Proceedings of Machine Learning Research vol 107:1–21, 2021 2nd Annual Conference on Mathematical and Scientific Machine Learning

Reduced Order Modeling using Shallow ReLU Networks with Grassmann Layers

Kayla Bollinger KBOLLING@ANDREW.CMU.EDU
and
Hayden Schaeffer HSCHAEFF@ANDREW.CMU.EDU
Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA, USA

Editors: Joan Bruna, Jan S Hesthaven, Lenka Zdeborova

Abstract

This paper presents a nonlinear model reduction method for systems of equations using a structured neural network. The neural network takes the form of a "three-layer" network with the first layer constrained to lie on the Grassmann manifold and the first activation function set to identity, while the remaining network is a standard two-layer ReLU neural network. The Grassmann layer determines the reduced basis for the input space, while the remaining layers approximate the nonlinear input-output system. The training alternates between learning the reduced basis and the nonlinear approximation, and is shown to be more effective than fixing the reduced basis and training the network only. An additional benefit of this approach is, for data that lie on low-dimensional subspaces, that the number of parameters in the network does not need to be large. We show that our method can be applied to scientific problems in the data-scarce regime, which is typically not well-suited for neural networks approximations. Examples include reduced order modeling for nonlinear dynamical systems and several aerospace engineering problems.

Keywords: Reduced Order Modeling, Grassmann Manifold, ReLU neural networks, Surrogate Models.

1. Introduction

Deep neural networks (DNN) are a popular approximation technique for problems in image processing, computer vision, natural language processing, and other data-based applications. Their popularity and success is due to their high accuracy, in particular, when trained with a sufficiently large training set. It has been shown that the expressive power of a neural network is a function of the number of trainable parameters, or equivalently for DNN, the number of layers Cybenko (1989); Yarotsky (2017), and thus one can obtain a given level of accuracy by using a large enough set of parameters. However, this can come with potential issues, especially when applied to problems in scientific computing and high-consequence decision making. Specifically, using a large set of trainable parameters often comes at the cost of longer training times, unnecessary model complexity, and more expensive evaluations. The increase of complexity and evaluation cost can make the network impractical for applications that require repeated queries, like uncertainty analysis and optimization. In addition, expressiveness itself is not the only requirement for applicability of a model, one must also consider stability, robustness, and interpretability which all depend on the structure and size of the network.

The goal of this work is to construct a reduced order model (ROM) for approximating high-dimensional functions by incorporating a model reduction layer within a shallow neural network.

The hope is to maintain the expressiveness of the neural network, but lower the overall complexity through the introduction of these new layers. Since the cost and complexity often increases dramatically with the dimension of the input space, the so called *curse of dimensionality*, reducing the effective dimension of the initial layer should lead to large gains. In addition, this creates a blend between interpretability (through the model reduction layers) and expressiveness (through the neural network), which can be beneficial for many applications.

ROMs are used to approximate high-dimensional complex systems by simpler, and often interpretable, low-dimensional functions which capture the overall dominant behavior of the high-dimensional system. Projection-based model reduction techniques often construct a low-dimensional approximation directly on the data. One popular approach is the Proper Orthogonal Decomposition (POD), which learns a low-dimensional subspace using the dominant principal components of the data-matrix Berkooz et al. (1993); Holmes et al. (2012). Then, the original system of equations are transformed to the reduced-space by projecting onto the low-dimensional subspace. Some other approaches for approximating a reduced basis includes: global sensitivity analysis Saltelli et al. (2008), sliced inverse regression Li (1991), and a compressive sensing based approach from Fornasier et al. (2012). In many cases, it is advantageous to learn both the reduced basis and the governing equations in the reduced domain. For dynamical systems, the dynamic mode decomposition (DMD) extracts a reduced order model by projecting the data onto a low-dimensional subspace and learning a linear evolution equation on the reduced basis Rowley et al. (2009); Schmid (2010). The operator inference technique Peherstorfer and Willcox (2016) can be used to learn a polynomial governing system on the reduced basis, which can better capture the nonlinear behavior on the reduced space, in particular, when the original high-dimensional function is also a polynomial.

