SMAI-JCM
SMAI Journal of Computational Mathematics

Model Reduction And Neural Networks For Parametric PDEs

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki & Andrew M. Stuart

Volume 7 (2021), p. 121-157.

<http://smai-jcm.centre-mersenne.org/item?id=SMAI-JCM_2021__7__121_0>

Publication membre du
Centre Mersenne pour l'édition scientifique ouverte
http://www.centre-mersenne.org/

Soumission sur https://smai-jcm.centre-mersenne.org/ojs/submission

SMAI Journal of Computational Mathematics
Vol. 7, 121-157 (2021)

Model Reduction And Neural Networks For Parametric PDEs

Kaushik Bhattacharya [1]
Bamdad Hosseini [2]
Nikola B. Kovachki [3]
Andrew M. Stuart [4]

[1] Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, USA
E-mail address: bhatta@caltech.edu
[2] Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA
E-mail address: bamdadh@caltech.edu
[3] Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA
E-mail address: nkovachki@caltech.edu
[4] Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA
E-mail address: astuart@caltech.edu.

Abstract. We develop a general framework for data-driven approximation of input-output maps between infinite-dimensional spaces. The proposed approach is motivated by the recent successes of neural networks and deep learning, in combination with ideas from model reduction. This combination results in a neural network approximation which, in principle, is defined on infinite-dimensional spaces and, in practice, is robust to the dimension of finite-dimensional approximations of these spaces required for computation. For a class of input-output maps, and suitably chosen probability measures on the inputs, we prove convergence of the proposed approximation methodology. We also include numerical experiments which demonstrate the effectiveness of the method, showing convergence and robustness of the approximation scheme with respect to the size of the discretization, and compare it with existing algorithms from the literature; our examples include the mapping from coefficient to solution in a divergence form elliptic partial differential equation (PDE) problem, and the solution operator for viscous Burgers' equation.

2020 Mathematics Subject Classification. 65N75, 62M45, 68T05, 60H30, 60H15.

Keywords. approximation theory, deep learning, model reduction, neural networks, partial differential equations.

1. Introduction

At the core of many computational tasks arising in science and engineering is the problem of repeatedly evaluating the output of an expensive forward model for many statistically similar inputs. Such settings include the numerical solution of parametric partial differential equations (PDEs), time-stepping for evolutionary PDEs and, more generally, the evaluation of input-output maps defined by black-box computer models. The key idea in this paper is the development of a new data-driven emulator which is defined to act between the infinite-dimensional input and output spaces of maps such as those defined by PDEs. By defining approximation architectures on infinite-dimensional spaces, we provide the basis for a methodology which is robust to the resolution of the finite-dimensionalizations used to create implementable algorithms.

This work is motivated by the recent empirical success of neural networks in machine learning applications such as image classification, aiming to explore whether this success has any implications for algorithm development in different applications arising in science and engineering. We further wish to compare the resulting new methods with traditional algorithms from the field of numerical analysis

The work is supported by MEDE-ARL funding (W911NF-12-0022). AMS is also partially supported by NSF (DMS 1818977) and AFOSR (FA9550-17-1-0185). BH is partially supported by a Von Kármán instructorship at the California Institute of Technology.

for the approximation of infinite-dimensional maps, such as the maps defined by parametric PDEs or the solution operator for time-dependent PDEs. We propose a method for approximation of such solution maps purely in a data-driven fashion by lifting the concept of neural networks to produce maps acting between infinite-dimensional spaces. Our method exploits approximate finite-dimensional structure in maps between Banach spaces of functions through three separate steps: (i) reducing the dimension of the input; (ii) reducing the dimension of the output, and (iii) finding a map between the two resulting finite-dimensional latent spaces. Our approach takes advantage of the approximation power of neural networks while allowing for the use of well-understood, classical dimension reduction (and reconstruction) techniques. Our goal is to reduce the complexity of the input-to-output map by replacing it with a data-driven emulator. In achieving this goal we design an emulator which enjoys mesh-independent approximation properties, a fact which we establish through a combination of theory and numerical experiments; to the best of our knowledge, these are the first such results in the area of neural networks for PDE problems.

To be concrete, and to guide the literature review which follows, consider the following prototypical parametric PDE

$$(\mathcal{P}_x y)(s) = 0, \qquad \forall\, s \in D,$$

where $D \subset \mathbb{R}^d$ is a bounded open set, $\mathcal{P}_x$ is a differential operator depending on a parameter $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ is the solution to the PDE (given appropriate boundary conditions). The Banach spaces $\mathcal{X}$ and $\mathcal{Y}$ are assumed to be spaces of real-valued functions on D. Here, and in the rest of this paper, we consistently use s to denote the independent variable in spatially dependent PDEs, and reserve x and y for the input and output of the PDE model of interest. We adopt this idiosyncratic notation (from the PDE perspective) to keep our exposition in line with standard machine learning notation for input and output variables.

Example 1.1. Consider second order elliptic PDEs of the form

$$\begin{aligned} -\nabla \cdot (a(s)\nabla u(s)) &= f(s), \quad s \in D \\ u(s) &= 0, \qquad s \in \partial D \end{aligned} \tag{1.1}$$

which are prototypical of many scientific applications. As a concrete example of a mapping defined by this equation, we restrict ourselves to the setting where the forcing term f is fixed, and consider the diffusion coefficient a as the input parameter x and the PDE solution u as output y. In this setting, we have $\mathcal{X} = L^\infty(D;\mathbb{R}_+)$, $\mathcal{Y} = H_0^1(D;\mathbb{R})$, and $\mathcal{P}_x = -\nabla_s \cdot (a\nabla_s \cdot) - f$, equipped with homogeneous Dirichlet boundary conditions. This is the Darcy flow problem which we consider numerically in Section 4.3.

1.1. **Literature Review**

The recent success of neural networks on a variety of high-dimensional machine learning problems [49] has led to a rapidly growing body of research pertaining to applications in scientific problems [1, 7, 13, 25, 30, 39, 68, 76, 83]. In particular, there is a substantial number of articles which investigate the use of neural networks as surrogate models, and more specifically for obtaining the solution of (possibly parametric) PDEs.

We summarize the two most prevalent existing neural network based strategies in the approximation of PDEs in general, and parametric PDEs specifically. The first approach can be thought of as image-to-image regression. The goal is to approximate the parametric solution operator mapping elements of $\mathcal{X}$ to $\mathcal{Y}$. This is achieved by discretizing both spaces to obtain finite-dimensional input and output spaces of dimension K. We assume to have access to data in the form of observations of input x and output y discretized on K-points within the domain D. The methodology then proceeds by defining a neural network $F : \mathbb{R}^K \to \mathbb{R}^K$ and regresses the input-to-output map by minimizing a misfit functional

defined using the point values of x and y on the discretization grid. The articles [1, 7, 29, 39, 83] apply this methodology for various forward and inverse problems in physics and engineering, utilizing a variety of neural network architectures in the regression step; the related paper [42] applies a similar approach, but the output space is $\mathbb{R}$. This innovative set of papers demonstrate some success. However, from the perspective of the goals of our work, their approaches are not robust to mesh-refinement: the neural network is defined as a mapping between two Euclidean spaces of values on mesh points. The rates of approximation depend on the underlying discretization and an overhaul of the architecture would be required to produce results consistent across different discretizations. The papers [54, 55] make a conceptual step in the direction of interest to us in this paper, as they introduce an architecture based on a neural network approximation theorem for operators from [12]; but as implemented the method still results in parameters which depend on the mesh used. Applications of this methodology may be found in [11, 52, 58].

The second approach does not directly seek to find the parametric map from $\mathcal{X}$ to $\mathcal{Y}$ but rather is thought of, for fixed $x \in \mathcal{X}$, as being a parametrization of the solution $y \in \mathcal{Y}$ by means of a deep neural network [24, 25, 40, 47, 68, 75]. This methodology parallels collocation methods for the numerical solution of PDEs by searching over approximation spaces defined by neural networks. The solution of the PDE is written as a neural network approximation in which the spatial (or, in the time-dependent case, spatio-temporal) variables in D are inputs and the solution is the output. This parametric function is then substituted into the PDE and the residual is made small by optimization. The resulting neural network may be thought of as a novel structure which composes the action of the operator $\mathcal{P}_x$, for fixed x, with a neural network taking inputs in D [68]. While this method leads to an approximate solution map defined on the input domain D (and not on a $K-$point discretization of the domain), the parametric dependence of the approximate solution map is fixed. Indeed for a new input parameter x, one needs to re-train the neural network by solving the associated optimization problem in order to produce a new map $y : D \to \mathbb{R}$; this may be prohibitively expensive when parametric dependence of the solution is the target of analysis. Furthermore the approach cannot be made fully data-driven as it needs knowledge of the underlying PDE, and furthermore the operations required to apply the differential operator may interact poorly with the neural network approximator during the back-propagation (adjoint calculation) phase of the optimization.

The work [70] examines the forward propagation of neural networks as the flow of a time-dependent PDE, combining the continuous time formulation of ResNet [34, 80] with the idea of neural networks acting on spaces of functions: by considering the initial condition as a function, this flow map may be thought of as a neural network acting between infinite-dimensional spaces. The idea of learning PDEs from data using neural networks, again generating a flow map between infinite dimensional spaces, was studied in the 1990s in the papers [32, 44] with the former using a PCA methodology, and the latter using the method of lines. More recently the works [37, 79] also employ a PCA methodology for the output space but only consider very low dimensional input spaces. Furthermore the works [28, 28, 31, 50] proposed a model reduction approach for dynamical systems by use of dimension reducing neural networks (autoencoders). However only a fixed discretization of space is considered, yielding a method which does not produce a map between two infinite-dimensional spaces.

The development of numerical methods for parametric problems is not, of course, restricted to the use of neural networks. Earlier works in the engineering literature started in the 1970s focused on computational methods which represent PDE solutions in terms of known basis functions that contain information about the solution structure [2, 61]. This work led to the development of the reduced basis method (RBM) which is widely adopted in engineering; see [3, 36, 67] and the references therein. The methodology was also used for stochastic problems, in which the input space $\mathcal{X}$ is endowed with a probabilistic structure, in [10]. The study of RBMs led to broader interest in the approximation theory community focusing on rates of convergence for the RBM approximation of maps between

Banach spaces, and in particular maps defined through parametric dependence of PDEs; see [23] for an overview of this work.

Ideas from model reduction have been combined with data-driven learning in the sequence of papers [6, 59, 64, 65, 66]. The setting is the learning of data-driven approximations to time-dependent PDEs. Model reduction is used to find a low-dimensional approximation space and then a system of ordinary differential equations (ODEs) is learned in this low-dimensional latent space. These ODEs are assumed to have vector fields from a known class with unknown linear coefficients; learning is thus reduced to a least squares problem. The known vector fields mimic properties of the original PDE (for example are restricted to linear and quadratic terms for the equations of geophysical fluid dynamics); additionally transformations may be used to render the original PDE in a desirable form (such as having only quadratic nonlinearities.)

The development of theoretical analyses to understand the use of neural networks to approximate PDEs is currently in its infancy, but interesting results are starting to emerge [35, 45, 46, 72]. A recurrent theme in the analysis of neural networks, and in these papers in particular, is that the work typically asserts the *existence* of a choice of neural network parameters which achieve a certain approximation property; because of the non-convex optimization techniques used to determine the network parameters, the issue of *finding* these parameters in practice is rarely addressed. Recent works take a different perspective on data-driven approximation of PDEs, motivated by small-data scenarios; see the paper [15] which relates, in part, to earlier work focused on the small-data setting [8, 56]. These approaches are more akin to data assimilation [48, 69] where the data is incorporated into a model.

1.2. **Our Contribution**

The primary contributions of this paper are as follows:

(1) we propose a novel data-driven methodology capable of learning mappings between Hilbert spaces;

(2) the proposed method combines model reduction with neural networks to obtain algorithms with controllable approximation errors as maps between Hilbert spaces;

(3) as a result of this approximation property of maps between Hilbert spaces, the learned maps exhibit desirable mesh-independence properties;

(4) we prove that our architecture is sufficiently rich to contain approximations of arbitrary accuracy, as a mapping between function spaces;

(5) we present numerical experiments that demonstrate the efficacy of the proposed methodology, demonstrate desirable mesh-indepence properties, elucidate its properties beyond the confines of the theory, and compare with other methods for parametric PDEs.

Section 2 outlines the approximation methodology, which is based on use of *principal component analysis (PCA)* in a Hilbert space to finite-dimensionalize the input and output spaces, and a neural network between the resulting finite-dimensional spaces. Section 3 contains statement and proof of our main approximation result, which invokes a global Lipschitz assumption on the map to be approximated. In Section 4 we present our numerical experiments, some of which relax the global Lipschitz assumption, and others which involve comparisons with other approaches from the literature. Section 5 contains concluding remarks, including directions for further study. We also include auxiliary results in the appendix that complement and extend the main theoretical developments of the article. Appendix A extends the analysis of Section 3 from globally Lipschitz maps to locally Lipschitz maps with controlled growth rates. Appendix B contains supporting lemmas that are used throughout the

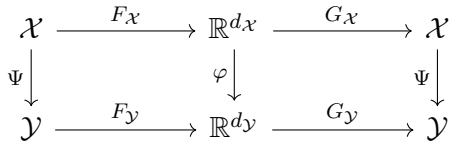


Figure 2.1. A diagram summarizing various maps of interest in our proposed approach for the approximation of input-output maps between infinite-dimensional spaces.

paper while Appendix C proves an analyticity result pertaining to the solution map of the Poisson equation that is used in one of the numerical experiments in Section 4.

2. **Proposed Method**

Our method combines PCA-based dimension reduction on the input and output spaces $\mathcal{X}, \mathcal{Y}$ with a neural network that maps the dimension-reduced spaces. After a pre-amble in Subsection 2.1, giving an overview of our approach, we continue in Subsection 2.2 with a description of PCA in the Hilbert space setting, including intuition about its approximation quality. Subsection 2.3 gives the background on neural networks needed for this paper, and Subsection 2.4 compares our methodology to existing methods.

2.1. **Overview**

Let $\mathcal{X}$, $\mathcal{Y}$ be separable Hilbert spaces and $\Psi : \mathcal{X} \to \mathcal{Y}$ be some, possibly nonlinear, map. Our goal is to approximate Ψ from a finite collection of evaluations $\{x_j, y_j\}_{j=1}^N$ where $y_j = \Psi(x_j)$. We assume that the x_j are i.i.d. with respect to (w.r.t.) a probability measure μ supported on $\mathcal{X}$. Note that with this notation the output samples y_j are i.i.d. w.r.t. the push-forward measure $\Psi_\sharp\mu$. The approximation of Ψ from the data $\{x_j, y_j\}_{j=1}^N$ that we now develop should be understood as being designed to be accurate with respect to norms defined by integration with respect to the measures μ and $\Psi_\sharp\mu$ on the spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively.

Instead of attempting to directly approximate Ψ, we first try to exploit possible finite-dimensional structure within the measures μ and $\Psi_\sharp\mu$. We accomplish this by approximating the identity mappings $I_{\mathcal{X}} : \mathcal{X} \to \mathcal{X}$ and $I_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Y}$ by a composition of two maps, known as the *encoder* and the *decoder* in the machine learning literature [33, 38], which have finite-dimensional range and domain, respectively. We will then interpolate between the finite-dimensional outputs of the encoders, usually referred to as the *latent codes*. Our approach is summarized in Figure 2.1.

Here, $F_{\mathcal{X}}$ and $F_{\mathcal{Y}}$ are the encoders for the spaces $\mathcal{X}, \mathcal{Y}$ respectively, whilst $G_{\mathcal{X}}$ and $G_{\mathcal{Y}}$ are the decoders, and φ is the map interpolating the latent codes. The intuition behind Figure 2.1, and, to some extent, the main focus of our analysis, concerns the quality of the approximations

$$G_{\mathcal{X}} \circ F_{\mathcal{X}} \approx I_{\mathcal{X}}, \tag{2.1a}$$

$$G_{\mathcal{Y}} \circ F_{\mathcal{Y}} \approx I_{\mathcal{Y}}, \tag{2.1b}$$

$$G_{\mathcal{Y}} \circ \varphi \circ F_{\mathcal{X}} \approx \Psi. \tag{2.1c}$$

In order to achieve (2.1c) it is natural to choose φ as

$$\varphi := F_{\mathcal{Y}} \circ \Psi \circ G_{\mathcal{X}}; \tag{2.2}$$

then the approximation (2.1c) is limited only by the approximations (2.1a), (2.1b) of the identity maps on $I_{\mathcal{X}}$ and $I_{\mathcal{Y}}$. We further label the approximation in (2.1c) by

$$\Psi_{PCA} := G_{\mathcal{Y}} \circ \varphi \circ F_{\mathcal{X}}, \tag{2.3}$$

since we later choose PCA as our dimension reduction method. We note that Ψ_{PCA} is not used in practical computations since φ is generally unknown. To make it practical we replace φ with a data-driven approximation $\chi \approx \varphi$ obtaining,

$$\Psi_{NN} := G_{\mathcal{Y}} \circ \chi \circ F_{\mathcal{X}}. \tag{2.4}$$

Later we choose χ to be a neural network, hence the choice of the subscript *NN*. The combination of PCA for the encoding/decoding along with the neural network approximation χ for φ, forms the basis of our computational methodology.

The compositions $G_{\mathcal{X}} \circ F_{\mathcal{X}}$ and $G_{\mathcal{Y}} \circ F_{\mathcal{Y}}$ are commonly referred to as *autoencoders*. There is a large literature on dimension-reduction methods [5, 19, 38, 63, 71] both classical and rooted in neural networks. In this work, we will focus on PCA which is perhaps one of the simplest such methods known [63]. We make this choice due to its simplicity of implementation, excellent numerical performance on the problems we study in Section 4, and its amenability to analysis. The dimension reduction in the input and output spaces is essential, as it allows for function space algorithms that make use of powerful finite-dimensional approximation methods, such as the neural networks we use here.

Many classical dimension reduction methods may be seen as encoders. But not all are as easily inverted as PCA – often there is no unambiguous, or no efficient, way to obtain the decoder. Whilst neural network based methods such as deep autoencoders [38] have shown empirical success in finite dimensional applications they currently lack theory and practical implementation in the setting of function spaces, and are therefore not currently suitable in the context of the goals of this paper.

Nonetheless methods other than PCA are likely to be useful within the general goals of high or infinite-dimensional function approximation. Indeed, with PCA, we approximate the solution manifold (image space) of the operator Ψ by the *linear* space defined in equation (2.7). We emphasize however that, usually, Ψ is a nonlinear operator and our approximation succeeds by capturing the induced nonlinear input-output relationship within the latent codes by using a neural network. We will show in Section 3.2 that the approximation error of the linear space to the solution manifold goes to zero as the dimension increases, however, this decay may be very slow [16, 22]. Therefore, it may be beneficial to construct nonlinear dimension reducing maps such as deep autoencoders on function spaces. We leave this as an interesting direction for future work.

Regarding the approximation of φ by neural networks, we acknowledge that there is considerable scope for the construction of the neural network, within different families and types of networks, and potentially by using other approximators. For our theory and numerics however we will focus on relatively constrained families of such networks, described in the following Subsection 2.3.

