$, $i = 1, \ldots, N$. 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, $v_{\boldsymbol{\nu}} \in \mathcal{V}$. We define the space $\ell^2(\Lambda; \mathcal{V})$ as the set of those sequences for which the corresponding norm

$$\|\boldsymbol{v}\|_{\mathcal{V},2} := \left(\sum_{\boldsymbol{\nu} \in \Lambda} \|v_{\boldsymbol{\nu}}\|_{\mathcal{V}}^2 \right)^{1/2} < \infty.$$

For the parametric domain, we consider the hypercube equipped with the uniform probability measure

$$\mathcal{U} = [-1,1]^d, \qquad \mathrm{d}\varrho(\boldsymbol{y}) = 2^{-d}\,\mathrm{d}\boldsymbol{y}. \tag{2}$$

For $1 \leq p \leq \infty$, we write $L_\varrho^p(\mathcal{U})$ for the corresponding Lebesgue spaces and $\|\cdot\|_{L_\varrho^p(\mathcal{U})}$ for their norms. Next, we define the Bochner space $L_\varrho^p(\mathcal{U}; \mathcal{V})$ as the space consisting of (equivalence classes of) strongly ϱ-measurable functions $f : \mathcal{U} \to \mathcal{V}$ for which

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

Generally speaking, one cannot deal directly with an infinite-dimensional output space $\mathcal{V}$. Hence, we introduce a *discretization* (e.g. a finite element discretization). This is a finite-dimensional subspace $\mathcal{V}_h \subset \mathcal{V}$ of dimension $K = \dim(\mathcal{V}_h)$. We let $\mathcal{P}_h : \mathcal{V} \to \mathcal{V}_h$ be the orthogonal projection onto $\mathcal{V}_h$ and $\{\varphi_k\}_{k=1}^K$ be a (not necessarily orthogonal) basis of $\mathcal{V}_h$. Moreover, for $f \in L_\varrho^2(\mathcal{U}; \mathcal{V})$ we let $\mathcal{P}_h f \in L_\varrho^2(\mathcal{U}; \mathcal{V}_h)$ be the function defined almost everywhere as

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

Holomorphy and polynomial approximation

The focus of this work is the approximation of smooth functions $f : \mathcal{U} \to \mathcal{V}$. Note that f may either be scalar-valued, e.g. $\mathcal{V} = \mathbb{R}$, or Hilbert-valued, in which case $\mathcal{V}$ is a separable, but potentially infinite-dimensional Hilbert space. By smooth, we mean that f has a holomorphic extension to a suitable complex region $\mathcal{O}$ of the form $\mathcal{U} \subseteq \mathcal{O} \subseteq \mathbb{C}^d$.

Specifically, we take these regions to be *Bernstein polyellipses*. Given parameters $\boldsymbol{\rho} > \mathbf{1}$, these are of the form

$$\mathcal{E}_{\boldsymbol{\rho}} = \mathcal{E}_{\rho_1} \times \cdots \times \mathcal{E}_{\rho_d},$$

where, for $\rho > 1$, $\mathcal{E}_\rho = \{\frac{1}{2}(z + z^{-1}) : z \in \mathbb{C}, 1 \leq |z| \leq \rho\} \subset \mathbb{C}$ are the classical Bernstein ellipses. Note that Bernstein polyellipses, much like classical Bernstein ellipses, are intimately connected with multivarate polynomial approximation (Chkifa, Cohen, and Schwab 2015; Cohen and DeVore 2015; Cohen, DeVore, and Schwab 2011).

To be precise, in this work we consider the classes of functions $\mathcal{HA} = \mathcal{HA}(\gamma, \epsilon, d)$, where $\gamma > 0$ and $\epsilon > 0$ are constants. This class consists of 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 $\boldsymbol{\rho} = (\rho_j)_{j=1}^d$ also satisfy

$$\frac{1}{d+1}\left(\frac{d! \prod_{j=1}^d \log(\rho_j)}{1+\epsilon}\right)^{1/d} \geq \gamma. \tag{3}$$

The motivation for this definition is the following. For any $f \in \mathcal{HA}(\gamma, \epsilon, d)$ its best s-term polynomial approximation f_s in L^2_ϱ-orthogonal polynomials (normalized, tensor Legendre polynomials) decays exponentially fast in s: namely,

$$\|f - f_s\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \leq \exp(-\gamma s^{1/d}), \quad \forall s \geq \bar{s}(d, \epsilon, \boldsymbol{\rho}), \tag{4}$$

where $\bar{s}(d, \epsilon, \boldsymbol{\rho})$ is a constant (Opschoor, Schwab, and Zech 2019). This is the rate we seek to obtain with a DNN approximation.