For certain problems in scientific computing, both point queries and gradient information can be made available. Gradient-based model reduction algorithms often use the dominate eigenspace of the second moment matrix as a way of finding the reduced basis. This approach is called the active subspace method Russi (2010); Constantine et al. (2014); Constantine (2015) and has been applied to the identification and analysis of structures in hypersonic flow Constantine et al. (2015a), transonic flow Lukaczyk et al. (2014), hydrological models Jefferson et al. (2015, 2017), and ion batteries Constantine and Doostan (2017). Active subspaces can also be used to improve the cost of solving Bayesian inverse problems, see for example Cui et al. (2014); Constantine et al. (2016). If the input space has ambient dimension m, then under mild assumptions on the function, the error associated with using the k-dimensional active subspace (for $k \leq m$) is given by the tail sum of the eigenvalues from $k+1$ to m. Thus, when the eigenvalues of the second moment matrix decay rapidly, the active subspace method will be accurate. These methods depend on estimating the spectra accurately using gradient measurements of the high-dimensional function. To avoid the need for many accurate measurements, multifidelity methods Lam et al. (2020) can be used to reduce the overall cost by utilizing both high-fidelity and low-fidelity gradient evaluations.

There are several challenges related to reduced order modeling (ROM). Consider the scalar case: $f : \mathbb{R}^m \to \mathbb{R}$, where $m \gg 1$ and where we have constructed the low-dimensional subspace associated with the matrix $U \in \mathbb{R}^{m \times k}$ and projected the function onto the reduced basis, i.e. $F : \mathbb{R}^k \to \mathbb{R}$ with $F(y) = f(Uy)$, where $y \in \mathbb{R}^k$. Even though the input to F is k-dimensional, if we are required to evaluate f directly, then we may not have reduced the overall cost. Instead, one can build a surrogate function $g : \mathbb{R}^k \to \mathbb{R}$ so that $g(U^T x)$ is an approximation to $f(x)$ over a set of given samples. In this way, an evaluation of g has complexity depending on k and not the ambient dimension m Rowley et al. (2009); Schmid (2010); Peherstorfer and Willcox (2016); Constantine

et al. (2017); Hokanson and Constantine (2018). Note that in this work, we look at general maps $f : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}^n$, where m and n are not necessarily equal, and we do not assume that the data is obtained in a particular way. This is in contrast to the DMD method which obtains a ROM for the output space when the data is obtained sequentially (i.e. time snapshots of a dynamical system).

Additionally, we focus on the data-scarce setting, where the number of samples of the data may be smaller than required to get an accurate approximation to the second moment matrix or needed for an accurate approximation in the ambient dimension. This leads to several potential sources of errors throughout the modeling process. First, inaccuracies in the second moment matrix can be viewed as "noise" to the active subspace approximation and thus can lead to an inaccurate approximation of the dominate eigenspace. Secondly, there is error in the dataset from having both low-fidelity and limited samples. Lastly, in this setting, over-parameterized functions could lead to overfitting and inconsistent results.

1.1. Contribution of this work

We propose a joint optimization method for simultaneously learning the reduced basis and the surrogate model from point-queries. Since the low-data setting can lead to inaccuracies in the eigenspace of the second moment matrix, rather than using the active subspace method directly, we will use it as an initial guess within our method. To approximate the nonlinear system on the reduced basis, we build a surrogate function parameterized by a two-layer fully connected ReLU network. The two-layer network with the subspace reduction layer can be thought of as a "three-layer" network with the first layer constrained to be a low-dimensional projection layer and the first activation function to be identity.

Although the data-scarce regime is typically not well-suited for neural networks, since we are jointly optimizing the low-dimensional projection within the network structure, the number of free parameters is far fewer than a full network in the ambient dimension. Thus, this approach potentially avoids some of the generalizability issues associated with over-parameterization on small datasets, at least for the applications in this work. We show empirically that this approach is robust to the sample-size and can be applied to various aerospace engineering problems.