2.2. **PCA On Function Space**

Since we will perform PCA on both $\mathcal{X}$ and $\mathcal{Y}$, and since PCA requires a Hilbert space setting, the development here is in a generic real, separable Hilbert space $\mathcal{H}$ with inner-product and norm denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ respectively. We let ν denote a probability measure supported on $\mathcal{H}$, and make the assumption of a finite fourth moment: $\mathbb{E}_{u\sim\nu}\|u\|^4 < \infty$. We denote by $\{u_j\}_{j=1}^N$ a finite collection of N i.i.d. draws from ν that will be used as the training data on which PCA is based. Later we apply the PCA methodology in two distinct settings where the space $\mathcal{H}$ is taken to be the input space $\mathcal{X}$ and the data $\{u_j\}$ are the input samples $\{x_j\}$ drawn from the input measure μ, or $\mathcal{H}$ is taken to be the output space $\mathcal{Y}$ and the data $\{u_j\}$ are the corresponding outputs $\{y_j = \Psi(x_j)\}$ drawn from the push-forward measure $\Psi_\sharp\mu$. The following exposition, and the subsequent analysis in Section 3.1, largely

follows the works [9, 73, 74]. We will consider the standard version of non-centered PCA, although more sophisticated versions such as kernel PCA have been widely used and analyzed [71] and could be of potential interest within the overall goals of this work. We choose to work in the non-kernelized setting as there is an unequivocal way of producing the decoder.

For any subspace $V \subseteq \mathcal{H}$, denote by $\Pi_V : \mathcal{H} \to V$ the orthogonal projection operator and define the *empirical projection error*,

$$R_N(V) \coloneqq \frac{1}{N} \sum_{j=1}^{N} \|u_j - \Pi_V u_j\|^2. \tag{2.5}$$

PCA consists of projecting the data onto a finite-dimensional subspace of $\mathcal{H}$ for which this error is minimal. To that end, consider the *empirical, non-centered covariance* operator

$$C_N \coloneqq \frac{1}{N} \sum_{j=1}^{N} u_j \otimes u_j \tag{2.6}$$

where $\otimes$ denotes the outer product. It may be shown that C_N is a non-negative, self-adjoint, trace-class operator on $\mathcal{H}$, of rank at most N [82]. Let $\phi_{1,N}, \dots \phi_{N,N}$ denote the eigenvectors of C_N and $\lambda_{1,N} \geq \lambda_{2,N} \geq \cdots \geq \lambda_{N,N} \geq 0$ its corresponding eigenvalues in decreasing order. Then for any $d \geq 1$ we define the *PCA subspaces*

$$V_{d,N} = \operatorname{span}\{\phi_{1,N}, \phi_{2,N}, \dots, \phi_{d,N}\} \subset \mathcal{H}. \tag{2.7}$$

It is well known [60, Thm. 12.2.1] that $V_{d,N}$ solves the minimization problem

$$\min_{V \in \mathcal{V}_d} R_N(V),$$

where $\mathcal{V}_d$ denotes the set of all d-dimensional subspaces of $\mathcal{H}$. Furthermore

$$R_N(V_{d,N}) = \sum_{j=d+1}^{N} \lambda_{j,N}, \tag{2.8}$$

hence the approximation is controlled by the rate of decay of the spectrum of C_N.

With this in mind, we define the *PCA encoder* $F_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^d$ as the mapping from $\mathcal{H}$ to the coefficients of the orthogonal projection onto $V_{d,N}$ namely,

$$F_{\mathcal{H}}(u) = (\langle u, \phi_{1,N} \rangle, \dots, \langle u, \phi_{d,N} \rangle)^T \in \mathbb{R}^d. \tag{2.9}$$

Correspondingly, the PCA decoder $G_{\mathcal{H}} : \mathbb{R}^d \to \mathcal{H}$ constructs an element of $\mathcal{H}$ by taking as its input the coefficients constructed by $F_{\mathcal{H}}$ and forming an expansion in the empirical basis by zero-padding the PCA basis coefficients, that is

$$G_{\mathcal{H}}(s) = \sum_{j=1}^{d} s_j \phi_{j,N} \qquad \forall\, s \in \mathbb{R}^d. \tag{2.10}$$

In particular,

$$(G_{\mathcal{H}} \circ F_{\mathcal{H}})(u) = \sum_{j=1}^{d} \langle u, \phi_{j,N} \rangle \phi_{j,N}, \quad \text{equivalently} \quad G_{\mathcal{H}} \circ F_{\mathcal{H}} = \sum_{j=1}^{d} \phi_{j,N} \otimes \phi_{j,N}.$$

Hence $G_{\mathcal{H}} \circ F_{\mathcal{H}} = \Pi_{V_{d,N}}$, a d-dimensional approximation to the identity $I_{\mathcal{H}}$.

We will now give a qualitative explanation of this approximation to be made quantitative in Subsection 3.1. It is natural to consider minimizing the infinite data analog of (2.5), namely the *projection error*

$$R(V) \coloneqq \mathbb{E}_{u \sim \nu} \|u - \Pi_V u\|^2, \tag{2.11}$$

over $\mathcal{V}_d$ for $d \geq 1$. Assuming ν has a finite second moment, there exists a unique, self-adjoint, non-negative, trace-class operator $C : \mathcal{H} \to \mathcal{H}$ termed the *non-centered covariance* such that $\langle v, Cz\rangle = \mathbb{E}_{u\sim\nu}[\langle v, u\rangle\langle z, u\rangle]$, $\forall\, v, z \in \mathcal{H}$ (see [4]). From this, one readily finds the form of C by noting that

$$\langle v, \mathbb{E}_{u\sim\nu}[u \otimes u]z\rangle = \mathbb{E}_{u\sim\nu}[\langle v, (u \otimes u)z\rangle] = \mathbb{E}_{u\sim\nu}[\langle v, u\rangle\langle z, u\rangle], \tag{2.12}$$

implying that $C = \mathbb{E}_{u\sim\nu}[u \otimes u]$. Moreover, it follows that

$$\operatorname{tr} C = \mathbb{E}_{u\sim\nu}[\operatorname{tr} u \otimes u] = \mathbb{E}_{u\sim\nu}\|u\|^2 < \infty.$$

Let $\phi_1, \phi_2, \ldots$ denote the eigenvectors of C and $\lambda_1 \geq \lambda_2 \geq \ldots$ the corresponding eigenvalues. In the infinite data setting ($N = \infty$) it is natural to think of C and its first d eigenpairs as known. We then define the *optimal projection space*

$$V_d = \operatorname{span}\{\phi_1, \phi_2, \ldots, \phi_d\}. \tag{2.13}$$

It may be verified that V_d solves the minimization problem $\min_{V\in\mathcal{V}_d} R(V)$ and that $R(V_d) = \sum_{j=d+1}^{\infty} \lambda_j$.

With this infinite data perspective in mind observe that PCA makes the approximation $V_{d,N} \approx V_d$ from a finite dataset. The approximation quality of $V_{d,N}$ w.r.t. V_d is related to the approximation quality of ϕ_j by $\phi_{j,N}$ for $j = 1, \ldots, N$ and therefore to the approximation quality of C by C_N. Another perspective is via the Karhunen–Loeve Theorem (KL) [53]. For simplicity, assume that ν is mean zero, then $u \sim \nu$ admits an expansion of the form $u = \sum_{j=1}^{\infty} \sqrt{\lambda_j}\xi_j\phi_j$ where $\{\xi_j\}_{j=1}^{\infty}$ is a sequence of scalar-valued, mean zero, pairwise uncorrelated random variables. We can then truncate this expansion and make the approximations

$$u \approx \sum_{j=1}^{d} \sqrt{\lambda_j}\xi_j\phi_j \approx \sum_{j=1}^{d} \sqrt{\lambda_{j,N}}\xi_j\phi_{j,N},$$

where the first approximation corresponds to using the optimal projection subspace V_d while the second approximation replaces V_d with $V_{d,N}$. Since it holds that $\mathbb{E}C_N = C$, we expect $\lambda_j \approx \lambda_{j,N}$ and $\phi_j \approx \phi_{j,N}$. These discussions suggest that the quality of the PCA approximation is controlled, on average, by the rate of decay of the eigenvalues of C, and the approximation of the eigenstructure of C by that of C_N.

2.3. **Neural Networks**

A neural network is a nonlinear function $\chi : \mathbb{R}^n \to \mathbb{R}$ defined by a sequence of compositions of affine maps with point-wise nonlinearities. In particular,

$$\chi(s) = W_t\sigma(\ldots\sigma(W_2\sigma(W_1 s + b_1) + b_2)) + b_t, \qquad s \in \mathbb{R}^n, \tag{2.14}$$

where $W_1, \ldots, W_t$ are *weight matrices* (that are not necessarily square) and $b_1, \ldots, b_t$ are vectors, referred to as *biases*. We refer to $t \geq 1$ as the *depth of the neural network*. The function $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ is a monotone, nonlinear *activation function* that is defined from a monoton function $\sigma : \mathbb{R} \to \mathbb{R}$ applied entrywise to any vector in $\mathbb{R}^d$ with $d \geq 1$. Note that in (2.14) the input dimension of σ may vary between layers but regardless of the input dimension the function σ applies the same operations to all entries of the input vector. We primarily consider the Rectified Linear Unit (ReLU) activation functions, i.e.,

$$\sigma(s) := (\max\{0, s_1\}, \max\{0, s_2\}, \ldots, \max\{0, s_d\})^T \in \mathbb{R}^d \qquad \forall\, s \in \mathbb{R}^d. \tag{2.15}$$

The weights and biases constitute the parameters of the network. In this paper we learn these parameters in the following standard way [49]: given a set of data $\{x_j, y_j\}_{j=1}^N$ we choose the parameters of χ to solve an appropriate regression problem by minimizing a data-dependent cost functional, using stochastic gradient methods. Neural networks have been demonstrated to constitute an efficient class of regressors and interpolators for high-dimensional problems empirically, but a complete theory of

their efficacy is elusive. For an overview of various neural network architectures and their applications, see [33]. For theories concerning their approximation capabilities see [21, 45, 57, 72, 81].

For the approximation results given in Section 3, we will work with a specific class of neural networks, following [81]; we note that other approximation schemes could be used, however, and that we have chosen a proof setting that aligns with, but is not identical to, what we implement in the computations described in Section 4. We will fix $\sigma \in C(\mathbb{R};\mathbb{R})$ to be the ReLU function (2.15) and consider the set of neural networks mapping $\mathbb{R}^n$ to $\mathbb{R}$

$$\mathcal{M}(n;t,r) := \left\{ \begin{aligned} &\chi(s) = W_t\sigma(\dots\sigma(W_2\sigma(W_1 s + b_1) + b_2)) + b_t \in \mathbb{R}, \\ &\text{for all } s \in \mathbb{R}^n \text{ and such that } \textstyle\sum_{k=1}^{t} |W_k|_0 + |b_k|_0 \le r. \end{aligned} \right\}$$

Here $|\cdot|_0$ gives the number of non-zero entries in a matrix so that $r \ge 0$ denotes the number of active weights and biases in the network while $t \ge 1$ is the total number of layers. Moreover, we define the class of stacked neural networks mapping $\mathbb{R}^n$ to $\mathbb{R}^m$:

$$\mathcal{M}(n,m;t,r) := \left\{ \begin{aligned} &\chi(s) = (\chi^{(1)}(s),\dots,\chi^{(m)}(s))^T \in \mathbb{R}^m, \\ &\text{where } \chi^{(j)} \in \mathcal{M}(n;t^{(j)},r^{(j)}), \text{ with } t^{(j)} \le t, r^{(j)} \le r. \end{aligned} \right\}$$

From this, we build the set of *zero-extended* neural networks

$$\mathcal{M}(n,m;t,r,M) := \left\{ \chi = \begin{cases} \tilde{\chi}(s), & s \in [-M,M]^n \\ 0, & s \notin [-M,M]^n \end{cases}, \text{ for some } \tilde{\chi} \in \mathcal{M}(n,m,t,r) \right\},$$

where the new parameter $M > 0$ is the side length of the hypercube in $\mathbb{R}^n$ within which χ can be non-zero. This construction is essential to our approximation as it allows us to handle non-compactness of the latent spaces after PCA dimension reduction.

2.4. **Comparison to Existing Methods**

In the general setting of arbitrary encoders, the formula (2.1c) for the approximation of Ψ yields a complicated map, the representation of which depends on the dimension reduction methods being employed. However, in the setting where PCA is used, a clear representation emerges which we now elucidate in order to highlight similarities and differences between our methodology and existing methods appearing in the literature.

Let $F_{\mathcal{X}} : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{X}}}$ be the PCA encoder w.r.t. the data $\{x_j\}_{j=1}^N$ given by (2.9) and, in particular, let $\phi^{\mathcal{X}}_{1,N},\dots,\phi^{\mathcal{X}}_{d_{\mathcal{X}},N}$ be the eigenvectors of the resulting empirical covariance. Similarly let $\phi^{\mathcal{Y}}_{1,N},\dots,\phi^{\mathcal{Y}}_{d_{\mathcal{Y}},N}$ be the eigenvectors of the empirical covariance w.r.t. the data $\{y\}_{j=1}^N$. For the function φ defined in (2.2), or similarly for approximations χ thereof found through the use of neural networks, we denote the components by $\varphi(s) = (\varphi_1(s),\dots,\varphi_{d_{\mathcal{Y}}}(s))$ for any $s \in \mathbb{R}^{d_{\mathcal{X}}}$. Then (2.1c) becomes $\Psi(x) \approx \sum_{j=1}^{d_{\mathcal{Y}}} \alpha_j(x)\phi^{\mathcal{Y}}_{j,N}$ with coefficients

$$\alpha_j(x) = \varphi_j(F_{\mathcal{X}}(x)) = \varphi_j(\langle x,\phi^{\mathcal{X}}_{1,N}\rangle_{\mathcal{X}},\dots,\langle x,\phi^{\mathcal{X}}_{d_{\mathcal{X}},N}\rangle_{\mathcal{X}}), \qquad \forall\, x \in \mathcal{X}.$$

The solution data $\{y\}_{j=1}^N$ fixes a basis for the output space, and the dependence of $\Psi(x)$ on x is captured solely via the scalar-valued coefficients α_j. This parallels the formulation of the classical reduced basis method [23] where the approximation is written as

$$\Psi(x) \approx \sum_{j=1}^{m} \alpha_j(x)\phi_j. \tag{2.16}$$

Many versions of the method exist, but two particularly popular ones are: (i) when $m = N$ and $\phi_j = y_j$; and (ii) when, as is done here, $m = d_{\mathcal{Y}}$ and $\phi_j = \phi^{\mathcal{Y}}_{j,N}$. The latter choice is also referred to as the reduced basis with a proper orthogonal decomposition.

The crucial difference between our method and the RBM is in the formation of the coefficients α_j. In RBM these functions are obtained in an intrusive manner by approximating the PDE within the finite-dimensional reduced basis and as a consequence the method cannot be used in a setting where a PDE relating inputs and outputs is not known, or may not exist. In contrast, our proposed methodology approximates φ by regressing or interpolating the latent representations $\{F_{\mathcal{X}}(x_j), F_{\mathcal{Y}}(y_j)\}_{j=1}^N$. Thus our proposed method makes use of the entire available dataset and does not require explicit knowledge of the underlying PDE mapping, making it a non-intrusive method applicable to black-box models.

The form (2.16) of the approximate solution operator can also be related to the Taylor approximations developed in [14, 17] where a particular form of the input x is considered, namely $x = \bar{x} + \sum_{j\geq 1} a_j \tilde{x}_j$ where $\bar{x} \in \mathcal{X}$ is fixed, $\{a_j\}_{j\geq 1} \in \ell^\infty(\mathbb{N};\mathbb{R})$ are uniformly bounded, and $\{\tilde{x}_j\}_{j\geq 1} \in \mathcal{X}$ have some appropriate norm decay. Then, assuming that the solution operator $\Psi : \mathcal{X} \to \mathcal{Y}$ is analytic [18], it is possible to make use of the Taylor expansion

$$\Psi(x) = \sum_{h\in\mathcal{F}} \alpha_h(x)\psi_h,$$

where $\mathcal{F} = \{h \in \mathbb{N}^\infty : |h|_0 < \infty\}$ is the set of multi-indices and

$$\alpha_h(x) = \prod_{j\geq 1} a_j^{h_j} \in \mathbb{R}, \qquad \psi_h = \frac{1}{h!}\partial^h\Psi(0) \in \mathcal{Y};$$

here the differentiation ∂^h is with respect to the sequence of coefficients $\{a_j\}_{j\geq 1}$. Then Ψ is approximated by truncating the Taylor expansion to a finite subset of $\mathcal{F}$. For example this may be done recursively, by starting with $h = 0$ and building up the index set in a greedy manner. The method is not data-driven, and requires knowledge of the PDE to define equations to be solved for the ψ_h.

3. Approximation Theory

In this section, we prove our main approximation result: given any $\epsilon > 0$, we can find an $\epsilon-$approximation Ψ_{NN} of Ψ. We achieve this by making the appropriate choice of PCA truncation parameters, by choosing sufficient amounts of data, and by choosing a sufficiently rich neural network architecture to approximate φ by χ.

In what follows we define $F_{\mathcal{X}}$ to be a PCA encoder given by (2.9), using the input data $\{x_j\}_{j=1}^N$ drawn i.i.d. from μ, and $G_{\mathcal{Y}}$ to be a PCA decoder given by (2.10), using the data $\{y_j = \Psi(x_j)\}_{j=1}^N$. We also define

$$\begin{aligned} e_{NN}(x) &= \|(G_{\mathcal{Y}} \circ \chi \circ F_{\mathcal{X}})(x) - \Psi(x)\|_Y \\ &= \|\Psi_{NN}(x) - \Psi(x)\|_{\mathcal{Y}}. \end{aligned}$$

We prove the following theorem:

Theorem 3.1. *Let $\mathcal{X}$, $\mathcal{Y}$ be real, separable Hilbert spaces and let μ be a probability measure supported on $\mathcal{X}$ such that $\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^4 < \infty$. Suppose $\Psi : \mathcal{X} \to \mathcal{Y}$ is a μ-measurable, globally Lipschitz map. For any $\epsilon > 0$, there are dimensions $d_{\mathcal{X}} = d_{\mathcal{X}}(\epsilon) \in \mathbb{N}$, $d_{\mathcal{Y}} = d_{\mathcal{Y}}(\epsilon) \in \mathbb{N}$, a requisite amount of data $N = N(d_{\mathcal{X}}, d_{\mathcal{Y}}) \in \mathbb{N}$, parameters t, r, M depending on $d_{\mathcal{X}}, d_{\mathcal{Y}}$ and ϵ, and a zero-extended stacked neural network $\chi \in \mathcal{M}(d_{\mathcal{X}}, d_{\mathcal{Y}}; t, r, M)$ such that*

$$\mathbb{E}_{\{x_j\}\sim\mu}\mathbb{E}_{x\sim\mu}(e_{NN}(x)^2) < \epsilon.$$

Remark 3.2. This theorem is a consequence of Theorem 3.5 which we state and prove below. For clarity and ease of exposition we state and prove Theorem 3.5 in a setting where Ψ is globally Lipschitz. With a more stringent moment condition on μ, the result can also be proven when Ψ is locally Lipschitz; we state and prove this result in Theorem A.1.