Problem statement

Let $f : \mathcal{U} \to \mathcal{V}$ be a holomorphic and potentially Hilbert-valued function. We assume m sample points $\{\boldsymbol{y}_i\}_{i=1}^m$ are drawn independently and identically according to the uniform measure ϱ. For each $\boldsymbol{y}_i$, we suppose an approximate value of $f(\boldsymbol{y}_i)$ is computed in the space $\mathcal{V}_h$. Hence, the measurements of f are

$$d_i = f(\boldsymbol{y}_i) + n_i \in \mathcal{V}_h, \quad i = 1, \ldots, m. \tag{5}$$

Note that this encompasses the fact that, for example in parametric PDEs, $f(\boldsymbol{y}_i)$ is computed via a black-box numerical routine (e.g. a numerical PDE solve) that must, of course, return a value in a finite-dimensional space. Thus the measurement error term n_i contains the error resulting from numerical solver.

Our goal is to approximate f from the measurements $\{d_i\}_{i=1}^m$ using a DNN. Write the projection $\mathcal{P}_h f$ in terms of the basis $\{\varphi_k\}_{k=1}^K$ as

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

We aim to approximate the vector-valued map $\mathbb{R}^d \to \mathbb{R}^K$, $\boldsymbol{y} \mapsto (c_k(\boldsymbol{y}))_{k=1}^K$, with a DNN $\hat{\Phi} : \mathbb{R}^d \to \mathbb{R}^K$. We consider feedforward DNN architectures, i.e.

$$\hat{\Phi}(\boldsymbol{y}) = \mathcal{A}_{L+1}(\rho(\mathcal{A}_L(\rho(\cdots \rho(\mathcal{A}_0(\boldsymbol{y}))\cdots)))), \quad L \geq 1$$

where ρ is the activation function and $\mathcal{A}_l$ are the affine maps. Given such a DNN Φ the resulting approximation to f is

$$f(\boldsymbol{y}) \approx f_\Phi(\boldsymbol{y}) = \sum_{k=1}^K (\Phi(\boldsymbol{y}))_k \varphi_k \in \mathcal{V}_h.$$

Note that when $\mathcal{V} = \mathbb{R}$, i.e. f is scalar-valued, such considerations are unnecessary, since $\mathcal{V}_h = \mathcal{V} = \mathbb{R}$ and we simply have $f(\boldsymbol{y}) \approx \Phi(\boldsymbol{y})$.

3 Learning holomorphic, Hilbert-valued functions via DNNs

We now present a theoretical result demonstrating that DNNs can be learned to efficiently approximate holomorphic functions. Note that (Opschoor, Schwab, and Zech 2019) has previously shown that there is a DNN that achieves the same rate of approximation (4) as the best s-term polynomial approximation. In the following result, we show that such a DNN can also be learned efficiently from the data (5) through a standard training strategy. Due to space limitations we omit some details. The full details and proof can be found in (Adcock et al. 2020).

Theorem 3.1 *Let $m \geq 2$,*

$$\widetilde{m} := cm/(\log^2(m)\min\{\log(m) + d, \log(m)\log(2d)\}),$$

such that $\widetilde{m} \geq 2^d$ and $\{\boldsymbol{y}_i\}_{i=1}^m$ be drawn randomly and independently from ϱ. Then, with high probability, there is

- *a class of neural networks $\mathcal{N}$ whose size, maximum depth and number of trainable parameters are at most polynomial in $\widetilde{m}$ (the rate of growth can be specified, along with the dependence on d),*
- *a regularization functional $\mathcal{J} : \mathcal{N} \to [0, \infty)$ based on a certain norm of the trainable parameters,*

such that for every $f \in \mathcal{HA}(\gamma, \epsilon, d)$ with noisy evaluations (5)*, any minimizer $\hat{\Phi}$ of the regularized training problem*

$$\underset{\Phi \in \mathcal{N}}{\text{minimize}} \ \sqrt{\tfrac{1}{m}\textstyle\sum_{i=1}^m \|f_\Phi(\boldsymbol{y}_i) - d_i\|_{\mathcal{V},2}^2} + \lambda\mathcal{J}(\Phi), \tag{6}$$

satisfies, for $\boldsymbol{e} = (n_i)_{i=1}^m/\sqrt{m}$ and for m large enough,

$$\|f - f_{\hat{\Phi}}\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \lesssim \underbrace{\exp(-\gamma\widetilde{m}^{1/(2d)}/\sqrt{2})}_{(a)} + \underbrace{\|\boldsymbol{e}\|_{\mathcal{V},2}}_{(b)} + \underbrace{\|f - \mathcal{P}_h(f)\|_{L^\infty(L^2_\varrho;\mathcal{V})}}_{(c)}.$$