Several works have proposed various neural network based ROMs, or ROMs to enhance neural networks. For example, the POD method can be used with a neural network in order to construct a non-linear ROM for applications in fluid flows Hesthaven and Ubbiali (2018); Lui and Wolf (2019). For applications in PDEs, Bhattacharya et al. (2020) proposed a PCA-based approach that utilized a neural network to map reduced spaces. In Daniel et al. (2020), neural networks are used to pick models from a dictionary of local ROMs (constructed via a POD based approach). The POD based approaches differ from our work since they reduce the output space, rather than the model's dependencies on the input variables. Alternatively, neural networks can serve as ROMs when using an encoder/decoder structure, for example Hinton and Salakhutdinov (2006); Ravi (2017). In a related direction, several works have studied (low-dimensional) manifold learning using deep neural networks Shaham et al. (2018); Chui and Mhaskar (2018); Zhu et al. (2018). While encoder/decoder networks seem to perform well empirically, they do lack interpretability, which may limit their use in scientific computing. As mentioned in O'Leary-Roseberry et al. (2020), unless the dimension of the reduced basis is known a priori, it may be difficult to find this in a systematic way within the encoder/decoder framework. In contrast, with a linear method (like active subspaces), the dimension of the reduced basis is given by the decay of the spectral values. In our proposed algorithm, we

balance between these two perspectives by making the encoder linear with an interpretable reduced order basis based on a theoretical foundation (through the active subspace approach) and improve the accuracy of the ROM by using a neural network decoder. It is also worth mentioning that our network is small relative to what would be needed for a full encoder/decoder network, which typically utilizes a deep network structure. So while our network decoder is still a black-box, by using a shallow network with a small number of parameters we can better probe into the intrinsic dependencies.

Lastly, we note that in a concurrent work O'Leary-Roseberry et al. (2020), a similar approach for constructing ROMs using neural networks is used. However, a notable difference is that while O'Leary-Roseberry et al. (2020) uses the active subspace method to construct a (fixed) low dimension basis for the input space, our proposed algorithm uses it as an initial guess and optimizes the the low dimensional basis simultaneously with the surrogate model.

2. Problem Statement

Given a function $f : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}^n$, where $m \gg 1$, the goal is to construct a surrogate model of the form:

$$f(x) \approx g(U^T x),$$

where $U \in \mathbb{R}^{m\times k}$ is a matrix that maps the input space to a k-dimensional subspace $k < m$ and $g : \mathbb{R}^k \to \mathbb{R}^n$ is the approximation of the function with respect to k-dimensional inputs. Therefore, the problem is to approximate $f \approx g \circ U^T$ by learning both U and g from a set of M-samples $\{x_\ell\}_{\ell=1}^M$ (with respect to a sampling measure ρ) in order to find a model that depends on fewer inputs than the ambient dimension.

For our approach, we represent the lower dimensional function g by a shallow neural network and we construct U by a gradient-based dimensional reduction method (inspired by the active subspace technique). In particular, we write $g = g_\theta$ where $\theta \in \mathbb{R}^d$ is the set of trainable parameters that define the shallow neural network and constrain the range of U to be on the Grassmann manifold $Gr(k, m)$. To learn the parameters θ and U, we solve the following minimization problem:

$$\min_{\theta\in\mathbb{R}^d,\ \text{Range}\, U\in Gr(k,m)} \frac{1}{M}\sum_{\ell=1}^{M} \left\| f(x_\ell) - g_\theta(U^T x_\ell)\right\|_2^2 + \lambda\|\theta\|_2^2 \tag{1}$$

where the first term is the empirical risk and the second term is a regularizer on the parameters of the neural network, with $\lambda > 0$. To solve Equation (1), we use an alternating minimization strategy between minimizing the fit of the network for a fixed matrix U and minimizing over U with fixed parameters θ. The algorithm is detailed in Section 3. In Sections 2.1-2.2, we describe the construction of g and U, and recall some of the theory related to the active subspace methods.

2.1. Active Subspaces

We first recall some of the theory of active subspaces (see Constantine et al. (2014)), its application to vector-valued functions, and its connection to our problem.