Remark 3.3. The neural network $\chi \in \mathcal{M}(d_{\mathcal{X}}, d_{\mathcal{Y}}; t, r, M)$ has maximum number of layers $t \leq c[\log(M^2 d_{\mathcal{Y}}/\epsilon) + 1]$, with the number of active weights and biases in each component of the network $r \leq c(\epsilon/4M^2)^{-d_{\mathcal{X}}/2}[\log(M^2 d_{\mathcal{Y}}/\epsilon) + 1]$, with an appropriate constant $c = c(d_{\mathcal{X}}, d_{\mathcal{Y}}) \geq 0$ and support side-length $M = M(d_{\mathcal{X}}, d_{\mathcal{Y}}) > 0$. These bounds on t and r follow from Theorem 3.5 with $\tau = \epsilon^{\frac{1}{2}}$. Note, however, that in order to achieve error ϵ, the dimensions $d_{\mathcal{X}}, d_{\mathcal{Y}}$ must be chosen to grow as $\epsilon \to 0$; thus the preceding statements do not explicitly quantify the needed number of parameters, and depth, for error ϵ; to do so would require quantifying the dependence of c, M on $d_{\mathcal{X}}, d_{\mathcal{Y}}$ (a property of neural networks) and the dependence of $d_{\mathcal{X}}, d_{\mathcal{Y}}$ on ϵ (a property of the measure μ and spaces $\mathcal{X}, \mathcal{Y}$ – see Theorem 3.4). The theory in [81], which we employ for the existence result for the neural network produces the constant c which depends on the dimensions $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ in an unspecified way.

The double expectation reflects averaging over all possible new inputs x drawn from μ (inner expectation) and over all possible realizations of the i.i.d. dataset $\{x_j, y_j = \Psi(x_j)\}_{j=1}^N$ (outer expectation). The theorem as stated above is a consequence of Theorem 3.5 in which the error is broken into multiple components that are then bounded separately. Note that the theorem does not address the question of whether the optimization technique used to fit the neural network actually finds the choice which realizes the theorem; this gap between theory and practice is difficult to overcome, because of the non-convex nature of the training problem, and is a standard feature of theorems in this area [45, 72].

The idea of the proof is to quantify the approximations $G_{\mathcal{X}} \circ F_{\mathcal{X}} \approx I_{\mathcal{X}}$ and $G_{\mathcal{Y}} \circ F_{\mathcal{Y}} \approx I_{\mathcal{Y}}$ and $\chi \approx \varphi$ so that Ψ_{NN} given by (2.4) is close to Ψ. The first two approximations, which show that Ψ_{PCA} given by (2.3) is close to Ψ, are studied in Subsection 3.1 (see Theorem 3.4). Then, in Subsection 3.2, we find a neural network χ able to approximate φ to the desired level of accuracy; this fact is part of the proof of Theorem 3.5. The zero-extension of the neural network arises from the fact that we employ a density theorem for a class of neural networks within continuous functions defined on compact sets. Since we cannot guarantee that $F_{\mathcal{X}}$ is bounded, we simply set the neural network output to zero on the set outside a hypercube with side-length $2M$. We then use the fact that this set has small μ-measure, for sufficiently large M.

3.1. **PCA And Approximation**

We work in the general notation and setting of Subsection 2.2 so as to obtain approximation results that are applicable to both using PCA on the inputs and on the outputs. In addition, denote by $(\mathrm{HS}(\mathcal{H}), \langle \cdot, \cdot \rangle_{HS}, \| \cdot \|_{HS})$ the space of Hilbert–Schmidt operators over $\mathcal{H}$. We are now ready to state the main result of this subsection. Our goal is to control the projection error $R(V_{d,N})$ when using the finite-data PCA subspace in place of the optimal projection space since the PCA subspace is what is available in practice. Theorem 3.4 accomplishes this by bounding the error $R(V_{d,N})$ by the optimal error $R(V_d)$ plus a term related to the approximation $V_{d,N} \approx V_d$. While previous results such as [9, 73, 74] focused on bounds for the excess error in probability w.r.t. the data, we present bounds in expectation, averaging over the data. Such bounds are weaker, but allow us to remove strict conditions on the data distribution to obtain more general results; for example, our theory allows for ν to be a Gaussian measure.

Theorem 3.4. *Let R be given by* (2.11) *and $V_{d,N}$, V_d by* (2.7), (2.13) *respectively. Then there exists a constant $Q \geq 0$, depending only on the data generating measure ν, such that*

$$\mathbb{E}_{\{u_j\} \sim \nu}[R(V_{d,N})] \leq \sqrt{\frac{Qd}{N}} + R(V_d),$$

where the expectation is over the dataset $\{u_j\}_{j=1}^N \overset{iid}{\sim} \nu$.

The proof generalizes that employed in [9, Thm. 3.1]. We first find a bound on the average excess error $\mathbb{E}[R(V_{d,N}) - R_N(V_{d,N})]$ using Lemma B.2. Then using Fan's Theorem [27] (Lemma B.1), we bound the average sum of the tail eigenvalues of C_N by the sum of the tail eigenvalues of C, in particular, $\mathbb{E}[R_N(V_{d,N})] \leq R(V_d)$.

Proof. For brevity we simply write $\mathbb{E}$ instead of $\mathbb{E}_{\{u_j\}\sim\nu}$ throughout the proof. For any subspace $V \subseteq \mathcal{H}$, we have

$$\begin{aligned} R(V) &= \mathbb{E}_{u\sim\nu}[\|u\|^2 - 2\langle u, \Pi_V u\rangle + \langle \Pi_V u, \Pi_V u\rangle] = \mathbb{E}_{u\sim\nu}[\mathrm{tr}(u\otimes u) - \langle \Pi_V u, \Pi_V u\rangle] \\ &= \mathbb{E}_{u\sim\nu}[\mathrm{tr}(u\otimes u) - \langle \Pi_V, u\otimes u\rangle_{HS}] = \mathrm{tr}\, C - \langle \Pi_V, C\rangle_{HS} \end{aligned}$$

where we used two properties of the fact that Π_V is an orthogonal projection operator, namely $\Pi_V^2 = \Pi_V = \Pi_V^*$ and

$$\langle \Pi_V, v\otimes z\rangle_{HS} = \langle v, \Pi_V z\rangle = \langle \Pi_V v, \Pi_V z\rangle \quad \forall\, v, z \in \mathcal{H}.$$

Repeating the above arguments for $R_N(V)$ in place of $R(V)$, with the expectation replaced by the empirical average, yields $R_N(V) = \mathrm{tr}\, C_N - \langle \Pi_V, C_N\rangle_{HS}$. By noting that $\mathbb{E}[C_N] = C$ we then write

$$\begin{aligned} \mathbb{E}[R(V_{d,N}) - R_N(V_{d,N})] &= \mathbb{E}\langle \Pi_{V_{d,N}}, C_N - C\rangle_{HS} \leq \sqrt{d}\,\mathbb{E}\|C_N - C\|_{HS} \\ &\leq \sqrt{d}\sqrt{\mathbb{E}\|C_N - C\|_{HS}^2} \end{aligned}$$

where we used Cauchy–Schwarz twice along with the fact that $\|\Pi_{V_{d,N}}\|_{HS} = \sqrt{d}$ since $V_{d,N}$ is d-dimensional. Now by Lemma B.2, which quantifies the Monte Carlo error between C and C_N in the Hilbert–Schmidt norm, we have that

$$\mathbb{E}[R(V_{d,N}) - R_N(V_{d,N})] \leq \sqrt{\frac{Qd}{N}},$$

for a constant $Q \geq 0$. Hence by(2.8),

$$\mathbb{E}[R(V_{d,N})] \leq \sqrt{\frac{Qd}{N}} + \mathbb{E}\sum_{j=d+1}^{N} \lambda_{j,N}.$$

It remains to estimate the second term above. Letting S_d denote the set of subspaces of d orthonormal elements in $\mathcal{H}$, Fan's Theorem (Proposition B.1) gives

$$\begin{aligned} \sum_{j=1}^{d} \lambda_j &= \max_{\{v_1,\dots,v_d\}\in S_d} \sum_{j=1}^{d} \langle Cv_j, v_j\rangle = \max_{\{v_1,\dots,v_d\}\in S_d} \mathbb{E}_{u\sim\nu} \sum_{j=1}^{d} |\langle u, v_j\rangle|^2 \\ &= \max_{\{v_1,\dots,v_d\}\in S_d} \mathbb{E}_{u\sim\nu} \sum_{j=1}^{d} \|\Pi_{\mathrm{span}\{v_j\}} u\|^2 = \max_{V\in\mathcal{V}_d} \mathbb{E}_{u\sim\nu}\|\Pi_V u\|^2 \\ &= \mathbb{E}_{u\sim\nu}\|u\|^2 - \min_{V\in\mathcal{V}_d} E_{u\sim\nu}\|\Pi_{V^\perp} u\|^2. \end{aligned}$$

Observe that $\sum_{j=1}^{\infty} \lambda_j = \mathrm{tr}\, C = \mathbb{E}_{u\sim\nu}\|u\|^2$ and so

$$\sum_{j=d+1}^{\infty} \lambda_j = \mathbb{E}_{u\sim\nu}\|u\|^2 - \sum_{j=1}^{d} \lambda_j = \min_{V\in\mathcal{V}_d} \mathbb{E}_{u\sim\nu}\|\Pi_{V^\perp} u\|^2.$$

We now repeat the above calculations for $\lambda_{j,N}$, the eigenvalues of C_N, by replacing the expectation with the empirical average to obtain

$$\sum_{j=d+1}^{N} \lambda_{j,N} = \min_{V\in\mathcal{V}_d} \frac{1}{N} \sum_{k=1}^{N} \|\Pi_{V^\perp} u_k\|^2,$$

and so

$$\mathbb{E}\sum_{j=d+1}^{N}\lambda_{j,N} \leq \min_{V\in\mathcal{V}_d}\mathbb{E}\frac{1}{N}\sum_{k=1}^{N}\|\Pi_{V^\perp}u_k\|^2 = \min_{V\in\mathcal{V}_d}\mathbb{E}_{u\sim\mu}\|\Pi_{V^\perp}u\|^2 = \sum_{j=d+1}^{\infty}\lambda_j.$$

Finally, we conclude that

$$\mathbb{E}[R(V_{d,N})] \leq \sqrt{\frac{Qd}{N}} + \sum_{j=d+1}^{\infty}\lambda_j = \sqrt{\frac{Qd}{N}} + R(V_d).$$

■

3.2. **Neural Networks And Approximation**

In this subsection we study the approximation of φ given in (2.2) by neural networks, combining the analysis with results from the preceding subsection to prove our main approximation result, Theorem 3.5. We will work in the notation of Section 2. We assume that $(\mathcal{X}, \langle\cdot,\cdot\rangle_{\mathcal{X}}, \|\cdot\|_{\mathcal{X}})$ and $(\mathcal{Y}, \langle\cdot,\cdot\rangle_{\mathcal{Y}}, \|\cdot\|_{\mathcal{Y}})$ are real, separable Hilbert spaces; μ is a probability measure supported on $\mathcal{X}$ with a finite fourth moment $\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^4 < \infty$, and $\Psi : \mathcal{X} \to \mathcal{Y}$ is measurable and globally L-Lipschitz: there exists a constant $L > 0$ such that

$$\forall\, x, z \in \mathcal{X} \quad \|\Psi(x) - \Psi(z)\|_{\mathcal{Y}} \leq L\|x - z\|_{\mathcal{X}}.$$

Note that this implies that Ψ is linearly bounded: for any $x \in \mathcal{X}$

$$\|\Psi(x)\|_{\mathcal{Y}} \leq \|\Psi(0)\|_{\mathcal{Y}} + \|\Psi(x) - \Psi(0)\|_{\mathcal{Y}} \leq \|\Psi(0)\|_{\mathcal{Y}} + L\|x\|_{\mathcal{X}}.$$

Hence we deduce existence of the fourth moment of the pushforward $\Psi_\sharp\mu$:

$$\mathbb{E}_{y\sim\Psi_\sharp\mu}\|y\|_{\mathcal{Y}}^4 = \int_{\mathcal{X}}\|\Psi(x)\|_{\mathcal{Y}}^4 d\mu(x) \leq \int_{\mathcal{X}}(\|\Psi(0)\|_{\mathcal{Y}} + L\|x\|_{\mathcal{X}})^4 d\mu(x) < \infty$$

since we assumed $\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^4 < \infty$.

Let us recall some of the notation from Subsections 2.2 and 2.4. Let $V_{d_{\mathcal{X}}}^{\mathcal{X}}$ be the $d_{\mathcal{X}}$-dimensional optimal projection space given by (2.13) for the measure μ and $V_{d_{\mathcal{X}},N}^{\mathcal{X}}$ be the $d_{\mathcal{X}}$-dimensional PCA subspace given by (2.7) with respect to the input dataset $\{x_j\}_{j=1}^N$. Similarly let $V_{d_{\mathcal{Y}}}^{\mathcal{Y}}$ be the $d_{\mathcal{Y}}$-dimensional optimal projection space for the pushforward measure $\Psi_\sharp\mu$ and $V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}$ be the $d_{\mathcal{Y}}$-dimensional PCA subspace with respect to the output dataset $\{y_j = \Psi(x_j)\}_{j=1}^N$. We then define the input PCA encoder $F_{\mathcal{X}} : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{X}}}$ by (2.9) and the input PCA decoder $G_{\mathcal{X}} : \mathbb{R}^{d_{\mathcal{X}}} \to \mathcal{X}$ by (2.10) both with respect to the orthonormal basis used to construct $V_{d_{\mathcal{X}},N}^{\mathcal{X}}$. Similarly we define the output PCA encoder $F_{\mathcal{Y}} : \mathcal{Y} \to \mathbb{R}^{d_{\mathcal{Y}}}$ and decoder $G_{\mathcal{Y}} : \mathbb{R}^{d_{\mathcal{Y}}} \to \mathcal{Y}$ with respect to the orthonormal basis used to construct $V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}$. Finally we recall $\varphi : \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}^{d_{\mathcal{Y}}}$ the map connecting the two latent spaces defined in (2.2). The approximation Ψ_{PCA} to Ψ based only on the PCA encoding and decoding is given by (2.3). In the following theorem, we prove the existence of a neural network giving an ϵ-close approximation to φ for fixed latent code dimensions $d_{\mathcal{X}}, d_{\mathcal{Y}}$ and quantify the error of the full approximation Ψ_{NN}, given in (2.4), to Ψ. We will be explicit about which measure the projection error is defined with respect to. In particular, we will write (2.11) as

$$R^{\mu}(V) = \mathbb{E}_{x\sim\mu}\|x - \Pi_V x\|_{\mathcal{X}}^2$$

for any subspace $V \subseteq \mathcal{X}$ and similarly

$$R^{\Psi_\sharp\mu}(V) = \mathbb{E}_{y\sim\Psi_\sharp\mu}\|y - \Pi_V y\|_{\mathcal{Y}}^2$$

for any subspace $V \subseteq \mathcal{Y}$.

Theorem 3.5. *Let $\mathcal{X}$, $\mathcal{Y}$ be real, separable Hilbert spaces and let μ be a probability measure supported on $\mathcal{X}$ such that $\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^4 < \infty$. Suppose $\Psi : \mathcal{X} \to \mathcal{Y}$ is a μ-measurable, globally Lipschitz map. Fix $d_{\mathcal{X}}, d_{\mathcal{Y}}$, $N \geq \max\{d_{\mathcal{X}}, d_{\mathcal{Y}}\}$, $\delta \in (0,1)$ and $\tau > 0$. Define $M = \sqrt{\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^2/\delta}$. Then there exists a constant $c = c(d_{\mathcal{X}}, d_{\mathcal{Y}}) \geq 0$ and a zero-extended stacked neural network $\chi \in \mathcal{M}(d_{\mathcal{X}}, d_{\mathcal{Y}}; t, r, M)$ with $t \leq c(d_{\mathcal{X}}, d_{\mathcal{Y}})[\log(M\sqrt{d_{\mathcal{Y}}}/\tau) + 1]$ and $r \leq c(d_{\mathcal{X}}, d_{\mathcal{Y}})(\tau/2M)^{-d_{\mathcal{X}}}[\log(M\sqrt{d_{\mathcal{Y}}}/\tau) + 1]$, so that*

$$\mathbb{E}_{\{x_j\}\sim\mu}\mathbb{E}_{x\sim\mu}\big(e_{NN}(x)\big)^2 \leq C\left(\tau^2 + \sqrt{\delta} + \sqrt{\frac{d_{\mathcal{X}}}{N}} + R^{\mu}(V_{d_{\mathcal{X}}}^{\mathcal{X}}) + \sqrt{\frac{d_{\mathcal{Y}}}{N}} + R^{\Psi_\sharp\mu}(V_{d_{\mathcal{Y}}}^{\mathcal{Y}})\right), \tag{3.1}$$

where $C > 0$ is independent of $d_{\mathcal{X}}, d_{\mathcal{Y}}, N, \delta$ and τ.

The first two terms on the r.h.s. arise from the neural network approximation of φ while the last two pairs of terms are from the finite-dimensional approximation of $\mathcal{X}$ and $\mathcal{Y}$ respectively as prescribed by Theorem 3.4. The way to interpret the result is as follows: first choose $d_{\mathcal{X}}, d_{\mathcal{Y}}$ so that $R^{\mu}(V_{d_{\mathcal{X}}}^{\mathcal{X}})$ and $R^{\Psi_\sharp\mu}(V_{d_{\mathcal{Y}}}^{\mathcal{Y}})$ are small – these are intrinsic properties of the measures μ and $\Psi_\sharp\mu$; secondly, choose the amount of data N large enough to make $\max\{d_{\mathcal{X}}, d_{\mathcal{Y}}\}/N$ small, essentially controlling how well we approximate the intrinsic covariance structure of μ and $\Psi_\sharp\mu$ using samples; thirdly choose δ small enough to control the error arising from restricting the domain of φ; and finally choose τ sufficiently small to control the approximation of φ by a neural network restricted to a compact set. Note that the size and values of the parameters of the neural network χ will depend on the choice of δ as well as $d_{\mathcal{X}}, d_{\mathcal{Y}}$ and N in a manner which we do not specify. In particular, the dependence of c on $d_{\mathcal{X}}, d_{\mathcal{Y}}$ is not explicit in the theorem of [81] which furnishes the existence of the requisite neural network χ. The parameter τ specifies the error tolerance between χ and φ on $[-M, M]^{d_{\mathcal{X}}}$. Intuitively, as $(\delta, \tau) \to 0$, we expect the number of parameters in the network to also grow [57]. Quantifying this growth would be needed to fully understand the computational complexity of our method.