Many recent works have shown that DNNs can efficiently approximate different classes of functions – see, e.g., (Adcock and Dexter 2020; Petersen and Voigtlaender 2018; Opschoor, Schwab, and Zech 2019; Grohs et al. 2019) and references therein). Such results are of the form of *existence theorems*, i.e. they assert the existence of a DNN of a give architecture with favourable approximation properties – for example, as noted above (Opschoor, Schwab, and Zech 2019) shows the existence of DNN with the same approximation rates as the best s-term polynomial approximation –

but no constructive means for learning it, nor an estimate on the number and type of samples needed to do so. We term Theorem 3.1 a *practical existence theorem*, since it asserts not only the existence of such a DNN, but also that it can be trained. Moreover, through the first term (a) in the error bound, which is exponentially small in $\widetilde{m}$, it shows that the DNN can be trained in a sample-efficient way. It shows that such a DNN achieves the same error as the best s-term polynomial approximation, with a sample complexity that scales quadratically in s. This is, in fact, exactly the same sample complexity as achieved by polynomial-based compressed sensing methods for holomorphic function approximation (Adcock, Brugiapaglia, and Webster 2017; Dexter, Tran, and Webster 2019). Hence, (a) asserts the DNN procedure achieves the sample approximation rate in terms of $\widetilde{m}$ as current best-in-class schemes.

This aside, Theorem 3.1 also makes explicit the effect of the three main errors in the training process. First, the *approximation error* (a), that depends on number of samples $\widetilde{m}$. Second, the *sampling error* (b), i.e. the error in the numerical PDE solve. And third, the *discretization error* (c). This error is due to working in the finite-dimensional space $\mathcal{V}_h$ rather than $\mathcal{V}$. However, the key point is that it is proportional to the error of the orthogonal projection $\mathcal{P}_h(f)$, i.e. the best approximation to f from $\mathcal{V}_h$.

4 Numerical experiments

This result demonstrates the potential efficacy of the DNN approach for Hilbert-valued function approximation. Namely, the DNN approach can perform as well as current best-in-class schemes based on compressed sensing for holomorphic, Hilbert-valued function approximation. However, we caution that the training procedure outlined in Theorem 3.1 is not expected to lead to DNNs that perform any better. Its proof heavily leverages the relation between DNNs and polynomials and loss function minimization and standard optimization problems for compressed sensing. On the other hand, the generality of the fully-connected DNN architectures allows them to be applied to problems where polynomial approximations are known to fail, e.g. functions with discontinuities or low parametric regularity. This motivates the need for numerical experimentation, which compares the practical performance of DNN training with standard architectures and loss functions to other schemes such as polynomial-based compressed sensing.

Scalar-valued function approximation

In this section, we consider scalar-valued function approximation. Our purpose is to compare the standard DNN training to best-in-class compressed sensing schemes. For further details of the study, and for additional experiments, see (Adcock and Dexter 2020). We focus on two factors. First, the sample complexity, i.e. the behaviour of the error as m varies, and second, the effect of noise in the measurements.

We consider ReLU DNN architectures with a fixed number of nodes N per layer and depth L. We vary the ratio $\beta := L/N$ to try to obtain the best approximation. We use `TensorFlow` and perform calculations in single

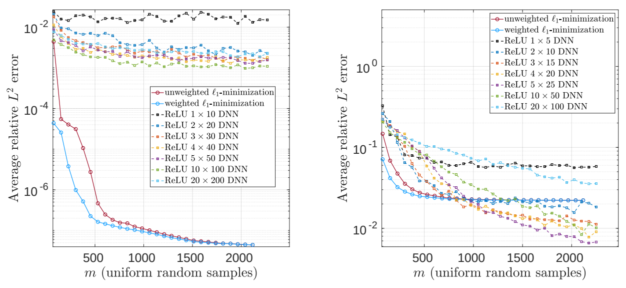

Figure 1: Error approximating **(left)** $f_1(\boldsymbol{y})$ and **(right)** $f_2(\boldsymbol{y})$ with $d = 8$.