Consider a domain $\Omega \subseteq \mathbb{R}^m$ (centered at the origin) equipped with a probability density ρ, that is

$$\rho(x) > 0 \text{ for } x \in \Omega, \quad \rho(x) = 0 \text{ for } x \notin \Omega.$$

Let f be continuously differentiable, square integrable with respect to ρ, and let the pairwise products of its partial derivatives also be integrable with respect to ρ. Consider the symmetric and positive semidefinite matrix $C \in \mathbb{R}^{m\times m}$ defined by

$$C := \mathbb{E}\left[Df^T Df\right]$$

$$\text{where} \quad (Df(x))_{i,j} = \frac{\partial f_i}{\partial x_j}(x) \quad \text{and} \quad \left(Df^T(x)Df(x)\right)_{i,j} = \sum_{k=1}^{n} \frac{\partial f_k}{\partial x_i}(x)\frac{\partial f_k}{\partial x_j}(x).$$

Its eigenvalue decomposition can be written as $C := W\Lambda W^T$ with $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_m)$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$. Define the block-wise structure as follows: $W_1 \in \mathbb{R}^{m\times k}$, $W_2 \in \mathbb{R}^{m\times m-k}$, $\Lambda_1 \in \mathbb{R}^{k\times k}$, and $\Lambda_2 \in \mathbb{R}^{m-k\times m-k}$ and thus:

$$W = \begin{bmatrix} W_1 & W_2 \end{bmatrix} \quad , \quad \Lambda = \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix}.$$

As in Constantine et al. (2014), we define the rotated coordinates $y \in \mathbb{R}^k$ and $z \in \mathbb{R}^{m-k}$ by $y = W_1^T x$ and $z = W_2^T x$. We next prove an extension of Lemma 2.2 from Constantine et al. (2014) related to the rotated coordinates of vector-valued functions.

Lemma 1 *The sum of the mean-squared gradients of each f_i with respect to the rotated coordinates y and z satisfy:*

$$\sum_{i=1}^{n} \mathbb{E}\left[\nabla_y(f_i)^T \nabla_y(f_i)\right] = \lambda_1 + \cdots + \lambda_k$$

$$\sum_{i=1}^{n} \mathbb{E}\left[\nabla_z(f_i)^T \nabla_z(f_i)\right] = \lambda_{k+1} + \cdots + \lambda_m$$

where λ_i for $i \in [m]$ are the ordered eigenvalues of $C = \mathbb{E}\left[Df^T Df\right]$.

The proof of Lemma 1 is in Appendix A.1. Since λ_j are non-negative and ordered, Lemma 1 provides a comparison between the amount that f varies on each of the subspaces associated to W_i. In particular, if λ_{k+1} is relatively small, then f depends mainly on the subspace associated with W_1, i.e. $\{y : y = W_1^T x, x \in \Omega\} \subseteq \mathbb{R}^k$. If $\lambda_{k+1} = 0$, then f is invariant to the subspace W_2. Note that we have assumed that each component of $f = [f_1, \cdots, f_n]^T$ depends jointly on the same lower dimensional subspace. This means that obtaining a reduced order model for the function f can be done simultaneously in all components. If this is not the case, then the model reduction must be applied to each component f_i for $i \in [n]$ separately, i.e. a sequence of parallel optimization problems for $f_i : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}$. The results in this section still hold in this setting, since they degenerate to the one-dimensional case discussed in Constantine et al. (2014).

For a fixed U, the minimizer of the risk:

$$\min_{g=[g_1,\cdots,g_n]^T} \sum_{i=1}^{n} \int_{\Omega} \left(f_i(x) - g_i(U^T x)\right)^2 \rho(x)dx \tag{2}$$

is given by the conditional expectation:

$$g_i(W_1^T x) = g_i(y) := \mathbb{E}\left[f_i(W_1 y + W_2 z)|y\right] = \int f_i(W_1 y + W_2 z)\rho(z|y)\, dz \tag{3}$$

where $\rho(z|y)$ is the conditional probability density. As we see in the following theorem, we have that indeed f is close to $g \circ W_1^T$ whenever λ_{k+1} is small.