Proof. Recall the constant Q from Theorem 3.4. In what follows we take Q to be the maximum of the two such constants when arising from application of the theorem on the two different probability spaces $(\mathcal{X}, \mu)$ and $(\mathcal{Y}, \Psi_\sharp\mu)$. Through the proof we use $\mathbb{E}$ to denote $\mathbb{E}_{\{x_j\}\sim\mu}$ the expectation with respect to the dataset $\{x_j\}_{j=1}^N$.
We begin by approximating the error incurred by using Ψ_{PCA} given by (2.3):

$$\begin{aligned}
\mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{PCA}(x) - \Psi(x)\|_{\mathcal{Y}}^2 &= \mathbb{E}\mathbb{E}_{x\sim\mu}\|(G_{\mathcal{Y}} \circ F_{\mathcal{Y}} \circ \Psi \circ G_{\mathcal{X}} \circ F_{\mathcal{X}})(x) - \Psi(x)\|_{\mathcal{Y}}^2 \\
&= \mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x) - \Psi(x)\right\|_{\mathcal{Y}}^2 \\
&\leq 2\mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x) - \Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(x)\right\|_{\mathcal{Y}}^2 \\
&\quad + 2\mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(x) - \Psi(x)\right\|_{\mathcal{Y}}^2 \\
&\leq 2L^2\mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x - x\right\|_{\mathcal{X}}^2 + 2\mathbb{E}\mathbb{E}_{y\sim\Psi_\sharp\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}y - y\right\|_{\mathcal{Y}}^2 \\
&= 2L^2\mathbb{E}[R^{\mu}(V_{d_{\mathcal{X}},N}^{\mathcal{X}})] + 2\mathbb{E}[R^{\Psi_\sharp\mu}(V_{d_{\mathcal{Y}},N}^{\mathcal{Y}})]
\end{aligned} \tag{3.2}$$

noting that the operator norm of an orthogonal projection is 1. Theorem 3.4 allows us to control this error, and leads to

$$\begin{aligned}\mathbb{E}\mathbb{E}_{x\sim\mu}(e_{NN}(x)^2) &= \mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{NN}(x)-\Psi(x)\|_{\mathcal{Y}}^2 \\ &\le 2\mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{NN}(x)-\Psi_{PCA}(x)\|_{\mathcal{Y}}^2 + 2\mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{PCA}(x)-\Psi(x)\|_{\mathcal{Y}}^2 \\ &\le 2\mathbb{E}_{x\sim\mu}\|\Psi_{NN}(x)-\Psi_{PCA}(x)\|_{\mathcal{Y}}^2 \\ &\quad + 4L^2\left(\sqrt{\frac{Qd_{\mathcal{X}}}{N}} + R^{\mu}(V_{d_{\mathcal{X}}}^{\mathcal{X}})\right) + 4\left(\sqrt{\frac{Qd_{\mathcal{Y}}}{N}} + R^{\Psi\sharp\mu}(V_{d_{\mathcal{Y}}}^{\mathcal{Y}})\right).\end{aligned} \tag{3.3}$$

We now approximate φ by a neural network χ as a step towards estimating $\|\Psi_{NN}(x)-\Psi_{PCA}(x)\|_{\mathcal{Y}}$. To that end we first note from Lemma B.3 that φ is Lipschitz, and hence continuous, as a mapping from $\mathbb{R}^{d_{\mathcal{X}}}$ into $\mathbb{R}^{d_{\mathcal{Y}}}$. Identify the components $\varphi(s) = (\varphi^{(1)}(s),\dots,\varphi^{(d_{\mathcal{Y}})}(s))$ where each function $\varphi^{(j)} \in C(\mathbb{R}^{d_{\mathcal{X}}};\mathbb{R})$. We consider the restriction of each component function to the set $[-M,M]^{d_{\mathcal{X}}}$. Let us now change variables by defining $\tilde{\varphi}^{(j)} : [0,1]^{d_{\mathcal{X}}} \to \mathbb{R}$ by $\tilde{\varphi}^{(j)}(s) = (1/2M)\varphi^{(j)}(2Ms-M)$ for any $s \in [0,1]^{d_{\mathcal{X}}}$. Note that equivalently we have $\varphi^{(j)}(s) = 2M\tilde{\varphi}^{(j)}((s+M)/2M)$ for any $s \in [-M,M]^{d_{\mathcal{X}}}$ and further $\varphi^{(j)}$ and $\tilde{\varphi}^{(j)}$ have the same Lipschitz constants on their respective domains. Applying [81, Thm. 1] to the $\tilde{\varphi}^{(j)}$(s) then yields existence of neural networks $\tilde{\chi}^{(1)},\dots,\tilde{\chi}^{(d_{\mathcal{Y}})} : [0,1]^{d_{\mathcal{X}}} \to \mathbb{R}$ such that

$$|\tilde{\chi}^{(j)}(s) - \tilde{\varphi}^{(j)}(s)| < \frac{\tau}{2M\sqrt{d_{\mathcal{Y}}}} \quad \forall\, s \in [0,1]^{d_{\mathcal{X}}},$$

for any $j \in \{1,\dots,d_{\mathcal{Y}}\}$. In fact, each neural network $\tilde{\chi}^{(j)} \in \mathcal{M}(d_{\mathcal{X}};t^{(j)},r^{(j)})$ with parameters $t^{(j)}$ and $r^{(j)}$ satisfying

$$t^{(j)} \le c^{(j)}\left[\log(M\sqrt{d_{\mathcal{Y}}}/\tau)+1\right], \qquad r^{(j)} \le c^{(j)}\left(\frac{\tau}{2M}\right)^{-d_{\mathcal{X}}}\left[\log(M\sqrt{d_{\mathcal{Y}}}/\tau)+1\right],$$

with constants $c^{(j)}(d_{\mathcal{X}}) > 0$. Hence defining $\chi^{(j)} : \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}$ by $\chi^{(j)}(s) := 2M\tilde{\chi}^{(j)}((s+M)/2M)$ for any $s \in [-M,M]^{d_{\mathcal{X}}}$, we have that

$$\left|(\chi^{(1)}(s),\dots,\chi^{(d_{\mathcal{Y}})}(s)) - \varphi(s)\right|_2 < \tau \quad \forall\, s \in [-M,M]^{d_{\mathcal{X}}}.$$

We can now simply define $\chi : \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}^{d_{\mathcal{Y}}}$ as the stacked network $(\chi^{(1)},\dots,\chi^{d_{\mathcal{Y}}})$ extended by zero outside of $[-M,M]^{d_{\mathcal{X}}}$ to immediately obtain

$$\sup_{s\in[-M,M]^{d_{\mathcal{X}}}} \left|\chi(s)-\varphi(s)\right|_2 < \tau. \tag{3.4}$$

Thus, by construction $\chi \in \mathcal{M}(d_{\mathcal{X}},d_{\mathcal{Y}},t,r,M)$ with at most $t \le \max_j t^{(j)}$ many layers and $r \le r^{(j)}$ many active weights and biases in each of its components.

Let us now define the set $A = \{x \in \mathcal{X} : F_{\mathcal{X}}(x) \in [-M,M]^{d_{\mathcal{X}}}\}$. By Lemma B.4, $\mu(A) \ge 1-\delta$ and $\mu(A^c) \le \delta$. Define the approximation error

$$e_{PCA}(x) = \|\Psi_{NN}(x) - \Psi_{PCA}(x)\|_{\mathcal{Y}}$$

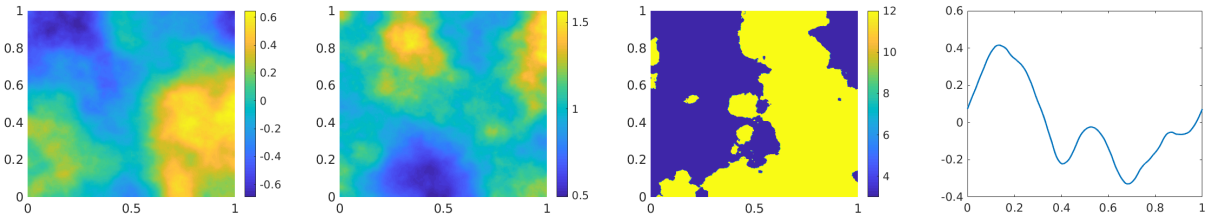
and decompose its expectation as

$$\mathbb{E}_{x\sim\mu}(e_{PCA}(x)^2) = \underbrace{\int_A e_{PCA}(x)^2 d\mu(x)}_{:=I_A} + \underbrace{\int_{A^c} e_{PCA}(x)^2 d\mu(x)}_{:=I_{A^c}}.$$

For the first term,

$$I_A \le \int_A \|(G_{\mathcal{Y}}\circ\chi\circ F_{\mathcal{X}})(x) - (G_{\mathcal{Y}}\circ\varphi\circ F_{\mathcal{X}})(x)\|_{\mathcal{Y}}^2 d\mu(x) \le \tau^2, \tag{3.5}$$

by using the fact, established in Lemma B.3, that $G_{\mathcal{Y}}$ is Lipschitz with Lipschitz constant 1, the τ-closeness of χ to φ from (3.4), and $\mu(A) \le 1$. For the second term we have, using that $G_{\mathcal{Y}}$ has



(a) μ_{G} (b) μ_{L} (c) μ_{P} (d) μ_{B}

FIGURE 4.1. Representative samples for each of the probability measures $\mu_{\mathrm{G}}, \mu_{\mathrm{L}}, \mu_{\mathrm{P}}, \mu_{\mathrm{B}}$ defined in Subsection 4.1. μ_{G} and μ_{P} are used in Subsection 4.2 to model the inputs, μ_{L} and μ_{P} are used in Subsection 4.3, and μ_{B} is used in Subsection 4.4.

Lipschitz constant 1 and that χ vanishes on A^c,

$$\begin{aligned} I_{A^c} &\leq \int_{A^c} \|(G_{\mathcal{Y}} \circ \chi \circ F_{\mathcal{X}})(x) - (G_{\mathcal{Y}} \circ \varphi \circ F_{\mathcal{X}})(x)\|_{\mathcal{Y}}^2 d\mu(x) \\ &\leq \int_{A^c} |\chi(F_{\mathcal{X}}(x)) - \varphi(F_{\mathcal{X}}(x))|_2^2 d\mu(x) = \int_{A^c} |\varphi(F_{\mathcal{X}}(x))|_2^2 d\mu(x). \end{aligned} \tag{3.6}$$

Once more from Lemma B.3, we have that

$$|F_{\mathcal{X}}(x)|_2 \leq \|x\|_{\mathcal{X}}; \quad |\varphi(x)|_2 \leq |\varphi(0)|_2 + L|x|_2,$$

so that

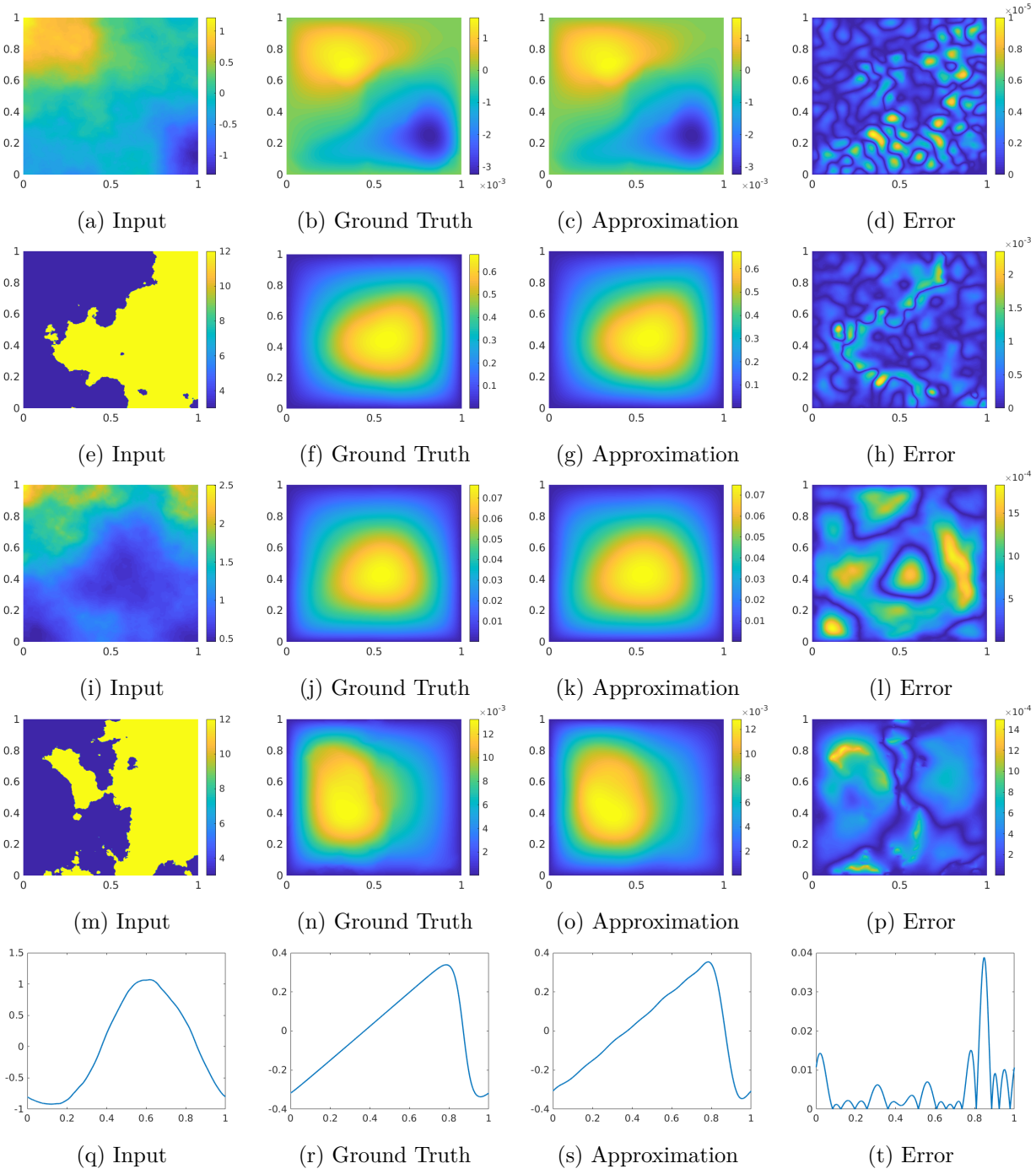
$$\begin{aligned} I_{A^c} &\leq 2\big(\mu(A^c)|\varphi(0)|_2^2 + \mu(A_c)^{\frac{1}{2}} L^2 (\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^4)^{\frac{1}{2}}\big), \\ &\leq 2\big(\delta|\varphi(0)|_2^2 + \delta^{\frac{1}{2}} L^2 (\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^4)^{\frac{1}{2}}\big). \end{aligned} \tag{3.7}$$

Combining (3.3), (3.5) and (3.7), we obtain the desired result. ∎

4. Numerical Results

We now present a series of numerical experiments that demonstrate the effectiveness of our proposed methodology in the context of the approximation of parametric PDEs. We work in settings which both verify our theoretical results and show that the ideas work outside the confines of the theory. The key idea underlying our work is to construct the neural network architecture so that it is defined *as a map between Hilbert spaces* and only then to discretize and obtain a method that is implementable in practice; prevailing methodologies first discretize and then apply a standard neural network. Our approach leads, when discretized, to methods that have properties which are uniform with respect to the mesh size used. We demonstrate this through our numerical experiments. In practice, we obtain an approximation Ψ_{num} to Ψ_{NN}, reflecting the numerical discretization used, and the fact that μ and its pushforward under Ψ are only known to us through samples and, in particular, samples of the pushforward of μ under the numerical approximation of the input-output map. However since, as we will show, our method is robust to the discretization used, we will not explicitly reflect the dependence of the numerical method in the notation that appears in the remainder of this section.

In Subsection 4.1 we introduce a class of parametric elliptic PDEs arising from the Darcy model of flow in porous media, as well as the time-dependent, parabolic, Burgers' equation, that define a variety of input-output maps for our numerical experiments; we also introduce the probability measures that we use on the input spaces. Subsection 4.2 presents numerical results for a Lipschitz map. Subsections 4.3, 4.4 present numerical results for the Darcy flow problem and the flow map



(a) Input (b) Ground Truth (c) Approximation (d) Error

(e) Input (f) Ground Truth (g) Approximation (h) Error

(i) Input (j) Ground Truth (k) Approximation (l) Error

(m) Input (n) Ground Truth (o) Approximation (p) Error

(q) Input (r) Ground Truth (s) Approximation (t) Error

Figure 4.2. Randomly chosen examples from the test set for each of the five considered problems. Each row is a different problem: linear elliptic, Poisson, Darcy flow with log-normal coefficients, Darcy flow with piecewise constant coefficients, and Burgers' equation respectively from top to bottom. The approximations are constructed with our best performing method (for $N = 1024$): Linear $d = 150$, Linear $d = 150$, NN $d = 70$, NN $d = 70$, NN $d = 15$ respectively from top to bottom.

for the Burgers' equation; this leads to non-Lipschitz input-output maps, beyond our theoretical developments. We emphasize that while our method is designed for approximating nonlinear operators Ψ, we include some numerical examples where Ψ is linear. Doing so is helpful for confirming some of our theory and comparing against other methods in the literature. Note that when Ψ is linear, each piece in the approximate decomposition (2.3) is also linear, in particular, φ is linear. Therefore it is sufficient to parameterize φ as a linear map (matrix of unknown coefficients) instead of a neural network. We include such experiments in Section 4.2 revealing that, while a neural network approximating φ arbitrarily well exists, the optimization methods used for training the neural network fail to find it. It may therefore be beneficial to directly build into the parametrization known properties of φ, such as linearity, when they are known. We emphasize that, for general nonlinear maps, linear methods significantly underperform in comparison with our neural network approximation and we will demonstrate this for the Darcy flow problem, and for Burgers' equation.

We use standard implementations of PCA, with dimensions specified for each computational example below. All computational examples use an identical neural network architecture: a 5-layer dense network with layer widths $500, 1000, 2000, 1000, 500$, ordered from first to last layer, and the SELU nonlinearity [43]. We note that Theorem 3.5 requires greater depth for greater accuracy but that we have found our 5-layer network to suffice for all of the examples described here. Thus we have not attempted to optimize the architecture of the neural network. We use stochastic gradient descent with Nesterov momentum (0.99) to train the network parameters [33], each time picking the largest learning rate that does not lead to blow-up in the error. While the network must be re-trained for each new choice of reduced dimensions $d_{\mathcal{X}}, d_{\mathcal{Y}}$, initializing the hidden layers with a pre-trained network can help speed up convergence.

4.1. **PDE Setting**

We will consider a variety of solution maps defined by second order elliptic PDEs of the form (1.1). which are prototypical of many scientific applications. We take $D = (0,1)^2$ to be the unit box, $a \in L^\infty(D;\mathbb{R}_+), f \in L^2(D;\mathbb{R})$, and let $u \in H_0^1(D;\mathbb{R})$ be the unique weak solution of (1.1). Note that, since D is bounded, $L^\infty(D;\mathbb{R}_+)$ is continuously embedded within the Hilbert space $L^2(D;\mathbb{R}_+)$. We will consider two variations of the input-output map generated by the solution operator for (1.1); in one, it is Lipschitz and lends itself to the theory of Subsection 3.2 and, in the other, it is not Lipschitz. We obtain numerical results which demonstrate our theory as well as demonstrating the effectiveness of our proposed methodology in the non-Lipschitz setting.

Furthermore, we consider the one-dimensional viscous Burgers' equation on the torus given as

$$\begin{aligned} \frac{\partial}{\partial t}u(s,t) + \frac{1}{2}\frac{\partial}{\partial s}(u(s,t))^2 = \beta\frac{\partial^2}{\partial s^2}u(s,t), \qquad & (s,t) \in \mathbb{T}^1 \times (0,\infty) \\ u(s,0) = u_0(s), \qquad & s \in \mathbb{T}^1 \end{aligned} \tag{4.1}$$

where $\beta > 0$ is the viscosity coefficient and $\mathbb{T}^1$ is the one dimensional unit torus obtained by equipping the interval $[0,1]$ with periodic boundary conditions. We take $u_0 \in L^2(\mathbb{T}^1;\mathbb{R})$ and have that, for any $t > 0$, $u(\cdot\,,t) \in H^r(\mathbb{T}^1;\mathbb{R})$ for any $r > 0$ is the unique weak solution to (4.1) [78]. In Subsection 4.4, we consider the input-output map generated by the flow map of (4.1) evaluated at a fixed time which is a locally Lipschitz operator.