precision. For training, we use the standard ℓ^2-loss function and the `Adam` optimizer (Kingma and Ba 2014). We train for a maximum of 50,000 epochs or until the tolerance $\varepsilon_{\text{tol}} = 5 \times 10^{-7}$ is reached. The weights and biases are initialized as normal random variables with mean 0 and variance 0.01. The compressed sensing schemes follow the setup of (Adcock, Brugiapaglia, and Webster 2017), and are based on Legendre polynomials in a hyperbolic cross index set, combined with either ℓ^1-minimization or weighted ℓ^1-minimization (the latter being known to offer improved performance). These experiments are performed in double precision in `Matlab` using the `SPGL1` solver (van den Berg and Friedlander 2019, 2009). The approximation error is taken as the relative $L^2(\mathcal{U})$ error, and is computed using a high-order isotropic Clenshaw–Curtis sparse grid quadrature rule consisting of roughly 1.9×10^6 points.

In Fig. 1, we consider two different smooth, scalar-valued functions:

$$f_1(\boldsymbol{y}) = \exp\left(-\frac{\sum_{k=1}^{d} \cos(y_k)}{8d}\right),$$

$$f_2(\boldsymbol{y}) = \left(\frac{\prod_{k=1}^{d/2} 1 + 4^k y_k^2}{\prod_{k=d/2+1}^{d} 100 + 5y_k}\right)^{1/d}.$$

This experiment demonstrates a gap between the practical existence theorem, Theorem 3.1, and practical performance of DNNs when trained with standard architectures and optimizers. The first function is extremely smooth and therefore has approximately sparse polynomial coefficients. In spite of Theorem 3.1, Fig. 1 demonstrates that compressed sensing significantly outperforms the DL approach. On the other hand, the function f_2 is less favourably approximated by polynomials, and the DL approach achieves competitive performance with compressed sensing. This points towards the efficacy of DL for broader classes of functions. Note that both experiments were also completed over 30 alternative ReLU DNN architectures of varying depth and width with similar results.

Next, in Fig. 2 we consider the effect of noise. These experiments suggest that the trained DNNs are numerically stable, with the noise contributing linearly to the overall error. Unlike the gap identified above in the approximation error, this behaviour is in good agreement with Theorem 3.1.

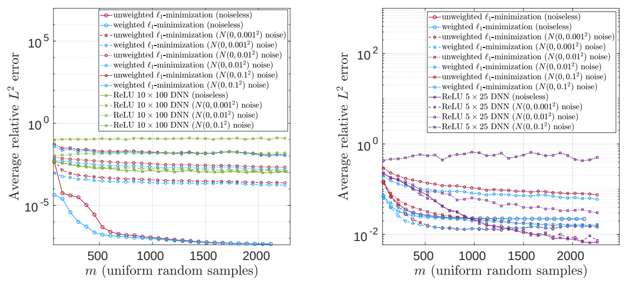

Figure 2: Errors of two best-performing ReLU DNNs in approximating **(left)** $f_1(\boldsymbol{y})$ and **(right)** $f_2(\boldsymbol{y})$ in $d = 8$ dimensions with and without $N(0, \sigma^2)$ noise for various σ.

Parametric PDEs with mixed boundary conditions

We conclude by applying DL to a parametric PDE problem. Specifically, we consider a non-trivial, two-dimensional Hilbert-valued function approximation problem arising as the solution of a parametric PDE with mixed boundary conditions. See, e.g. (Adcock et al. 2020; Geist et al. 2020), for parametric PDE problems with Dirichlet boundary conditions. Here, our objective is to examine the efficacy of the DNN approach on a non-standard parametric PDE.