Theorem 2 *Assume a given probability density ρ is such that the probabilistic Poincaré inequality with respect to $\rho(z|y)$ holds, that is, assuming that the gradient of $\phi : \mathbb{R}^m \to \mathbb{R}$ is square integrable, then:*

$$\mathbb{E}\left[(\phi - \mathbb{E}\left[\phi|y\right])^2 \Big| y\right] \leq c\,\mathbb{E}\left[\|\nabla_z \phi\|_2^2 \Big| y\right]$$

for some constant $c > 0$ depending on the domain Ω and ρ. Then the mean-squared error of $g \circ W_1^T$ satisfies

$$\mathbb{E}\left[\left\|f - g \circ W_1^T\right\|_2^2\right] \leq c\,(\lambda_{k+1} + \cdots \lambda_m),$$

where λ_i for $i \in [m]$ are the ordered eigenvalues of $C = \mathbb{E}\left[Df^T Df\right]$.

The proof of Theorem 2 is in Appendix A.2. This is a vector-valued version of Theorem 3.1 from Constantine et al. (2014). Note that if the eigenvalues decay sufficiently rapidly, then the risk will be small.

Since we are given a set of M-samples $\{x_\ell\}_{\ell=1}^M$, we obtain an approximation to C by the Monte Carlo estimate $\tilde{C} \in \mathbb{R}^{m \times m}$ defined by:

$$C \approx \tilde{C} := \frac{1}{M}\sum_{\ell=1}^{M} Df(x_\ell)^T Df(x_\ell), \quad \text{i.e.} \quad \tilde{C}_{i,j} = \frac{1}{M}\sum_{\ell=1}^{M}\left(\sum_{k=1}^{n} \frac{\partial f_k}{\partial x_i}(x_\ell)\frac{\partial f_k}{\partial x_j}(x_\ell)\right)$$

where $x_\ell \in \Omega$ are M independent and identically distributed (i.i.d.) samples with respect to the sampling measure ρ. To approximate the matrix $\tilde{W}_1$, we apply the eigenvalue decomposition to $\tilde{C}$, i.e. $\tilde{C} = \tilde{W}\Lambda\tilde{W}^T$, where $\tilde{W}_1 \in \mathbb{R}^{m \times k}$, $\tilde{W}_2 \in \mathbb{R}^{m \times m-k}$, are the block-matrices as defined earlier in this section. To compute $\tilde{W}_i$, we use the SVD approach. Let $A \in \mathbb{R}^{m \times nM}$ be defined by:

$$A := \frac{1}{\sqrt{M}}\left[Df(x_1)^T \cdots Df(x_M)^T\right], \tag{4}$$

whose SVD is given by $A = U\Sigma V^T$. Since $AA^T = \tilde{C}$, we then have that $U = \tilde{W}$.

2.2. Network Approximation and Alternating Approach

We propose the following minimization problem for obtaining a reduced order model $g(U^T x)$ given a set of M-samples $\{x_\ell\}_{\ell=1}^M$:

$$\min_{\theta \in \mathbb{R}^d,\ \text{Range}\, U \in Gr(k,m)} \frac{1}{M}\sum_{\ell=1}^{M} \left\|f(x_\ell) - g_\theta(U^T x_\ell)\right\|_2^2 + \lambda\|\theta\|_2^2. \tag{5}$$

To solve this minimization problem, we alternate between the following two subproblems:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{M}\sum_{\ell=1}^{M} \left\|f(x_\ell) - g_\theta(U^T x_\ell)\right\|_2^2 + \lambda\|\theta\|_2^2 \quad \text{(NN Approximation)}$$

and

$$\min_{\text{Range}\, U \in Gr(k,m)} \frac{1}{M} \sum_{\ell=1}^{M} \left\| f(x_\ell) - g_\theta(U^T x_\ell) \right\|_2^2 \quad \text{(Subspace Approximation)}.$$