We make use of four probability measures which we now describe. The first, which will serve as a base measure in two dimensions, is the Gaussian $\mu_{\mathrm{G}} = \mathcal{N}(0, (-\Delta + 9I)^{-2})$ with a zero Neumann boundary condition on the operator Δ. Then we define μ_{L} to be the log-normal measure defined as the push-forward of μ_{G} under the exponential map i.e. $\mu_{\mathrm{L}} = \exp_\sharp \mu_{\mathrm{G}}$. Furthermore, we define $\mu_{\mathrm{P}} = T_\sharp \mu_{\mathrm{G}}$

to be the push-forward of μ_{G} under the piecewise constant map

$$T(s) = \begin{cases} 12 & s \geq 0, \\ 3 & s < 0. \end{cases}$$

Lastly, we consider the Gaussian $\mu_{\mathrm{B}} = \mathcal{N}(0, 7^4(-\frac{d^2}{ds^2} + 7^2 I)^{-2.5})$ defined on $\mathbb{T}^1$. Figure 4.1 shows an example draw from each of the above measures. We will use as μ one of these four measures in each experiment we conduct. Such probability measures are commonly used in the stochastic modeling of physical phenomenon [53]. For example, μ_{P} may be thought of as modeling the permeability of a porous medium containing two different constituent parts [41]. Note that it is to be expected that a good choice of architecture will depend on the probability measure used to generate the inputs. Indeed good choices of the reduced dimensions $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ are determined by the input measure and its pushforward under Ψ, respectively.

For each subsequently described problem we use, unless stated otherwise, $N = 1024$ training examples from μ and its pushforward under Ψ, from which we construct Ψ_{NN}, and then 5000 unseen testing examples from μ in order to obtain a Monte Carlo estimate of the relative test error:

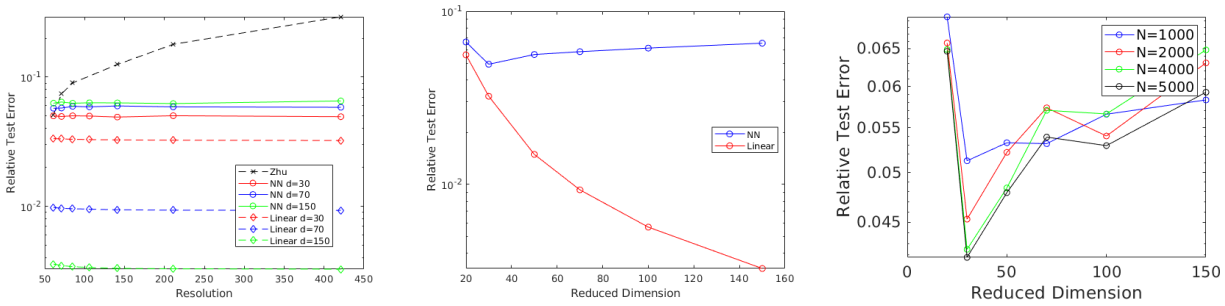
$$\mathbb{E}_{x\sim\mu} \frac{\|(G_2 \circ \chi \circ F_1)(x) - \Psi(x)\|_{\mathcal{Y}}}{\|\Psi(x)\|_{\mathcal{Y}}}.$$

For problems arising from (1.1), all data is collected on a uniform 421×421 mesh and the PDE is solved with a second order finite-difference scheme. For problems arising from (4.1), all data is collected on a uniform 4096 point mesh and the PDE is solved using a pseudo-spectral method. Data for all other mesh sizes is sub-sampled from the original. We refer to the size of the discretization in one direction e.g. 421, as the *resolution*. We fix $d_{\mathcal{X}} = d_{\mathcal{Y}}$ (the dimensions after PCA in the input and output spaces) and refer to this as *the reduced dimension*. We experiment with using a linear map as well as a dense neural network for approximating φ; in all figures we distinguish between these by referring to Linear or NN approximations respectively. When parameterizing with a neural network, we use the aforementioned stochastic gradient based method for training, while, when parameterizing with a linear map, we simply solve the linear least squares problem by the standard normal equations.

We also compare all of our results to the work of [83] which utilizes a 19-layer fully-connected convolutional neural network, referencing this approach as Zhu within the text. This is done to show that the image-to-image regression approach that many such works utilize yields approximations that are not consistent in the continuum, and hence across different discretizations; in contrast, our methodology is designed as a mapping between Hilbert spaces and as a consequence is robust across different discretizations. For some problems in Subsection 4.2, we compare to the method developed in [14], which we refer to as Chkifa. For the problems in Subsection 4.3, we also compare to the reduced basis method [23, 67] when instantiated with PCA. We note that both Chkifa and the reduced basis method are intrusive, i.e., they need knowledge of the governing PDE. Furthermore the method of Chkifa needs full knowledge of the generating process of the inputs. We re-emphasize that our proposed method is fully data-driven.

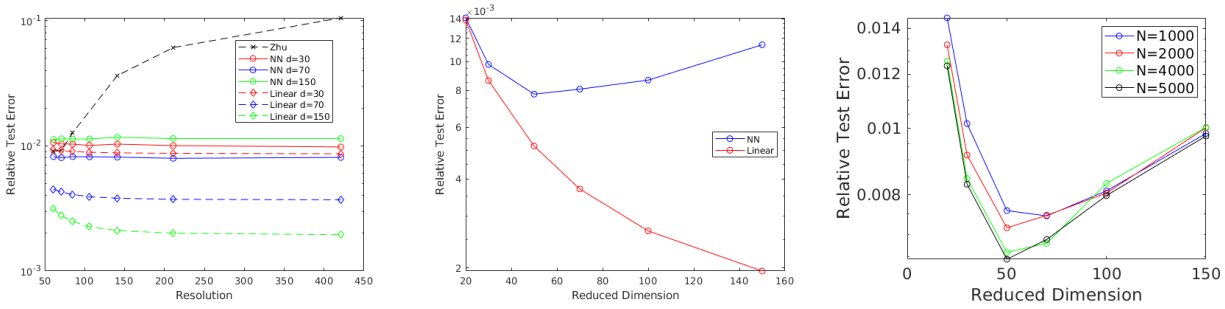
4.2. **Globally Lipschitz Solution Map**

We consider the input-output map $\Psi : L^2(D;\mathbb{R}) \to H_0^1(D;\mathbb{R})$ mapping $f \mapsto u$ in (1.1) with the coefficient a fixed. Since (1.1) is a linear PDE, Ψ is linear and therefore Lipschitz. We study two instantiations of this problem. In the first, we draw a single $a \sim \mu_{\mathrm{P}}$ and fix it. We then solve (1.1) with data $f \sim \mu_{\mathrm{G}}$. We refer to this as the *linear elliptic* problem. See row 1 of Figure 4.2 for an example. In the second, we set $a(w) = 1 \ \forall \ w \in D$, in which case (1.1) becomes the Poisson equation which we solve with data $f \sim \mu = \mu_{\mathrm{G}}$. We refer to this as the *Poisson* problem. See row 2 of Figure 4.2 for an example.



(a) (b) (c)

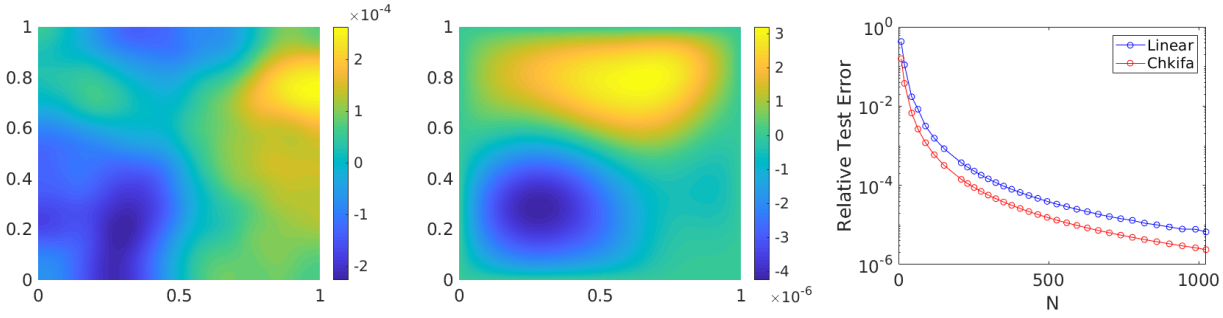
FIGURE 4.3. Relative test errors on the linear elliptic problem. Using $N = 1024$ training examples, panel (a) shows the errors as a function of the resolution while panel (b) fixes a 421×421 mesh and shows the error as a function of the reduced dimension. Panel (c) only shows results for our method using a neural network, fixing a 421×421 mesh and showing the error as a function of the reduced dimension for different amounts of training data.



(a) (b) (c)

FIGURE 4.4. Relative test errors on the Poisson problem. Using $N = 1024$ training examples, panel (a) shows the errors as a function of the resolution while panel (b) fixes a 421×421 mesh and shows the error as a function of the reduced dimension. Panel (c) only shows results for our method using a neural network, fixing a 421×421 mesh and showing the error as a function of the reduced dimension for different amounts of training data.

Figure 4.3 (a) shows the relative test errors as a function of the resolution on the linear elliptic problem, while Figure 4.4 (a) shows them on the Poisson problem. The primary observation to make about panel (a) in these two figures is that it shows that the error in our proposed method does not change as the resolution changes. In contrast, it also shows that the image-to-image regression approach of Zhu [83], whilst accurate at low mesh resolution, fails to be invariant to the size of the discretization and errors increase in an uncontrolled fashion as greater resolution is used. The fact that our dimension reduction approach achieves constant error as we refine the mesh, reflects its design as a method on Hilbert space which may be approximated consistently on different meshes. Since the operator Ψ here is linear, the true map of interest φ given by (2.2) is linear since $F_{\mathcal{Y}}$ and $G_{\mathcal{X}}$ are, by the definition of PCA, linear. It is therefore unsurprising that the linear approximation consistently outperforms the neural network, a fact also demonstrated in panel (a) of the two figures. While it is theoretically possible to find a neural network that can, at least, match the performance



(a) (b) (c)

Figure 4.5. Panel (a) shows a sample drawn from the model (4.2) while panel (b) shows the solution of the Poisson equation with the sample from (a) as the r.h.s. Panel (c) shows the relative test error as a function of the amount of PDE solves/ training data for the method of Chkifa and our method respectively. We use the reduced dimension $d = N$.

of the linear map, in practice, the non-convexity of the associated optimization problem can cause non-optimal behavior. Panels (b) of Figures 4.3 and 4.4 show the relative error as a function of the reduced dimension for a fixed mesh size. We see that while the linear maps consistently improve with the reduced dimension, the neural networks struggle as the complexity of the optimization problem is increased. This problem can usually be alleviated with the addition of more data as shown in panels (c), but there are still no guarantees that the optimal neural network is found. Since we use a highly-nonlinear 5-layer network to represent the linear φ, this issue is exacerbated for this problem and the addition of more data only slightly improves the accuracy as seen in panels (c). In Appendix D, we show the relative test error during the training process and observe that some overfitting occurs, indicating that the optimization problem is stuck in a local minima away from the optimal linear solution. This is an issue that is inherent to most deep neural network based methods. Our results suggest that building in a priori information about the solution map, such as linearity, can be very beneficial for the approximation scheme as it can help reduce the complexity of the optimization.

To compare to the method of Chkifa [14], we will assume the following model for the inputs,

$$f = \sum_{j=1}^{\infty} \xi_j \phi_j \tag{4.2}$$

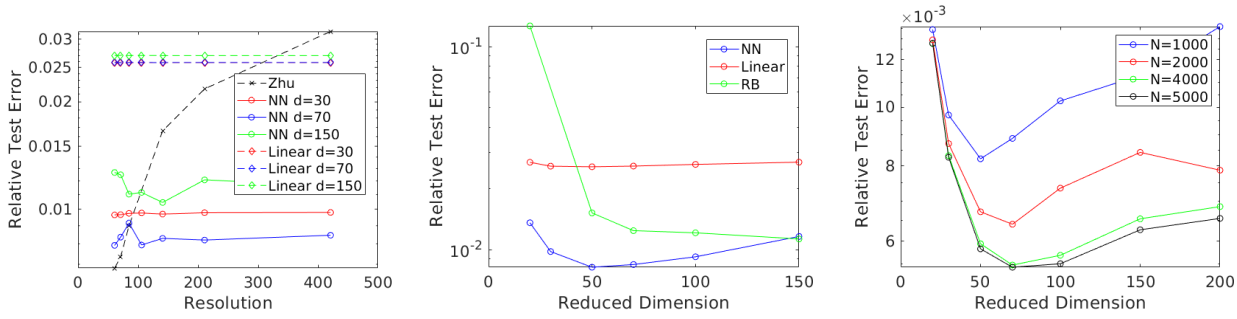
where $\xi_j \sim U(-1,1)$ is an i.i.d. sequence, and $\phi_j = \sqrt{\lambda_j}\psi_j$ where λ_j, ψ_j are the eigenvalues and eigenfunctions of the operator $(-\Delta + 100I)^{-4.1}$ with a zero Neumann boundary. This construction ensures that there exists $p \in (0,1)$ such that $(\|\phi_j\|_{L^\infty})_{j\geq 1} \in \ell^p(\mathbb{N};\mathbb{R})$ which is required for the theory in [18]. We assume this model for f, the r.h.s. of the Poisson equation, and consider the solution operator $\Psi : \ell^\infty(\mathbb{N};\mathbb{R}) \to H_0^1(D;\mathbb{R})$ mapping $(\xi_j)_{j\geq 1} \mapsto u$. Figure 4.5 panels (a)-(b) show an example input from (4.2) and its corresponding solution u. Since this operator is linear, its Taylor series representation simply amounts to

$$\Psi((\xi_j)_{j\geq 1}) = \sum_{j=1}^{\infty} \xi_j \eta_j \tag{4.3}$$

where $\eta_j \in H_0^1(D;\mathbb{R})$ satisfy

$$-\Delta \eta_j = \phi_j.$$

This is easily seen by plugging in our model (4.2) for f into the Poisson equation and formally inverting the Laplacian. We further observe that the $\ell^1(\mathbb{N};\mathbb{R})$ summability of the sequence $(\|\eta_j\|_{H_0^1})_{j\geq 1}$



(a) (b) (c)

FIGURE 4.6. Relative test errors on the Darcy flow problem with log-normal coefficients. Using $N = 1024$ training examples, panel (a) shows the errors as a function of the resolution while panel (b) fixes a 421×421 mesh and shows the error as a function of the reduced dimension. Panel (c) only shows results for our method using a neural network, fixing a 421×421 mesh and showing the error as a function of the reduced dimension for different amounts of training data.

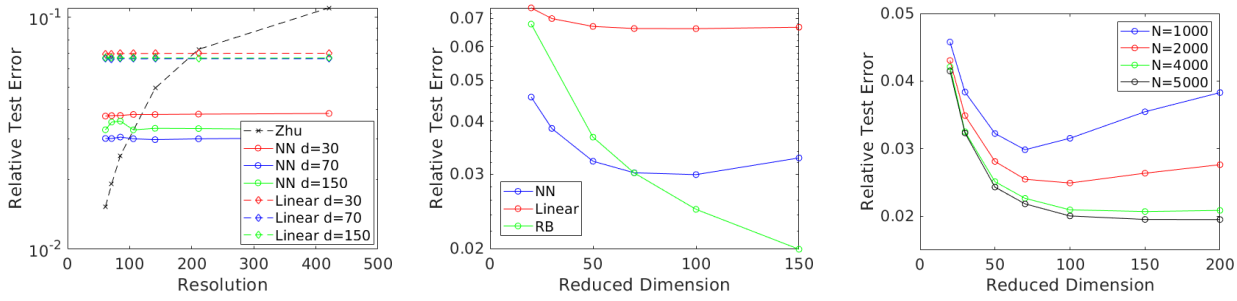
(inherited from $(\|\phi_j\|_{L^\infty})_{j\geq 1} \in \ell^p(\mathbb{N};\mathbb{R})$) implies that our power series (4.3) is summable in $H_0^1(D;\mathbb{R})$. Combining the two observations yields analyticity of Ψ with the same rates as in [18] obtained via Stechkin's inequality. For a proof, see Theorem C.1.

We employ the method of Chkifa simply by truncation of (4.3) to d elements, noting that in this simple linear setting there is no longer a need for greedy selection of the index set. We note that this truncation requires d PDE solves of the Poisson equation hence we compare to our method when using $N = d$ data points, since this also counts the number of PDE solves. Since the problem is linear, we use a linear map to interpolate the PCA latent spaces and furthermore set the reduced dimension of our PCA(s) to N. Panel (c) of Figure 4.5 shows the results. We see that the method of Chkifa outperforms our method for any fixed number of PDE solves, although the empirical rate of convergence appears very similar for both methods. Furthermore we highlight that while our method appears to have a larger error constant than that of Chkifa, it has the advantage that it requires no knowledge of the model 4.2 or of the Poisson equation; it is driven entirely by the training data.

4.3. **Darcy Flow**

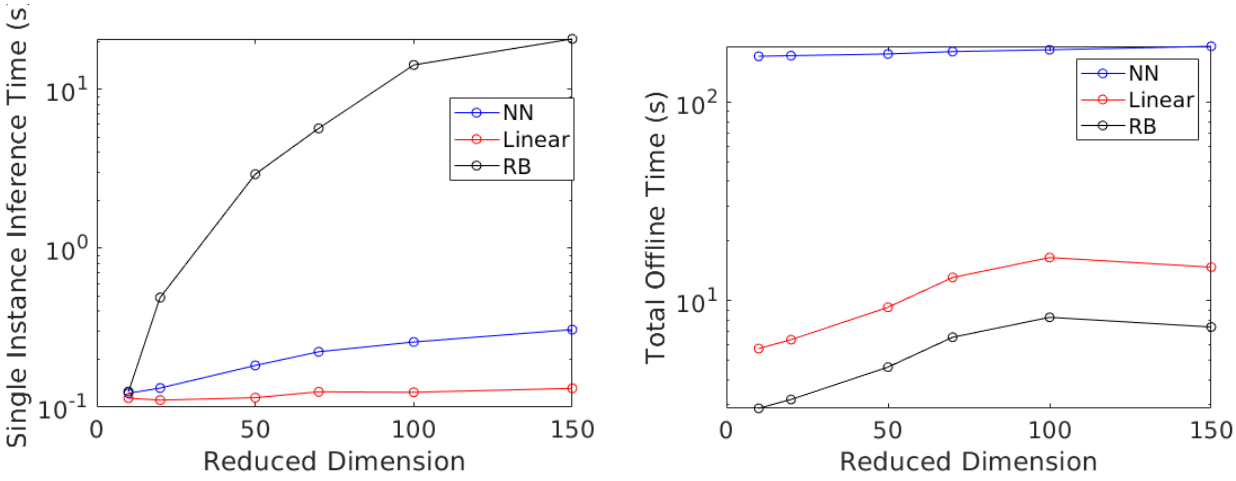
We now consider the input-output map $\Psi : L^\infty(D;\mathbb{R}_+) \to H_0^1(D;\mathbb{R})$ mapping $a \mapsto u$ in (1.1) with $f(s) = 1 \; \forall \, s \in D$ fixed. In this setting, the solution operator is nonlinear and is locally Lipschitz as a mapping from $L^\infty(D;\mathbb{R}_+)$ to $H_0^1(D;\mathbb{R})$ [20]. However our results require a Hilbert space structure, and we view the solution operator as a mapping from $L^2(D;\mathbb{R}_+) \supset L^\infty(D;\mathbb{R}_+)$ into $H_0^1(D;\mathbb{R}_+)$, noting that we will choose the probability measure μ on $L^2(D;\mathbb{R}_+)$ to satisfy $\mu(L^\infty(D;\mathbb{R}_+)) = 1$. In this setting, Ψ is not locally Lipschitz and hence Theorem 3.5 is not directly applicable. Nevertheless, our methodology exhibits competitive numerical performance. See rows 3 and 4 of Figure 4.2 for an example.