Let $\Omega \subseteq \mathbb{R}^2$ be a given bounded domain with polyhedral boundary $\partial\Omega = \Gamma$, such that $\Gamma = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \emptyset$. We focus on the following parametric diffusion example with mixed boundary conditions

$$\begin{aligned} -\nabla \cdot (\mathbb{K}(\boldsymbol{x}, \boldsymbol{y}) \nabla \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{y})) &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) && \boldsymbol{x} \in \Omega, \boldsymbol{y} \in \mathcal{U}, \\ \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{y}) &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) && \boldsymbol{x} \in \Gamma_D, \boldsymbol{y} \in \mathcal{U}, \\ \nabla \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{n} &= \boldsymbol{0} && \boldsymbol{x} \in \Gamma_N, \boldsymbol{y} \in \mathcal{U}, \end{aligned}$$

where, given $\boldsymbol{y} \in \mathcal{U}$, $\boldsymbol{f}(\boldsymbol{y}) \in [L^2(\Omega)]^2$ and $\boldsymbol{h}(\boldsymbol{y}) \in [H^{1/2}(\Gamma_D)]^2$, whose components are in the space of traces of functions in $H^1(\Omega)$. We define $\mathbb{K} = \operatorname{diag}(\boldsymbol{a})$, where the vector $\boldsymbol{a} = [a_1, a_2]^\top$. Then, to guarantee well-posedness and parametric regularity of the solution $\boldsymbol{u}$, we require uniform boundedness assumptions on each component of $\mathbb{K}$, see (Cohen, DeVore, and Schwab 2011) for more details. That is, there exists $a_{j,\max} \geq a_{j,\min} > 0$ such that

$$a_{j\,\min} \leq a_j(\boldsymbol{x}, \boldsymbol{y}) \leq a_{j,\max}, \ \forall \boldsymbol{x} \in \overline{\Omega}, \forall \boldsymbol{y} \in \mathcal{U}, j \in \{1, 2\}.$$

In our example, we set $\Omega = (0, 1)^2$ and define the Dirichlet and Neumann boundaries as

$$\begin{aligned} \Gamma_D &= \{\boldsymbol{x} \in [0, 1]^2 : x_1 = 0, x_1 = 1\}, \\ \Gamma_N &= \{\boldsymbol{x} \in [0, 1]^2 : x_2 = 0, x_2 = 1\}. \end{aligned}$$

Next, we define the uniformly positive tensor $\mathbb{K}$ with components

$$\begin{aligned} a_1(\boldsymbol{x}, \boldsymbol{y}) &= 3 + x_1 y_1 + x_2 y_2, \\ a_2(\boldsymbol{x}, \boldsymbol{y}) &= \exp(1 + y_1 (\tfrac{\sqrt{\pi} L}{2})^{1/2} + \zeta \sin(\pi x_1) y_2). \end{aligned}$$

Here, the first is a simple affine coefficient, and the second is a modification of the example from (Nobile, Tempone, and Webster 2008b) of a diffusion coefficient with one-dimensional (layered) spatial dependence (see (Adcock et al. 2020) for more details). We next define the Hilbert spaces

$$\boldsymbol{H}(\operatorname{div}; \Omega) := \{\boldsymbol{\tau} \in [L^2(\Omega)]^2 : \operatorname{div}(\tau) \in L^2(\Omega)\},$$

and, similar to the analysis of (Gatica 2014, Section 4.2), we introduce $\boldsymbol{\sigma}(\boldsymbol{y}) = \nabla \boldsymbol{u}(\boldsymbol{y}) \in \boldsymbol{H}(\operatorname{div}; \Omega)$ as an additional unknown. Due to limitation of space, we now simply state the mixed formulation and refer to its well posednes to (Gatica 2014, Section 4). In this way, defining $\boldsymbol{H} = [L^2(\Omega)]^2$ and

$$\boldsymbol{Q} = \{\boldsymbol{\tau} \in [\boldsymbol{H}(\operatorname{div}; \Omega)]^2 : \tau \cdot \boldsymbol{n} = 0 \operatorname{in} \Gamma_N\},$$

one arrives to the mixed variational formulation: given $\boldsymbol{y} \in \mathcal{U}$, find $(\boldsymbol{u}(\boldsymbol{y}), \boldsymbol{\sigma}(\boldsymbol{y})) \in \boldsymbol{H} \times \boldsymbol{Q}$ such that

$$\begin{aligned} \langle \boldsymbol{\sigma}, \boldsymbol{\tau} \rangle_{L^2(\Omega)} + \langle \boldsymbol{u}, \nabla \cdot \boldsymbol{\tau} \rangle_{L^2(\Omega)} &= \langle \boldsymbol{\tau} \cdot \boldsymbol{n}, \boldsymbol{h} \rangle_{\Gamma_D} \\ \langle \nabla \mathbb{K} \cdot \boldsymbol{\sigma}, \boldsymbol{v} \rangle_{L^2(\Omega)} + \langle \mathbb{K} \boldsymbol{v}, \nabla \cdot \boldsymbol{\sigma} \rangle_{L^2(\Omega)} &= -\langle \boldsymbol{f}, \boldsymbol{v} \rangle_{L^2(\Omega)} \end{aligned} \tag{7}$$

for all $(\boldsymbol{v}, \boldsymbol{\tau}) \in \boldsymbol{H} \times \boldsymbol{Q}$. Note that, in this case, $\mathcal{V} = L^2(\Omega) \times \boldsymbol{H}(\operatorname{div}; \Omega)$.