First, to approximate the nonlinear function over the subspace, we write g_θ as a shallow neural network with trainable parameters $\theta \in \mathbb{R}^d$. The shallow neural network takes the form of a two-layer ReLU network:

$$g_\theta(y) = A_2(\text{ReLU}(A_1 y + b_1))) + b_2$$

where the first fully connected linear layer is defined by the matrix $A_1 : \mathbb{R}^k \to \mathbb{R}^h$ and bias $b_1 \in \mathbb{R}^h$ and the second fully connected linear layer is defined by the matrix $A_2 : \mathbb{R}^h \to \mathbb{R}^n$ and bias $b_2 \in \mathbb{R}^n$. We use the ReLU activation function: $(\text{ReLU}(z))_i = \max(z_i, 0)$. The trainable parameter vector $\theta \in \mathbb{R}^d$ is defined as the vector of parameters after concatenating all elements of A_1, A_2, b_1, and b_2. The total number of learnable parameters is $d = h(k + n + 1) + n$.

In Section 2.1, it was shown that the minimizer over all functions g is obtained by the conditional expectation; however, approximating the conditional expectation (e.g. via the Monte Carlo approximation), would not necessarily be beneficial since the approximation would require evaluations in the ambient dimension m. Instead, we want to provide a nonlinear surrogate model in the smaller dimension $k < m$, whose complexity depends on k not m. This is one motivation for learning the model through regression.

The other reason to use regression is to deal with the various sources of noise. In the applications discussed in Section 1, the noise can arise from multiple sources involving computational errors or inaccuracies. The first is that the underlying function f may not be invariant to the subspace $\{z : z = W_2^T x, x \in \Omega\} \subseteq \mathbb{R}^{m-k}$ (or in a related sense, the decay of the eigenvalues from Theorem 2 is slow) and thus the model may incur "noise" from the information lost during the subspace reduction. Other sources of noise can come from the training data itself. For example, if the data pair $(x, f(x))$ is obtained by a numerical simulation of some complex system with modeling parameter x, then $f(x)$ is only an approximation to some underlying function. The numerical accuracy (or inaccuracy in this case) would appear as "noise" in the surrogate approximation. Additionally, when using a small sample set, which is the case for some of the experiments in Section 4, the error in approximating the data distribution (for example, the error between C and $\tilde{C}$) can appear as "noise". And lastly, if the data is obtained from an experiment or real-world observations, then measurement and/or acquisition noise may be present. This motivates the use of regularized regression, and in particular, the utilization of shallow neural networks to simultaneously fit the data and denoise the system.

To obtain U, we fix g_θ and then optimize over the Grassmann manifold $Gr(k, m)$–a compact manifold of all k-dimensional linear subspaces in $\mathbb{R}^m$. We use $Gr(k, m)$ since we are only concerned with obtaining a k-dimensional subspace, which is represented by the matrix U, and not with the representation itself (i.e. the choice of coordinates). When gradient information is available, we use the procedure described by Equation (4) to obtain an approximation to the active subspace $\tilde{W}_1$ and use it as the initial guess to U. When we only have function evaluations (and are unable to obtain the active subspace), then a random initialization scheme is used. An experimental comparison between different initializations is provided in Section 4.1. In general, experiments that use the active subspace as the initializer perform better (in terms of accuracy/generalizability). The minimization problem in U is nonconvex, and thus good initialization can be very helpful.

3. Methodology

The optimization of Equation (5) is done via the alternating minimization approach described in Section 2.2. The pseudocode for this algorithm is given in Algorithm 1, and is described in detail below. We initialize the trainable parameters of the shallow neural network, i.e. θ, using the standard Xavier initialization. In particular, each fully connected layer is comprised of weight matrices $A_1 \in \mathbb{R}^{h\times k}$, $A_2 \in \mathbb{R}^{n\times h}$ and biases $b_1 \in \mathbb{R}^h$, $b_2 \in \mathbb{R}^n$, which are initialized uniformly at random: $(A_1)_{i,j}, (b_1)_i \sim \mathcal{U}\left[-\frac{1}{\sqrt{k}}, \frac{1}{\sqrt{k}}\right]$ and $(A_2)_{i,j}, (b_2)_i \sim \mathcal{U}\left[-\frac{1}{\sqrt{h}}, \frac{1}{\sqrt{h}}\right]$. We note that our model takes k as a hyperparameter, which can be inferred using the spectral decay and the theory of active subspace. For the experiments presented in this paper either the appropriate value for k is known a priori (Experiment 1, Section 4.1), or a range of values for k are tested (Experiments 2-5, Sections 4.2-4.5).