Figure 4.6 (a) shows the relative test errors as a function of the resolution when $a \sim \mu = \mu_{\mathrm{L}}$ is log-normal while Figure 4.7 (a) shows them when $a \sim \mu = \mu_{\mathrm{P}}$ is piecewise constant. In both settings, we see that the error in our method is invariant to mesh-refinement. Since the problem is nonlinear, the neural network outperforms the linear map. However we see the same issue as in Figure 4.3 where increasing the reduced dimension does not necessarily improve the error due to the increased complexity of the optimization problem. Panels (b) of Figures 4.6 and 4.7 confirm this observation.



(a) (b) (c)

Figure 4.7. Relative test errors on the Darcy flow problem with piecewise constant coefficients. Using $N = 1024$ training examples, panel (a) shows the errors as a function of the resolution while panel (b) fixes a 421×421 mesh and shows the error as a function of the reduced dimension. Panel (c) only shows results for our method using a neural network, fixing a 421×421 mesh and showing the error as a function of the reduced dimension for different amounts of training data.

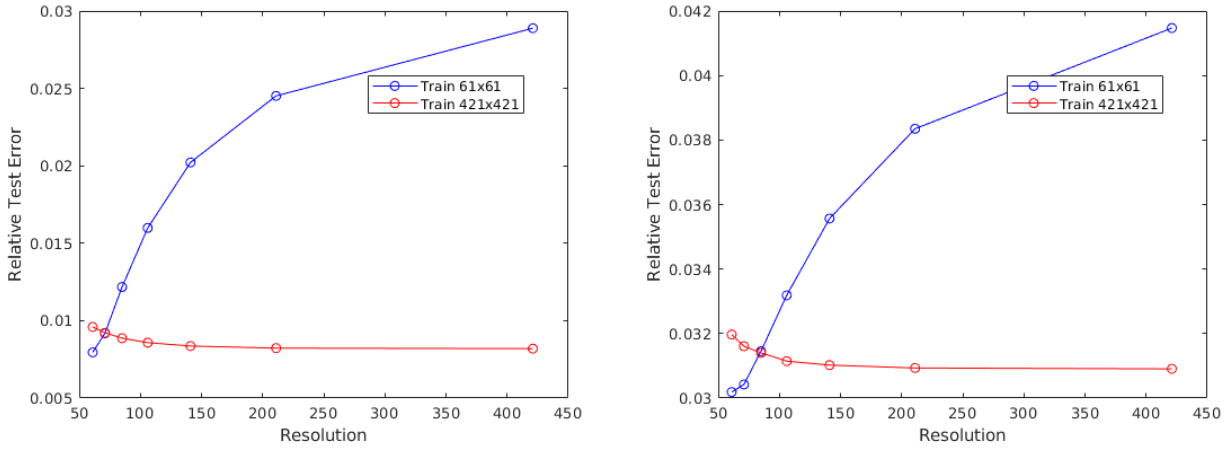


(a) Online (b) Offline

Figure 4.8. The online and offline computation times for the Darcy flow problem with piecewise constant coefficients. The number of training examples $N = 1024$ and grid resolution 421×421 are fixed. The results are reported in seconds and all computations are done on a single GTX 1080 Ti GPU.

This issue can be alleviated with additional training data. Indeed, panels (c) of Figures 4.6 and 4.7 show that the error curve is flattened with more data. We highlight that these results are consistent with our interpretation of Theorem 3.1: the reduced dimensions $d_{\mathcal{X}}, d_{\mathcal{Y}}$ are determined first by the properties of the measure μ and its pushforward, and then the amount of data necessary is obtained to ensure that the finite data approximation error is of the same order of magnitude as the finite-dimensional approximation error. In summary, the size of the training dataset N should increase with the number of reduced dimensions.

For this problem, we also compare to the reduced basis method (RB) when instantiated with PCA. We implement this by a standard Galerkin projection, expanding the solution in the PCA basis and



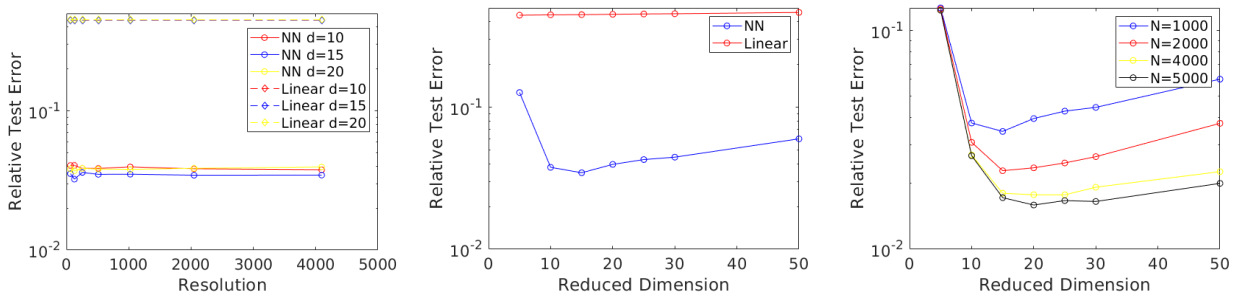
(a) μ_{L} (b) μ_{P}

FIGURE 4.9. Relative test errors on both Darcy flow problems with reduced dimension $d = 70$, training on a single mesh and transferring the solution to other meshes. When the training mesh is smaller than the desired output mesh, the PCA basis are interpolated using cubic splines. When the training mesh is larger than the desired output mesh, the PCA basis are sub-sampled.

using the weak form of (1.1) to find the coefficients. We note that the errors of both methods are very close, but we find that the online runtime of our method is significantly better. Letting K denote the mesh-size and d the reduced dimension, the reduced basis method has a runtime of $\mathcal{O}(d^2K + d^3)$ while our method has the runtime $\mathcal{O}(dK)$ plus the runtime of the neural network which, in practice, we have found to be negligible. We show the online inference time as well as the offline training time of the methods in Figure 4.8. While the neural network has the highest offline cost, its small online cost makes it a more practical method. Indeed, without parallelization when $d = 150$, the total time (online and offline) to compute all 5000 test solutions is around 28 hours for the RBM. On the other hand, for the neural network, it is 28 minutes. The difference is pronounced when needing to compute many solutions in parallel. Since most modern architectures are able to internally parallelize matrix-matrix multiplication, the total time to train and compute the 5000 examples for the neural network is only 4 minutes. This issue can however be slightly alleviated for the reduced basis method with more stringent multi-core parallelization. We note that the linear map has the lowest online cost and only a slightly worse offline cost than the RBM. This makes it the most suitable method for linear operators such as those presented in Section 4.2 or for applications where larger levels of approximation error can be tolerated.

We again note that the image-to-image regression approach of [83] does not scale with the mesh size. We do however acknowledge that for the small meshes for which the method was designed, it does outperform all other approaches. This begs the question of whether one can design neural networks that match the performance of image-to-image regression but remain invariant with respect to the size of the mesh. The contemporaneous work [51] takes a step in this direction.

Lastly, we show that our method also has the ability to transfer a solution learned on one mesh to another. This is done by interpolating or sub-sampling both of the input and output PCA basis from the training mesh to the desired mesh. Justifying this requires a smoothness assumption on the PCA basis; we are, however, not aware of any such results and believe this is an interesting future direction. The neural network is fixed and does not need to be re-trained on a new mesh. We show this



(a) (b) (c)

Figure 4.10. Relative test errors on the Burgers' Equation problem. Using $N = 1024$ training examples, panel (a) shows the errors as a function of the resolution while panel (b) fixes a 4096 mesh and shows the error as a function of the reduced dimension. Panel (c) only shows results for our method using a neural network, fixing a 4096 mesh and showing the error as a function of the reduced dimension for different amounts of training data.

in Figure 4.9 for both Darcy flow problems. We note that when training on a small mesh, the error increases as we move to larger meshes, reflecting the interpolation error of the basis. Nevertheless, this increase is rather small: as shown in Figure 4.9, we obtain a 3% and a 1% relative error increasing when transferring solutions trained on a 61×61 grid to a 421×421 grid on each respective Darcy flow problem. On the other hand, when training on a large mesh, we see almost no error increase on the small meshes. This indicates that the neural network learns a property that is intrinsic to the solution operator and independent of the discretization.

4.4. **Burgers' Equation**

We now consider the input-output map $\Psi : L^2(\mathbb{T}^1; \mathbb{R}) \to H^r(\mathbb{T}^1; \mathbb{R})$ mapping $u_0 \mapsto u|_{t=1}$ in (4.1) with $\beta = 10^{-2}$ fixed. In this setting, Ψ is nonlinear and locally Lipschitz but since we do not know the precise Lipschitz constant as defined in Appendix A, we cannot verify that the assumptions of Theorem A.1 hold; nevertheless the numerical results demonstrate the effectiveness of our methodology. We take $u_0 \sim \mu = \mu_B$; see rows 5 of Figure 4.2 for an example. Figure 4.10 (a) shows the relative test errors as a function of the resolution again demonstrating that our method is invariant to mesh-refinement. We note that, for this problem, the linear map does significantly worse than the neural network in contrast to the Darcy flow problem where the results were comparable. This is likely attributable to the fact that the solution operator for Burgers' equation is more strongly nonlinear. As before, we observe from Figure 4.10 panel (b) that increasing the reduced dimension does not necessarily improve the error due to the increased complexity of the optimization problem. This can again be mitigated by increasing the volume of training data, as indicated in Figure 4.10 (c); the curve of error versus reduced dimension is flattened as N increases.

5. **Conclusion**

In this paper, we proposed a general data-driven methodology that can be used to learn mappings between separable Hilbert spaces. We proved consistency of the approach when instantiated with PCA in the setting of globally Lipschitz forward maps. We demonstrated the desired mesh-independent

properties of our approach on parametric PDE problems, showing good numerical performance even on problems outside the scope of the theory.

This work leaves many interesting directions open for future research. To understand the interplay between the reduced dimension and the amount of data needed requires a deeper understanding of neural networks and their interaction with the optimization algorithms used to produce the approximation architecture. Even if the optimal neural network is found by that optimization procedure, the question of the number of parameters needed to achieve a given level of accuracy, and how this interacts with the choice of reduced dimensions $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ (choice of which is determined by the input space probability measure), warrants analysis in order to reveal the computational complexity of the proposed approach. Furthermore, the use of PCA limits the scope of problems that can be addressed to Hilbert, rather than general Banach spaces; even in Hilbert space, PCA may not be the optimal choice of dimension reduction. The development of autoenconders on function space is a promising direction that has the potential to address these issues; it also has many potential applications that are not limited to deployment within the methodology proposed here. Finally we also wish to study the use of our methodology in more challenging PDE problems, such as those arising in materials science, as well as for time-dependent problems such as multi-phase flow in porous media. Broadly speaking we view our contribution as a first step in the development of methods that generalize the ideas and applications of neural networks by operating on, and between, spaces of functions.

Acknowledgments

The authors are grateful to Anima Anandkumar, Kamyar Azizzadenesheli, Zongyi Li and Nicholas H. Nelsen for helpful discussions in the general area of neural networks for PDE-defined maps between Hilbert spaces. The authors thank Matthew M. Dunlop for sharing his code for solving elliptic PDEs and generating Gaussian random fields.

Appendix A. Neural Networks And Approximation (Locally Lipschitz Case)

This extends the approximation theory of Subsection 3.2 to the case of solution maps $\Psi : \mathcal{X} \to \mathcal{Y}$ that are μ-measurable and locally Lipschitz in the following sense

$$\forall\, x, z \in \mathcal{X} \quad \|\Psi(x) - \Psi(z)\|_{\mathcal{Y}} \le L(x,z)\|x - z\|_{\mathcal{X}}. \tag{A.1}$$

where the function $L : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is symmetric in its arguments, i.e., $L(x,z) = L(z,x)$, and for any fixed $w \in \mathcal{X}$ the function $L(\cdot, w) : \mathcal{X} \to \mathbb{R}_+$ is μ-measurable and non-decreasing in the sense that $L(s,w) \le L(x,w)$ if $\|x\|_{\mathcal{X}} \ge \|w\|_{\mathcal{X}}$. Note that this implies that Ψ is locally bounded: for any $x \in \mathcal{X}$

$$\|\Psi(x)\|_{\mathcal{Y}} \le \|\Psi(0)\|_{\mathcal{Y}} + \|\Psi(x) - \Psi(0)\|_{\mathcal{Y}} \le \|\Psi(0)\|_{\mathcal{Y}} + L(x,0)\|x\|_{\mathcal{X}}.$$

Hence we deduce that the pushforward $\Psi_\sharp\mu$ has bounded fourth moments provided that $\mathbb{E}_{x\sim\mu}(L(x,0)\|x\|_{\mathcal{X}})^4 < +\infty$:

$$\begin{aligned}\mathbb{E}_{y\sim\Psi_\sharp\mu}\|y\|_{\mathcal{Y}}^4 &= \int_{\mathcal{X}} \|\Psi(x)\|_{\mathcal{Y}}^4 d\mu(x) \le \int_{\mathcal{X}} (\|\Psi(0)\|_{\mathcal{Y}} + L(x,0)\|x\|_{\mathcal{X}})^4 d\mu(x) \\ &\le 2^3 \left(\|\Psi(0)\|_{\mathcal{Y}} + \int_{\mathcal{X}} L(x,0)^4 \|x\|_{\mathcal{X}}^4 d\mu(x) \right) < \infty,\end{aligned}$$

where we used a generalized triangle inequality proven in [77, Cor. 3.1].

Theorem A.1. *Let $\mathcal{X}$, $\mathcal{Y}$ be real, separable Hilbert spaces, Ψ a mapping from $\mathcal{X}$ into $\mathcal{Y}$ and let μ be a probability measure supported on $\mathcal{X}$ such that*

$$\mathbb{E}_{x\sim\mu} L(x,x)^2 < +\infty, \quad \mathbb{E}_{x\sim\mu} L(x,0)^2 \|x\|_{\mathcal{X}}^2 < \infty,$$

where $L(\cdot,\cdot)$ is given in (A.1). *Fix $d_{\mathcal{X}}, d_{\mathcal{Y}}$, $N \geq \max\{d_{\mathcal{X}}, d_{\mathcal{Y}}\}$, $\delta \in (0,1)$, $\tau > 0$ and define $M = \sqrt{\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^2/\delta}$. Then there exists a constant $c(d_{\mathcal{X}}, d_{\mathcal{Y}}) \geq 0$ and neural network $\chi \in \mathcal{M}(d_{\mathcal{X}}, d_{\mathcal{Y}}; t, r, M)$ with $t \leq c(\log(\sqrt{d_{\mathcal{Y}}}/\tau)+1)$ layers and $r \leq c(\epsilon^{-d_{\mathcal{X}}}\log(\sqrt{d_{\mathcal{Y}}}/\tau)+1)$ active weights and biases in each component, so that*

$$\mathbb{E}_{\{x_j\}\sim\mu}\mathbb{E}_{x\sim\mu}(e_{NN}(x)) \leq C\left(\tau + \sqrt{\delta} + \left(\sqrt{\frac{d_{\mathcal{X}}}{N}} + R^{\mu}(V_{d_{\mathcal{X}}}^{\mathcal{X}})\right)^{1/2} + \left(\sqrt{\frac{d_{\mathcal{Y}}}{N}} + R^{\Psi_\sharp\mu}(V_{d_{\mathcal{Y}}}^{\mathcal{Y}})\right)^{1/2}\right), \quad \text{(A.2)}$$

where $e_{NN}(x) := \|\Psi_{NN}(x) - \Psi(x)\|_{\mathcal{Y}}$ and $C > 0$ is independent of $d_{\mathcal{X}}, d_{\mathcal{Y}}, N, \delta$ and ϵ.

Proof. Our method of proof is similar to the proof of Theorem 3.5 and for this reason we shorten some of the arguments. Recall the constant Q from Theorem 3.4. In what follows we take Q to be the maximum of the two such constants when arising from application of the theorem on the two different probability spaces $(\mathcal{X}, \mu)$ and $(\mathcal{Y}, \Psi_\sharp\mu)$. As usual we employ the shorthand notation $\mathbb{E}$ to denote $\mathbb{E}_{\{x_j\}\sim\mu}$ the expectation with respect to the dataset $\{x_j\}_{j=1}^N$.
We begin by approximating the error incurred by using Ψ_{PCA} given by (2.3):

$$\begin{aligned}
\mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{PCA}(x) - \Psi(x)\|_{\mathcal{Y}} &= \mathbb{E}\mathbb{E}_{x\sim\mu}\|(G_{\mathcal{Y}}\circ F_{\mathcal{Y}}\circ\Psi\circ G_{\mathcal{X}}\circ F_{\mathcal{X}})(x) - \Psi(x)\|_{\mathcal{Y}} \\
&= \mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x) - \Psi(x)\right\|_{\mathcal{Y}} \\
&\leq \mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x) - \Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(x)\right\|_{\mathcal{Y}} + \mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}\Psi(x) - \Psi(x)\right\|_{\mathcal{Y}} \\
&\leq \mathbb{E}\mathbb{E}_{x\sim\mu}L(\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x, x)\left\|\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x - x\right\|_{\mathcal{X}} + \mathbb{E}\mathbb{E}_{y\sim\Psi_\sharp\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}y - y\right\|_{\mathcal{Y}},
\end{aligned} \quad \text{(A.3)}$$

noting that the operator norm of an orthogonal projection is 1. Now since $L(\cdot, x)$ is non-decreasing we infer that $L(\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x, x) \leq L(x,x)$ and then using Cauchy–Schwarz we obtain

$$\begin{aligned}
\mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{PCA}(x) - \Psi(x)\|_{\mathcal{Y}} &\leq \left(\mathbb{E}_{x\sim\mu}|L(x,x)|^2\right)^{1/2}\left(\mathbb{E}\mathbb{E}_{x\sim\mu}\left\|\Pi_{V_{d_{\mathcal{X}},N}^{\mathcal{X}}}x - x\right\|_{\mathcal{X}}^2\right)^{1/2} \\
&\quad + \mathbb{E}\mathbb{E}_{y\sim\Psi_\sharp\mu}\left\|\Pi_{V_{d_{\mathcal{Y}},N}^{\mathcal{Y}}}y - y\right\|_{\mathcal{Y}}, \\
&= L'\left(\mathbb{E}R^{\mu}(V_{d_{\mathcal{X}},N}^{\mathcal{X}})\right)^{1/2} + \left(\mathbb{E}[R^{\Psi_\sharp\mu}(V_{d_{\mathcal{Y}},N}^{\mathcal{Y}})]\right)^{1/2}
\end{aligned} \quad \text{(A.4)}$$

where we used Hölder's inequality in the last line and defined the new constant $L' := (\mathbb{E}_{x\sim\mu}|L(x,x)|^2)^{1/2}$. Theorem 3.4 allows us to control this error, and leads to

$$\begin{aligned}
\mathbb{E}\mathbb{E}_{x\sim\mu}e_{NN} &\leq \mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{NN}(x) - \Psi_{PCA}(x)\|_{\mathcal{Y}} + \mathbb{E}\mathbb{E}_{x\sim\mu}\|\Psi_{PCA}(x) - \Psi(x)\|_{\mathcal{Y}} \\
&\leq \mathbb{E}_{x\sim\mu}\|\Psi_{NN}(x) - \Psi_{PCA}(x)\|_{\mathcal{Y}} \\
&\quad + L'\left(\sqrt{\frac{Qd_{\mathcal{X}}}{N}} + R^{\mu}(V_{d_{\mathcal{X}}}^{\mathcal{X}})\right)^{1/2} + \left(\sqrt{\frac{Qd_{\mathcal{Y}}}{N}} + R^{\Psi_\sharp\mu}(V_{d_{\mathcal{Y}}}^{\mathcal{Y}})\right)^{1/2}.
\end{aligned} \quad \text{(A.5)}$$