To discretize this problem, we consider a uniform discretization, meaning an arbitrary finite-dimensional subspace, where the usual $[P_1]^2$-Lagrange finite elements are used for H and the Raviart-Thomas $[\boldsymbol{RT}_2]^2$ are used for $\boldsymbol{Q}$. Furthermore, given $\boldsymbol{y} \in \mathcal{U}$, we define the Finite Element (FE) and DNN approximations for $\boldsymbol{u}(\boldsymbol{y}) = [u_1, u_2]^\top(\boldsymbol{y})$ by

$$u_{h,j}(\boldsymbol{y}) = \sum_{k=1}^{K_1} c_{j,k}(\boldsymbol{y}) \varphi_k, \quad u_{\Phi,h,j}(\boldsymbol{y}) = \sum_{k=1}^{K_1} (\Phi(\boldsymbol{y}))_{k,j} \varphi_k,$$

for $j \in \{1, 2\}$ respectively, where $(\varphi_k)_{k=1}^{K_1}$ is a basis for P_1. Similarly, defining basis functions for the $\boldsymbol{RT}_2$-spaces as $(\widetilde{\varphi}_k)_{k=1}^{K_2}$, we can analogously define the FE and DNN approximations to $\boldsymbol{\sigma}$. We omit further details.

We consider different types of DNN architectures to approximate the coefficients of the finite element basis. In this study, we use the same finite element discretization in generating the sample training and testing data. We consider finite element discretizations on a mesh with $h = \sqrt{2}/32$, giving a total 22,914 degrees of freedom.

For comparison, we consider both the Leaky-ReLU and hyperbolic tangent activation functions. To train the DNN, we use the Adam optimizer along with an exponentially decay learning rate, minimizing the standard ℓ^2-mean squared error. The training data for the DNN is generated by solving (7) at a set of uniform random points $\{\boldsymbol{y}_i\}_{i=1}^m \subseteq \mathcal{U}$, with exact solution given by

$$\begin{aligned} & u_1(\boldsymbol{x}, \boldsymbol{y}) = \\ & \quad \sin(\pi(y_1 + y_2))(\cos(\pi x_2) \exp(x_1^2 - 1) + \cos(\pi x_1)), \\ & u_2(\boldsymbol{x}, \boldsymbol{y}) = \\ & \quad \cos(\pi(y_1 + y_2))(\cos(\pi x_2) \exp(x_1^2 - 1) + \sin(\pi x_1)). \end{aligned}$$

Fig. 3 shows the results of training with different loss functions. Generally speaking, networks using Leaky-ReLU achieve smaller training loss over the hyperbolic tangent. Moreover, the numerical results shows that deeper and wider networks are faster to train, meaning they can achieve a smaller training loss error after a fixed number of epochs. On the other hand, it is interesting to see that test errors

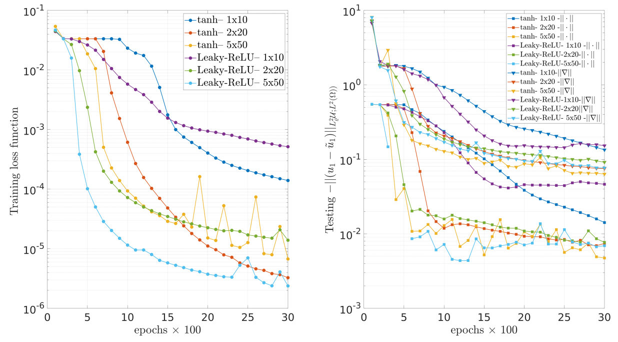

Figure 3: Shows the performance of different DNN architectures for approximating the mixed parametric PDE problem, using batch size of $m/2$, L hidden layers and N nodes per layer with $L/N = 0.1$. **(Left)** Comparison of training ℓ^2 loss function on $m = 200$ (uniform random samples). **(Right)** Shows the testing error Bochner norm of the function $u_{\Phi,h,1}$ and the error norm of its gradient $\sigma_{\Phi,h,1}$, square and triangle markers respectively.

are not so different for the different architectures, with the best two performing Leaky-ReLU and hyperbolic tangent networks giving similar results. Interestingly, even though separate DNNs are trained for each (due to the mixed formulation), the test errors for u_1 are substantially smaller (by several orders of magnitude) than those for its gradient ∇u_1.