We now approximate φ by a neural network χ as before. To that end we first note from Lemma B.3 that φ is locally Lipschitz, and hence continuous, as a mapping from $\mathbb{R}^{d_{\mathcal{X}}}$ into $\mathbb{R}^{d_{\mathcal{Y}}}$. Identify the components $\varphi(s) = (\varphi^{(1)}(s), \ldots, \varphi^{(d_{\mathcal{Y}})}(s))$ where each function $\varphi^{(j)} \in C(\mathbb{R}^{d_{\mathcal{X}}}; \mathbb{R})$. We consider the restriction of each component function to the set $[-M, M]^{d_{\mathcal{X}}}$.
By [81, Thm. 1] and using the same arguments as in the proof of Theorem 3.5 there exist neural networks $\chi^{(1)}, \ldots, \chi^{(d_{\mathcal{Y}})} : \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}$, $\chi^{(j)} \in \mathcal{M}(d_{\mathcal{X}}; t^{(j)}, r^{(j)})$, with layer and active weight parameters $t^{(j)}$ and $r^{(j)}$ satisfying $t^{(j)} \leq c^{(j)}[\log(M\sqrt{d_{\mathcal{Y}}}/\tau)+1]$, and $r^{(j)} \leq c^{(j)}(\tau/2M)^{-d_{\mathcal{X}}}[\log(M\sqrt{d_{\mathcal{Y}}}/\tau)+1]$

with constants $c^{(j)}(d_{\mathcal{X}}) > 0$, so that

$$\left|(\chi^{(1)}(s), \ldots, \chi^{(d_{\mathcal{Y}})}(s)) - \varphi(s)\right|_2 < \tau \quad \forall\, s \in [-M, M]^{d_{\mathcal{X}}}.$$

We now simply define $\chi : \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}^{d_{\mathcal{Y}}}$ as the stacked network $(\chi^{(1)}, \ldots, \chi^{d_{\mathcal{Y}}})$ extended by zero outside of $[-M, M]^{d_{\mathcal{X}}}$ to immediately obtain

$$\sup_{s \in [-M,M]^{d_{\mathcal{X}}}} \left|\chi(s) - \varphi(s)\right|_2 < \tau. \tag{A.6}$$

Thus, by construction $\chi \in \mathcal{M}(d_{\mathcal{X}}, d_{\mathcal{Y}}, t, r, M)$ with at most $t \leq \max_j t^{(j)}$ many layers and $r \leq r^{(j)}$ many active weights and biases in each of its components.
Define the set $A = \{x \in \mathcal{X} : F_{\mathcal{X}}(x) \in [-M, M]^{d_{\mathcal{X}}}\}$. By Lemma B.4 below, $\mu(A) \geq 1-\delta$ and $\mu(A^c) \leq \delta$. Define the approximation error $e_{PCA}(x) := \|\Psi_{NN}(x) - \Psi_{PCA}(x)\|_{\mathcal{Y}}$ and decompose its expectation as

$$\mathbb{E}_{x\sim\mu}[e_{PCA}(x)] = \int_A e_{PCA}(x) d\mu(x) + \int_{A^c} e_{PCA}(x) d\mu(x) =: I_A + I_{A^c}.$$

For the first term,

$$I_A \leq \int_A \|(G_{\mathcal{Y}} \circ \chi \circ F_{\mathcal{X}})(x) - (G_{\mathcal{Y}} \circ \varphi \circ F_{\mathcal{X}})(x)\|_{\mathcal{Y}} d\mu(x) \leq \tau, \tag{A.7}$$

by using the fact, established in Lemma B.3, that $G_{\mathcal{Y}}$ is Lipschitz with Lipschitz constant 1, the τ-closeness of χ to φ from (A.6), and $\mu(A) \leq 1$. For the second term we have, using that $G_{\mathcal{Y}}$ has Lipschitz constant 1 and that χ takes value zero on A^c,

$$\begin{aligned} I_{A^c} &\leq \int_{A^c} \|(G_{\mathcal{Y}} \circ \chi \circ F_{\mathcal{X}})(x) - (G_{\mathcal{Y}} \circ \varphi \circ F_{\mathcal{X}})(x)\|_{\mathcal{Y}} d\mu(x) \\ &\leq \int_{A^c} |\chi(F_{\mathcal{X}}(x)) - \varphi(F_{\mathcal{X}}(x))|_2 d\mu(x) = \int_{A^c} |\varphi(F_{\mathcal{X}}(x))|_2 d\mu(x). \end{aligned} \tag{A.8}$$

Once more from Lemma B.3 we have that

$$|F_{\mathcal{X}}(x)|_2 \leq \|x\|_{\mathcal{X}}; \quad |\varphi(v)|_2 \leq |\varphi(0)|_2 + L(G_{\mathcal{X}}(v), 0)|v|_2$$

so that, using the hypothesis on L and global Lipschitz property of $F_{\mathcal{X}}$ and $G_{\mathcal{X}}$ we can write

$$\begin{aligned} I_{A^c} &\leq \mu(A^c)|\varphi(0)|_2 + \int_{A^c} L(x,0)|F_{\mathcal{X}}(x)|_2 d\mu(x) \\ &\leq \mu(A^c)|\varphi(0)|_2 + \int_{A^c} L(x,0)\|x\|_{\mathcal{X}} d\mu(x) \\ &\leq \mu(A^c)|\varphi(0)|_2 + \mu(A^c)^{\frac{1}{2}} \left(\mathbb{E}_{x\sim\mu} L(x,0)^2 \|x\|_{\mathcal{X}}^2\right)^{1/2} \\ &\leq \delta|\varphi(0)|_2 + \delta^{\frac{1}{2}} L'' \\ &\leq \sqrt{\delta}(|\varphi(0)|_2 + L'') \end{aligned} \tag{A.9}$$

where $L'' := \left(\mathbb{E}_{x\sim\mu} L(x,0)^2 \|x\|_{\mathcal{X}}^2\right)^{1/2}$. Combining (3.3), (3.5) and (3.7) we obtain the desired result. ∎

Appendix B. Supporting Lemmas

In this Subsection we present and prove auxiliary lemmas that are used throughout the proofs in the article. The proof of Theorem 3.4 made use of the following proposition, known as Fan's Theorem, proved originally in [27]. We state and prove it here in the infinite-dimensional setting as this generalization may be of independent interest. Our proof follows the steps of Fan's original proof in the finite-dimensional setting. The work [62], through which we first became aware of Fan's result, gives an elegant generalization; however it is unclear whether that approach is easily applicable in infinite dimensions due to issues of compactness.

Lemma B.1 (Fan [27]). *Let $(\mathcal{H}, \langle\cdot,\cdot\rangle, \|\cdot\|)$ be a separable Hilbert space and $C : \mathcal{H} \to \mathcal{H}$ a non-negative, self-adjoint, compact operator. Denote by $\lambda_1 \geq \lambda_2 \geq \dots$ the eigenvalues of C and, for any $d \in \mathbb{N} \setminus \{0\}$, let S_d denote the set of collections of d orthonormal elements of $\mathcal{H}$. Then*

$$\sum_{j=1}^{d} \lambda_j = \max_{\{u_1,\dots,u_d\}\in S_d} \sum_{j=1}^{d} \langle Cu_j, u_j\rangle.$$

Proof. Let $\phi_1, \phi_2, \dots$ denote the orthonormal eigenfunctions of C corresponding to the eigenvalues $\lambda_1, \lambda_2, \dots$ respectively. Note that for $\{\phi_1, \dots, \phi_d\} \in S_d$, we have

$$\sum_{j=1}^{d} \langle C\phi_j, \phi_j\rangle = \sum_{j=1}^{d} \lambda_j \|\phi_j\|^2 = \sum_{j=1}^{d} \lambda_j.$$

Now let $\{u_1, \dots, u_d\} \in S_d$ be arbitrary. Then for any $j \in \{1, \dots, d\}$, we have $u_j = \sum_{k=1}^{\infty} \langle u_j, \phi_k\rangle \phi_k$ and thus

$$\begin{aligned}\langle Cu_j, u_j\rangle &= \sum_{k=1}^{\infty} \lambda_k |\langle u_j, \phi_k\rangle|^2 \\ &= \lambda_d \sum_{k=1}^{\infty} |\langle u_j, \phi_k\rangle|^2 + \sum_{k=d+1}^{\infty} (\lambda_k - \lambda_d)|\langle u_j, \phi_k\rangle|^2 + \sum_{k=1}^{d} (\lambda_k - \lambda_d)|\langle u_j, \phi_k\rangle|^2.\end{aligned}$$

Since $\|u_j\|^2 = 1$, we have $\|u_j\|^2 = \sum_{k=1}^{\infty} |\langle u_j, \phi_k\rangle|^2 = 1$ therefore

$$\begin{aligned}\lambda_d \sum_{k=1}^{\infty} |\langle u_j, \phi_k\rangle|^2 + \sum_{k=d+1}^{\infty} (\lambda_k - \lambda_d)|\langle u_j, \phi_k\rangle|^2 &= \lambda_d \sum_{k=1}^{d} |\langle u_j, \phi_k\rangle|^2 + \sum_{k=d+1}^{\infty} \lambda_k |\langle u_j, \phi_k\rangle|^2 \\ &\leq \lambda_d \sum_{k=1}^{\infty} |\langle u_j, \phi_k\rangle|^2 = \lambda_d\end{aligned}$$

using the fact that $\lambda_k \leq \lambda_d$, $\forall\, k > d$. We have shown $\langle Cu_j, u_j\rangle \leq \lambda_d + \sum_{k=1}^{d} (\lambda_k - \lambda_d)|\langle u_j, \phi_k\rangle|^2$. Thus

$$\begin{aligned}\sum_{j=1}^{d} (\lambda_j - \langle Cu_j, u_j\rangle) &\geq \sum_{j=1}^{d} \left(\lambda_j - \lambda_d - \sum_{k=1}^{d} (\lambda_k - \lambda_d)|\langle u_j, \phi_k\rangle|^2\right) \\ &= \sum_{j=1}^{d} (\lambda_j - \lambda_d)\left(1 - \sum_{k=1}^{d} |\langle u_k, \phi_j\rangle|^2\right).\end{aligned}$$

We now extend the finite set of $\{u_k\}_{k=1}^{d}$ from a $d-$dimensional orthonormal set to an orthonormal basis $\{u_k\}_{k=1}^{\infty}$ for $\mathcal{H}$. Note that $\lambda_j \geq \lambda_d$, $\forall\, j \leq d$ and that

$$\sum_{k=1}^{d} |\langle u_k, \phi_j\rangle|^2 \leq \sum_{k=1}^{\infty} |\langle u_k, \phi_j\rangle|^2 = \|\phi_j\|^2 = 1$$

therefore $\sum_{j=1}^{d} (\lambda_j - \langle Cu_j, u_j\rangle) \geq 0$ concluding the proof. ∎

Theorem 3.4 relies on a Monte Carlo estimate of the Hilbert–Schmidt distance between C and C_N that we state and prove below.

Lemma B.2. *Let C be given by* (2.12) *and C_N by* (2.6) *then there exists a constant $Q \geq 0$, depending only on ν, such that*

$$\mathbb{E}_{\{u_j\}\sim\nu} \|C_N - C\|_{HS}^2 = \frac{Q}{N}.$$

Proof. Define $C^{(u_j)} := u_j \otimes u_j$ for any $j \in \{1, \dots, N\}$ and $C^{(u)} := u \otimes u$ for any $u \in \mathcal{H}$, noting that $\mathbb{E}_{u\sim\nu}[C^{(u)}] = C = \mathbb{E}_{u_j\sim\nu}[C^{(u_j)}]$. Further we note that

$$\mathbb{E}_{u\sim\nu}\|C^{(u)}\|_{HS}^2 = \mathbb{E}_{u\sim\nu}\|u\|^4 < \infty$$

and, by Jensen's inequality, $\|C\|_{HS}^2 \le \mathbb{E}_{u\sim\nu}\|C^{(u)}\|_{HS}^2 < \infty$. Once again using the shorthand notation $\mathbb{E}$ in place of $\mathbb{E}_{\{u_j\}\sim\nu}$ we compute,

$$\begin{aligned}
\mathbb{E}\|C_N - C\|_{HS}^2 &= \mathbb{E}\|\frac{1}{N}\sum_{j=1}^N C^{(u_j)} - C\|_{HS}^2 = \mathbb{E}\|\frac{1}{N}\sum_{j=1}^N C^{(u_j)}\|_{HS}^2 - \|C\|_{HS}^2 \\
&= \frac{1}{N}\mathbb{E}_{u\sim\nu}\|C^{(u)}\|_{HS}^2 + \frac{1}{N^2}\sum_{j=1}^N\sum_{k\neq j}^N \langle \mathbb{E}[C^{(u_j)}], \mathbb{E}[C^{(u_k)}]\rangle_{HS} - \|C\|_{HS}^2 \\
&= \frac{1}{N}\mathbb{E}_{u\sim\nu}\|C^{(u)}\|_{HS}^2 + \frac{N^2-N}{N^2}\|C\|_{HS}^2 - \|C\|_{HS}^2 \\
&= \frac{1}{N}(\mathbb{E}_{u\sim\nu}\|C^{(u)}\|_{HS}^2 - \|C\|_{HS}^2) = \frac{1}{N}\mathbb{E}_{u\sim\nu}\|C^{(u)} - C\|_{HS}^2.
\end{aligned}$$

Setting $Q = \mathbb{E}_{u\sim\nu}\|C^{(u)} - C\|_{HS}^2$ completes the proof. ∎

The following lemma, used in the proof of Theorem 3.5, estimates Lipschitz constants of various maps required in the proof.

Lemma B.3. *The maps $F_\mathcal{X}$, $F_\mathcal{Y}$, $G_\mathcal{X}$ and $G_\mathcal{Y}$ are globally Lipschitz:*

$$\begin{aligned}
|F_\mathcal{X}(v) - F_\mathcal{X}(z)|_2 &\le \|v - z\|_\mathcal{X}, \quad \forall\, v, z \in \mathcal{X} \\
|F_\mathcal{Y}(v) - F_\mathcal{Y}(v)|_2 &\le \|v - z\|_\mathcal{Y}, \quad \forall\, v, z \in \mathcal{Y} \\
\|G_\mathcal{X}(v) - G_\mathcal{X}(z)\|_\mathcal{X} &\le |v - z|_2, \quad \forall\, v, z \in \mathbb{R}^{d_\mathcal{X}} \\
\|G_\mathcal{Y}(v) - G_\mathcal{Y}(z)\|_\mathcal{X} &\le |v - z|_2, \quad \forall\, v, z \in \mathbb{R}^{d_\mathcal{Y}}.
\end{aligned}$$

Furthermore, if Ψ is locally Lipschitz and satisfies

$$\forall\, x, w \in \mathcal{X} \quad \|\Psi(x) - \Psi(w)\|_\mathcal{Y} \le L(x, w)\|x - w\|_\mathcal{X},$$

with $L : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ that is symmetric with respect to its arguments, and increasing in the sense that $L(s, w) \le L(x, w)$, if $\|x\|_\mathcal{X} \ge \|s\|_\mathcal{X}$. Then φ is also locally Lipschitz and

$$|\varphi(v) - \varphi(z)|_2 \le L(G_\mathcal{X}(v), G_\mathcal{X}(z))|v - z|_2, \quad \forall\, v, z \in \mathbb{R}^{d_\mathcal{X}}.$$

Proof. We establish that $F_\mathcal{Y}$ and $G_\mathcal{X}$ are Lipschitz and estimate the Lipschitz constants; the proofs for $F_\mathcal{X}$ and $G_\mathcal{Y}$ are similar. Let $\phi_{1,N}^\mathcal{Y}, \dots, \phi_{d_\mathcal{Y},N}^\mathcal{Y}$ denote the eigenvectors of the empirical covariance with respect to the data $\{y_j\}_{j=1}^N$ which span $V_{d_\mathcal{Y},N}^\mathcal{Y}$ and let $\phi_{d_\mathcal{Y}+1,N}^\mathcal{Y}, \phi_{d_\mathcal{Y}+2,N}^\mathcal{Y}, \dots$ be an orthonormal extension to $\mathcal{Y}$. Then, by Parseval's identity,

$$|F_\mathcal{Y}(v) - F_\mathcal{Y}(z)|_2^2 = \sum_{j=1}^{d_\mathcal{Y}} \langle v - z, \phi_{j,N}^\mathcal{Y}\rangle_\mathcal{Y}^2 \le \sum_{j=1}^{\infty} \langle v - z, \phi_{j,N}^\mathcal{Y}\rangle_\mathcal{Y}^2 = \|v - z\|_\mathcal{Y}^2.$$

A similar calculation for $G_\mathcal{X}$, using $\phi_{1,N}^\mathcal{X}, \dots, \phi_{d_\mathcal{X},N}^\mathcal{X}$ the eigenvectors of the empirical covariance of the data $\{x_j\}_{j=1}^N$ yields

$$\|G_\mathcal{X}(v) - G_\mathcal{X}(z)\|_\mathcal{X}^2 = \left\|\sum_{j=1}^{d_\mathcal{X}} (v_j - z_j)\phi_{j,N}^\mathcal{X}\right\|_\mathcal{X}^2 = \sum_{j=1}^{d_\mathcal{X}} |v_j - z_j|^2 = |v - z|_2^2$$

for any $v, z \in \mathbb{R}^{d_{\mathcal{X}}}$, using the fact that the empirical eigenvectors can be extended to an orthonormal basis for $\mathcal{X}$. Recalling that $\varphi = F_{\mathcal{Y}} \circ \Psi \circ G_{\mathcal{X}}$, the above estimates immediately yield

$$|\varphi(v) - \varphi(z)|_2 \leq L(G_{\mathcal{X}}(v), G_{\mathcal{X}}(z))|v - z|_2, \quad \forall\, v, z \in \mathbb{R}^{d_{\mathcal{X}}}.$$

■

The following lemma establishes a bound on the size of the set A that was defined in the proof of Theorems 3.5 and A.1.

Lemma B.4. *Fix $0 < \delta < 1$, let $x \sim \mu$ be a random variable and set $M = \sqrt{\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^2/\delta}$. Define $F_{\mathcal{X}}$ using the random dataset $\{x_j\}_{j=1}^N \sim \mu$ then,*

$$\mathbb{P}\left(F_{\mathcal{X}}(x) \notin [-M, M]^{d_{\mathcal{X}}}\right) \leq \delta,$$

where the probability is computed with respect to both x and the x_j's.