In addition to this we also plot the best approximate solutions for $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ produced by the DNN. Figure 4 shows the approximations achieved by a DNN trained on $m = 200$ samples to error 6.3699×10^{-6} in the loss after 2900 epochs of training, and evaluated at input parameter value $\boldsymbol{y} = [0, -0.707106]$. It is noticeable that the approximation to ∇u_1 is slightly better than the approximation to ∇u_2. In Figure 5, we also compare the DNN solution for u_2 with the FE approximation.

5 Conclusion

Deep neural networks offer many advantages for efficiently learning scalar- and Hilbert-valued functions and, in particular, solutions to parametric PDE problems. Unlike virtually every polynomial-based approach, selection of a basis or dictionary is not necessary when applying DNNs. They also offer the capability to solve challenging UQ problems which possess discontinuities leading to low parametric regularity, problems for which methods such as sparse polynomial approximation via compressed sensing are poorly suited. In this work we presented a theoretical result which demonstrated the efficacy of the DNN approach for smooth function approximation. However, as observed in our first experiments and elaborated further in (Adcock and Dexter 2020), such theoretical rates may not be met in practice via standard architecture designs and training. This suggests a strong possibility to further modify DNN architecture and training procedures to achieve superior performance across a range of challenging high-dimensional problems, including those on which polynomials perform badly, e.g. nonsmooth functions. Finally, we also presented new results on DNN approximation of parametric PDEs with mixed boundary conditions. These results highlight the potential of the DNN approach, although further work is needed to tune the architectures to get optimal performance – including better approximation of the gradient – and to determine the method's efficacy, in particular, in comparison to current best-in-class schemes based on polynomial approximation.

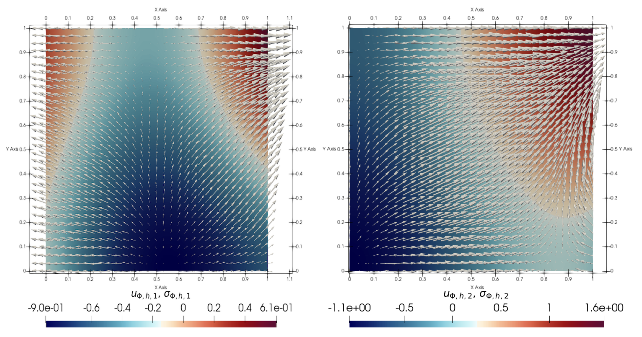

Figure 4: Visualization of the DNN approximation with hyperbolic tangent activation function, $L = 5$ hidden layers, and $N = 50$ nodes per hidden layer. The colours show the approximation to $u_{h,j}$, and the white arrows show the the approximation for $\sigma_{h,j}$ ($j = 1$ left and $j = 2$ right).

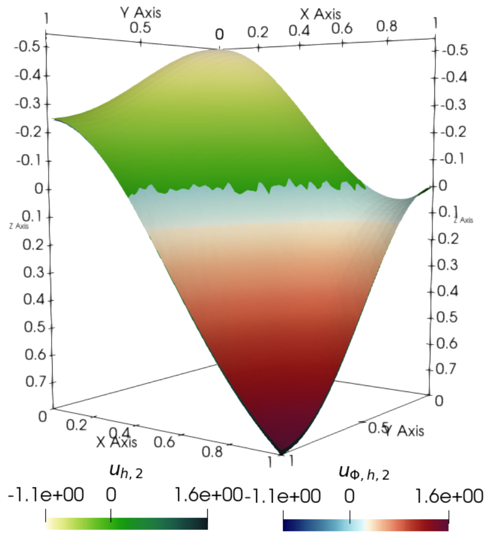

Figure 5: Comparison of the output of the DNN from Fig 4 to approximate $u_{2,h}(\boldsymbol{y})$. Here the testing error is $6.7321 \cdot 10^{-3}$, yellow/green colours the FE approximation, and blue/red colours the DNN approximation. We omit the visualization of u_1 since its indistinguishable from the FE approximation.