Proof. Denote by $\phi_{1,N}^{\mathcal{X}}, \dots, \phi_{d_{\mathcal{X}},N}^{\mathcal{X}}$ the orthonormal set used to define $V_{d_{\mathcal{X}},N}^{\mathcal{X}}$ (2.7) and let $\phi_{d_{\mathcal{X}}+1,N}^{\mathcal{X}}$, $\phi_{d_{\mathcal{X}}+2,N}^{\mathcal{X}}, \dots$ be an orthonormal extension of this basis to $\mathcal{X}$. For any $j \in \{1, \dots, d_{\mathcal{X}}\}$, by Chebyshev's inequality, we have

$$\mathbb{P}_{x\sim\mu}(|\langle x, \phi_{j,N}^{\mathcal{X}}\rangle_{\mathcal{X}}| \geq M) \leq \frac{\mathbb{E}_{x\sim\mu}|\langle x, \phi_{j,N}^{\mathcal{X}}\rangle_{\mathcal{X}}|^2}{M^2}.$$

Note that the expectation is taken only with respect to the randomness in x and not $\phi_{1,N}^{\mathcal{X}}, \dots, \phi_{d_{\mathcal{X}},N}^{\mathcal{X}}$, so the right hand side is itself a random variable. We further compute

$$\begin{aligned}
&\mathbb{P}_{x\sim\mu}(|\langle x, \phi_{1,N}^{\mathcal{X}}\rangle_{\mathcal{X}}| \geq M, \dots, |\langle x, \phi_{d_{\mathcal{X}},N}^{\mathcal{X}}\rangle_{\mathcal{X}}| \geq M) \\
&\quad \leq \frac{1}{M^2}\mathbb{E}_{x\sim\mu}\sum_{j=1}^{d_{\mathcal{X}}} |\langle x, \phi_{j,N}^{\mathcal{X}}\rangle_{\mathcal{X}}|^2 \leq \frac{1}{M^2}\mathbb{E}_{x\sim\mu}\sum_{j=1}^{\infty} |\langle x, \phi_{j,N}^{\mathcal{X}}\rangle_{\mathcal{X}}|^2 = \frac{1}{M^2}\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^2
\end{aligned}$$

noting that $\|x\|_{\mathcal{X}}^2 = \sum_{j=1}^{\infty} |\langle x, \xi_j\rangle_{\mathcal{X}}|^2$ for any orthonormal basis $\{\xi_j\}_{j=1}^{\infty}$ of $\mathcal{X}$ hence the randomness in $\phi_{1,N}^{\mathcal{X}}, \phi_{2,N}^{\mathcal{X}}, \dots$ is inconsequential. Thus we find that, with $\mathbb{P}$ denoting probability with respect to both $x \sim \mu$ and the random data used to define $F_{\mathcal{X}}$,

$$\mathbb{P}(|\langle x, \phi_{1,N}^{\mathcal{X}}\rangle_{\mathcal{X}}| \leq M, \dots, |\langle x, \phi_{d_{\mathcal{X}},N}^{\mathcal{X}}\rangle_{\mathcal{X}}| \leq M) \geq 1 - \frac{1}{M^2}\mathbb{E}_{x\sim\mu}\|x\|_{\mathcal{X}}^2, = 1 - \delta$$

the desired result. ■

Appendix C. Analyticity of the Poisson Solution Operator

Define $\mathcal{X} = \{\xi \in \ell^\infty(\mathbb{N}; \mathbb{R}) : \|\xi\|_{\ell^\infty} \leq 1\}$ and let $\{\phi_j\}_{j=1}^{\infty}$ be some sequence of functions with the property that $(\|\phi_j\|_{L^\infty})_{j\geq 1} \in \ell^p(\mathbb{N}; \mathbb{R})$ for some $p \in (0, 1)$. Define $\Psi : \mathcal{X} \to H_0^1(D; \mathbb{R})$ as mapping a set of coefficients $\xi = (\xi_1, \xi_2, \dots) \in \mathcal{X}$ to $u \in H_0^1(D; \mathbb{R})$ the unique weak solution of

$$-\Delta u = \sum_{j=1}^{\infty} \xi_j \phi_j \quad \text{in} \quad D, \qquad u|_{\partial D} = 0.$$

Note that since D is a bounded domain and $\xi \in \mathcal{X}$, we have that $\sum_{j=1}^{\infty} \xi_j \phi_j \in L^2(\Omega; \mathbb{R})$ since our assumption implies $(\|\phi_j\|_{L^\infty})_{j\geq 1} \in \ell^1(\mathbb{N}; \mathbb{R})$. Therefore u is indeed the unique weak-solution of the Poisson equation [26, Chap. 6] and Ψ is well-defined.

Theorem C.1. *Suppose that there exists $p \in (0,1)$ such that $(\|\phi_j\|_{L^\infty})_{j\geq 1} \in \ell^p(\mathbb{N};\mathbb{R})$. Then*

$$\lim_{K\to\infty} \sup_{\xi\in\mathcal{X}} \|\Psi(\xi) - \sum_{j=1}^{K} \xi_j\eta_j\|_{H_0^1} = 0$$

where, for each $j \in \mathbb{N}$, $\eta_j \in H_0^1(D;\mathbb{R})$ satisfies

$$-\Delta\eta_j = \phi_j \quad in \quad D, \qquad u|_{\partial D} = 0.$$

Proof. By linearity and Poincaré inequality, we obtain the Lipschitz estimate

$$\|\Psi(\xi^{(1)}) - \Psi(\xi^{(2)})\|_{H_0^1} \leq C\|\sum_{j=1}^{\infty}(\xi_j^{(1)} - \xi_j^{(2)})\phi_j\|_{L^2}, \quad \forall\, \xi^{(1)}, \xi^{(2)} \in \mathcal{X}$$

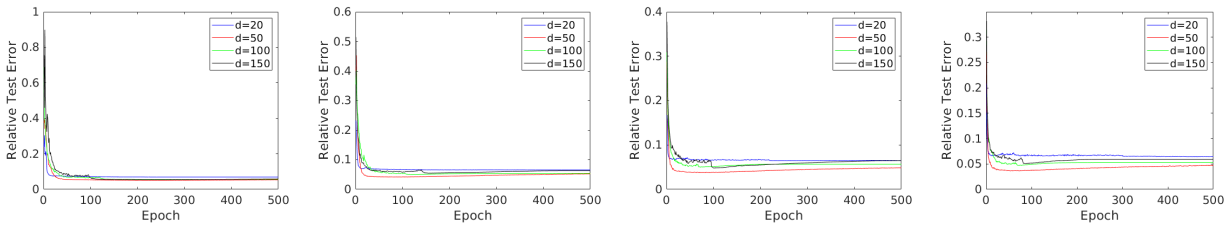
for some $C > 0$. Now let $\xi = (\xi_1, \xi_2, \dots) \in \mathcal{X}$ be arbitrary and define the sequence $\xi^{(1)} = (\xi_1, 0, 0, \dots)$, $\xi^{(2)} = (\xi_1, \xi_2, 0, \dots), \dots$. Note that, by linearity, for any $K \in \mathbb{N}$, $\Psi(\xi^{(K)}) = \sum_{j=1}^{K} \xi_j\eta_j$. Then, using our Lipschitz estimate,

$$\|\Psi(\xi) - \Psi(\xi^K)\|_{H_0^1} \lesssim \|\sum_{j=K+1}^{\infty} \xi_j\phi_j\|_{L^2} \lesssim \sum_{j=K+1}^{\infty} \|\phi_j\|_{L^\infty} \leq K^{1-\frac{1}{p}}\|(\|\phi_j\|_{L^\infty})_{j\geq 1}\|_{\ell^p}$$

where the last line follows by Stechkin's inequality [18, Sec. 3.3]. Taking the supremum over $\xi \in \mathcal{X}$ and the limit $K \to \infty$ completes the proof. ∎

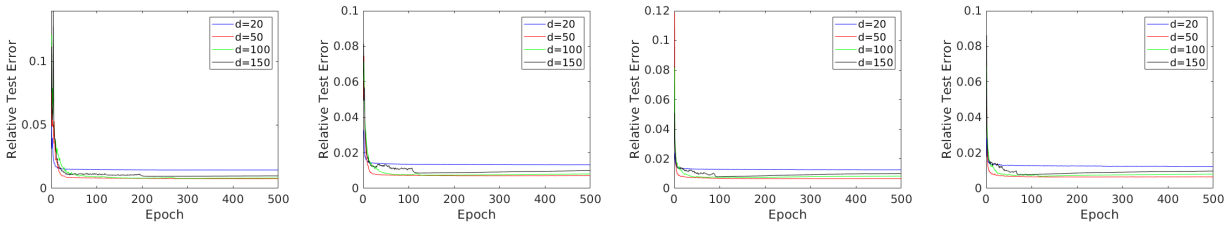
Appendix D. Error During Training

Figures D.1 and D.2 show the relative test error computed during the training process for the problems presented in Section 4.2. For both problems, we observe a slight amount of overfitting when more training samples are used and the reduced dimension is sufficiently large. This is because the true map of interest φ is linear while the neural network parameterization is highly non-linear hence more prone to overfitting larger amounts of data. While this suggests that simpler neural networks might perform better on this problem, we do not carry out such experiments as our goal is simply to show that building in a priori information about the problem (here linearity) can be beneficial as show in Figures 4.3 and 4.4.



(a) $N = 1000$ (b) $N = 2000$ (c) $N = 4000$ (d) $N = 5000$

Figure D.1. Relative test errors as a function of the training epoch on the linear elliptic problem. The amount of training examples used is varied in each panel.



(a) $N = 1000$ (b) $N = 2000$ (c) $N = 4000$ (d) $N = 5000$

Figure D.2. Relative test errors as a function of the training epoch on the Poisson problem. The amount of training examples used is varied in each panel.

References

[1] J. Adler and O. Oktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Probl.*, 33(12), 2017.

[2] B. O. Almroth, P. Stern, and F. A. Brogan. Automatic choice of global shape functions in structural analysis. *AIAA J.*, 16(5):525–528, 1978.

[3] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *C. R. Math. Acad. Sci. Paris*, 339(9):667–672, 2004.

[4] P. Baxendale. Gaussian Measures on Function Spaces. *Am. J. Math.*, 98(4):891–952, 1976.

[5] M. Belkin and P. Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396, 2003.

[6] P. Benner, P. Goyal, B. Kramer, B. Peherstorfer, and K. Willcox. Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. *Comput. Methods Appl. Mech. Eng.*, 372:113433, 2020.

[7] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput. Mech.*, pages 1–21, 2019.

[8] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Data assimilation in reduced modeling. *SIAM/ASA J. Uncertain. Quantif.*, 5(1):1–29, 2017.

[9] G. Blanchard, O. Bousquet, and L. Zwald. Statistical properties of kernel principal component analysis. *Machine Learning*, 66(2):259–294, 2007.

[10] S. Boyaval, C. Le Bris, T. Lelievre, Y. Maday, N. C. Nguyen, and A. T. Patera. Reduced basis techniques for stochastic problems. *Arch. Comput. Methods Eng.*, 17(4):435–454, 2010.

[11] S. Cai, Z. Wang, L. Lu, T. A Zaki, and G. E. Karniadakis. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. `https://arxiv.org/abs/2009.12935`, 2020.

[12] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[13] L. Cheng, N. Kovachki, M. Welborn, and T. F. Miller. Regression Clustering for Improved Accuracy and Training Costs with Molecular-Orbital-Based Machine Learning. *Journal of Chemical Theory and Computation*, 15(12):6668–6677, 2019.

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

[15] A. Cohen, W. Dahmen, and R. DeVore. State Estimation–The Role of Reduced Models. https://arxiv.org/abs/2002.00220, 2020.

[16] A. Cohen and R. DeVore. Approximation of high-dimensional parametric PDEs. *Acta Numer.*, 24:1–159, 2015.

[17] A. Cohen, R. DeVore, and C. Schwab. Convergence Rates of Best N-term Galerkin Approximations for a Class of Elliptic SPDEs. *Found. Comput. Math.*, 10(6):615–646, 2010.

[18] A. Cohen, R. DeVore, and C. Scwhab. Analytic regularity and polynomial approximation of parametric and stochastic elliptic PDEs. *Anal. Appl., Singap.*, 09(01):11–47, 2011.

[19] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005.

[20] M. Dashti, S. Harris, and A. M. Stuart. Besov priors for Bayesian inverse problems. *Inverse Probl. Imaging*, 6:183–200, 2012.

[21] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, and G. Petrova. Nonlinear Approximation and (Deep) ReLU Networks. https://arxiv.org/abs/1905.02199, 2019.

[22] R. DeVore. Nonlinear approximation. *Acta Numer.*, 7:51–150, 1998.

[23] R. DeVore. The Theoretical Foundation of Reduced Basis Methods. In *Model Reduction and Approximation*. Society for Industrial and Applied Mathematics, 2014.

[24] T. Dockhorn. A Discussion on Solving Partial Differential Equations using Neural Networks. https://arxiv.org/abs/1904.07200, 2019.

[25] W. E and B. Yu. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Communications in Mathematics and Statistics*, 2018.

[26] L. C. Evans. *Partial differential equations*. American Mathematical Society, 2010.

[27] K. Fan. On a Theorem of Weyl Concerning Eigenvalues of Linear Transformations I. *Proceedings of the National Academy of Sciences*, 35(11):652–655, 1949.

[28] S. Fresca, L. Dede, and A. Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. https://arxiv.org/abs/2001.04001, 2020.

[29] M. Geist, P. Petersen, M. Raslan, R. Schneider, and G. Kutyniok. Numerical solution of the parametric diffusion equation by deep neural networks. https://arxiv.org/abs/2004.12131, 2020.

[30] J. Gilmer, S. S Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. http://proceedings.mlr.press/v70/gilmer17a.html.

[31] F. J. Gonzalez and M. Balajewicz. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. https://arxiv.org/abs/1808.01346, 2018.

[32] R. Gonzalez-Garcia, R. Rico-Martínez, and I. G. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engineering*, 22:S965–S968, 1998.

[33] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[34] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse Probl.*, 34(1):014004, 2017.

[35] L. Herrmann, C. Schwab, and J. Zech. Deep ReLU Neural Network Expression Rates for Data-to-QoI Maps in Bayesian PDE Inversion. https://www.sam.math.ethz.ch/sam_reports/reports_final/reports2020/2020-02.pdf, 2020.

[36] J. S. Hesthaven, G. Rozza, B. Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*. SpringerBriefs in Mathematics. Springer, 2016.

[37] J. S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.*, 363:55–78, 2018.

[38] G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

[39] J. R Holland, J. D Baeder, and K. Duraisamy. Field Inversion and Machine Learning With Embedded Neural Networks: Physics-Consistent Neural Network Training. In *AIAA Aviation 2019 Forum*, page 3200, 2019.

[40] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning Neural PDE Solvers with Convergence Guarantees. In *International Conference on Learning Representations*, 2019. `https://openreview.net/forum?id=rklawn0qk7`.

[41] M. A. Iglesias, K. Lin, and A. M. Stuart. Well-posed Bayesian geometric inverse problems arising in subsurface flow. *Inverse Probl.*, 30:114001, 2014.

[42] Y. Khoo, J. Lu, and L. Ying. Solving parametric PDE problems with artificial neural networks. `https://arxiv.org/abs/1707.03351`, 2017.

[43] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-Normalizing Neural Networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 971–980. Curran Associates, 2017.

[44] K. Krischer, R. Rico-Martínez, I. G. Kevrekidis, H. H. Rotermund, G. Ertl, and J. L. Hudson. Model identification of a spatiotemporally varying catalytic reaction. *AIChE J.*, 39(1):89–98, 1993.

[45] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric PDEs. `https://arxiv.org/abs/1904.00377`, 2019.

[46] F. Laakmann and P. Petersen. Efficient Approximation of Solutions of Parametric Linear Transport Equations by ReLU DNNs. `https://arxiv.org/abs/2001.11441`, 2020.

[47] I. E Lagaris, A. Likas, and D. I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[48] K. Law, A. Stuart, and K. Zygalakis. *Data Assimilation: A Mathematical Introduction*, volume 62 of *Texts in Applied Mathematics*. Springer, 2015.

[49] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[50] K. Lee and K. T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.*, 404, 2020.

[51] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural Operator: Graph Kernel Networkfor Partial Differential Equations. `https://arxiv.org/abs/2003.03485`, 2020.

[52] C. Lin, Z. Li, L. Lu, S. Cai, M. Maxey, and G. E. Karniadakis. Operator learning for predicting multiscale bubble growth dynamics. `https://arxiv.org/abs/2012.12816`, 2020.

[53] G. J. Lord, C. E. Powell, and T. Shardlow. *An introduction to computational stochastic PDEs*, volume 50. Cambridge University Press, 2014.

[54] L. Lu, P. Jin, and G. E. Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. `https://arxiv.org/abs/1910.03193`, 2019.

[55] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 2020.

[56] Y. Maday, A. T. Patera, J. D. Penn, and M. Yano. A parameterized-background data-weak approach to variational data assimilation: formulation, analysis, and application to acoustics. *Int. J. Numer. Meth. Engng.*, 102(5):933–965, 2015.

[57] V. Maiorov and A. Pinkus. Lower Bounds for Approximation by MLP Neural Networks. *Neurocomputing*, 25:81–91, 1999.

[58] Z. Mao, L. Lu, O. Marxen, T. A Zaki, and G. E. Karniadakis. DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. `https://arxiv.org/abs/2011.03349`, 2020.

[59] S. A. McQuarrie, C. Huang, and K. Willcox. Data-driven reduced-order models via regularized operator inference for a single-injector combustion process. `https://arxiv.org/abs/2008.02862`, 2020.

[60] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[61] D. A. Nagy. Modal representation of geometrically nonlinear behavior by the finite element method. *Computers & Structures*, 10(4):683–688, 1979.

[62] M. L. Overton and R. S. Womersley. On the Sum of the Largest Eigenvalues of a Symmetric Matrix. *SIAM Journal of Matrix Analysis and Applications*, 13(1):41–45, 1992.

[63] K. Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[64] B. Peherstorfer. Sampling low-dimensional Markovian dynamics for pre-asymptotically recovering reduced models from data with operator inference. `https://arxiv.org/abs/1908.11233`, 2019.

[65] B. Peherstorfer and K. Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Comput. Methods Appl. Mech. Eng.*, 306:196–215, 2016.

[66] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox. Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.

[67] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: an introduction*. Springer, 2015.

[68] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[69] S. Reich and C. Cotter. *Probabilistic forecasting and Bayesian data assimilation*. Cambridge University Press, 2015.

[70] L. Ruthotto and E. Haber. Deep Neural Networks Motivated by Partial Differential Equations. *J. Math. Imaging Vis.*, 62:352–364, 2019.

[71] B. Schölkopf, A. Smola, and K. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

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

[73] J. Shawe-Taylor, C. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the Gram matrix and its relationship to the operator eigenspectrum. In *International Conference on Algorithmic Learning Theory*, pages 23–40. Springer, 2002.

[74] J. Shawe-Taylor, C. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the Gram matrix and the generalization error of kernel-PCA. *IEEE Transactions on Information Theory*, 51(7):2510–2522, 2005.

[75] Y. Shin, J. Darbon, and G. E. Karniadakis. On the convergence and generalization of physics informed neural networks. `https://arxiv.org/abs/2004.01806`, 2020.

[76] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross. EikoNet: Solving the Eikonal equation with Deep Neural Networks. `https://arxiv.org/abs/2004.00361`, 2020.

[77] S.-E. Takahasi, J. M. Rassias, S. Saitoh, and Y. Takahashi. Refined generalizations of the triangle inequality on Banach spaces. *Math. Inequal. Appl.*, 13(4):733–741, 2010.

[78] R. Temam. *Infinite-dimensional dynamical systems in mechanics and physics*, volume 68. Springer, 2012.

[79] Q. Wang, J. S. Hesthaven, and D. Ray. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *J. Comput. Phys.*, 384:289–307, 2019.

[80] E. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

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

[82] E. Zeidler. *Applied Functional Analysis: Applications to Mathematical Physics*. Springer, 2012.

[83] Y. Zhu and N. Zabaras. Bayesian Deep Convolutional Encoder-Decoder Networks for Surrogate Modeling and Uncertainty Quantification. *J. Comput. Phys.*, 366(C):415–447, 2018.