References

Adcock, B.; Brugiapaglia, S.; Dexter, N.; and Moraga, S. 2020. Deep Neural Networks Are Effective At Learning High-Dimensional Hilbert-Valued Functions From Limited Data. *Preprint* .

Adcock, B.; Brugiapaglia, S.; and Webster, C. G. 2017. Compressed sensing approaches for polynomial approximation of high-dimensional functions. In Boche, H.; Caire, G.; Calderbank, R.; März, M.; Kutyniok, G.; and Mathar, R., eds., *Compressed Sensing and its Applications: Second International MATHEON Conference 2015*, Applied and Numerical Harmonic Analysis, 93–124. Cham: Birkhäuser.

Adcock, B.; and Dexter, N. 2020. The gap between theory and practice in function approximation with deep neural networks. *arXiv:2001.07523* .

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

Antun, V.; Renna, F.; Poon, C.; Adcock, B.; and Hansen, A. C. 2020. On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proceedings of the National Academy of Sciences* .

Babuška, I.; Tempone, R.; and Zouraris, G. E. 2004. Galerkin Finite Element Approximations of Stochastic Elliptic Partial Differential Equations. *SIAM J. Numer. Anal.* 42(2): 800–825.

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

Chkifa, A.; Cohen, A.; Migliorati, G.; Nobile, F.; and Tempone, R. 2015. Discrete least squares polynomial approximation with random evaluations - application to parametric and stochastic elliptic pdes. *ESAIM Math. Model. Numer. Anal.* 49(3): 815–837.

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

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

Cohen, A.; and DeVore, R. A. 2015. Approximation of high-dimensional parametric PDEs. *Acta Numer.* .

Cyr, E. C.; Gulian, M. A.; Patel, R. G.; Perego, M.; and Trask, N. A. 2020. Robust Training and Initialization of Deep Neural Networks: An Adaptive Basis Viewpoint. In Lu, J.; and Ward, R., eds., *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*, 512–536. Princeton University, Princeton, NJ, USA: PMLR.

Dal Santo, N.; Deparis, S.; and Pegolotti, L. 2020. Data driven approximation of parametrized PDEs by Reduced Basis and Neural Networks. *J. Comput. Phys.* 416: 109550.

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

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

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

Gatica, G. 2014. A Simple Introduction to the Mixed Finite Element Method. Theory and Applications. *SpringerBriefs in Mathematics. Springer, Cham.* .

Geist, M.; Petersen, P.; Raslan, M.; Schneider, R.; and Kutyniok, G. 2020. Numerical Solution of the Parametric Diffusion Equation by Deep Neural Networks. *arXiv:2004.12131* .

Gottschling, N. M.; Antun, V.; Adcock, B.; and Hansen, A. C. 2020. The troublesome kernel: why deep learning for inverse problems is typically unstable. *arXiv:2001.01258* .

Grohs, P.; Perekrestenko, D.; Elbrächter, D.; and Bölcskei, H. 2019. Deep Neural Network Approximation Theory. *arXiv:1901.02220* .

Hampton, J.; and Doostan, A. 2015. Compressive Sampling of Polynomial Chaos Expansions: convergence Analysis and Sampling Strategies. *J. Comput. Phys.* 280: 363–386.

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

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

Kutyniok, G.; Petersen, P.; Raslan, M.; and Schneider, R. 2020. A Theoretical Analysis of Deep Neural Networks and Parametric PDEs. *arXiv:1904.00377* .

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

Nobile, F.; Tempone, R.; and Webster, C. G. 2008a. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.* 46(5): 2309–2345.

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

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

Opschoor, J. A. A.; Schwab, C.; and Zech, J. 2019. Exponential ReLU DNN expression of holomorphic maps in high dimension. Technical Report 35, ETH: Zürich. doi: https://doi.org/10.1142/S0219530518500203.

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

Smith, R. 2013. *Uncertainty Quantification: Theory, Implementation, and Applications*. Computational Science and Engineering. SIAM.

van den Berg, E.; and Friedlander, M. P. 2009. Probing the Pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.* 31(2): 890–912.

van den Berg, E.; and Friedlander, M. P. 2019. SPGL1: A solver for large-scale sparse reconstruction. https://friedlander.io/spgl1.