For the reduced basis subproblem, if Jacobian-information is given, i.e. measurements of $Df(x_\ell)$ over a set of samples, then we initialize U using the active subspace method, that is, $U = W_1$ described in Section 2.1. When the Jacobian is not available, we use two additional initialization methods for U (see also Constantine et al. (2017)). The first is to initialize U as the identity operator on the first k-coordinates of the input space, i.e.

$$U = \begin{bmatrix} I_{k\times k} \\ 0_{m-k\times k} \end{bmatrix}.$$

The second way is to initialize U at random with the constraint that is remains orthogonal. This is done by creating a random Gaussian matrix of size $m \times k$ whose elements are chosen from the normal distribution with mean zero and standard deviation 1, and then setting U to be the orthogonal matrix Q computed via the reduced QR factorization of the random Gaussian matrix.

To optimize Equation (5) we alternate between optimizing over θ, then over U. When the active subspace is used as the initial guess, this ordering can lead to dramatic gains in the early training phase. For the NN approximation subproblem, we use the ADAM method Kingma and Ba (2014) with an initial learning rate τ. For the subspace approximation subproblem, we use the Pymanopt package (Townsend et al. (2016)) to solve the Grassmann manifold-constrained least-squares problem using a steepest descent method. We refer to any iteration carried out in these optimization steps as an inner iteration, and we refer to the completion of one-sweep of optimizing over both θ and U as an outer iteration.

For the NN approximation subproblem (see Section 2.2), we set a fixed number of N_θ inner iterations. The number of inner iterations can be set by the users (and thus included as a hyperparameter), or it can be determined by a stopping condition on the loss function for each subproblem. In the experimental results in Section 4, we chose a sufficiently large number of steps, $N_\theta = 5000$, to ensure plateauing of the loss functions for these examples. Some tests showed that early stopping (with a relaxed stopping condition) can reduce the training time while resulting in a similar overall loss (for the full optimization problem). After each outer iteration the learning rate decays, i.e. we reinitialize the hyperparameters in the ADAM method with a reduced learning rate $\tau \leftarrow 0.9\tau$. We set a fixed number of P outer iterations based on empirical tests, in particular, P should be large enough to allow the loss in Equation (5) to plateau. We define $\theta^{p,n}$ and $U^{p,n}$ to be the computed values of θ and U after p outer iterations and n (of their respective) inner iterations.

Algorithm 1 Shallow neural-network/active subspace-based alternating minimization scheme

Given: M input/output tuples $(x_\ell, f(x_\ell), Df(x_\ell))$
Initialize: $\theta \in \mathbb{R}^d$, $U \in \mathbb{R}^{m\times k}$, learning rate = τ, iteration counts N_θ and P
while $p \leq P$ **do**
(1) Compute θ^{p,N_θ} starting with $\theta^{p,0}$ and applying the ADAM method to:

$$\min_{\theta\in\mathbb{R}^d} \frac{1}{M}\sum_{\ell=1}^{M} \left\| f(x_\ell) - g_\theta((U^{p,0})^T x_\ell)\right\|_2^2 + \lambda\|\theta\|_2^2$$

for N_θ steps. Then update $\theta^{p+1,0} = \theta^{p,N_\theta}$.
(2) Compute U^{p,N_U} as the solution to the Grassmann manifold-constrained least squares problem:

$$\min_{\text{Range}\, U\in Gr(k,m)} \frac{1}{M}\sum_{\ell=1}^{M} \left\| f(x_\ell) - g_{\theta^{p+1,0}}(U^T x_\ell)\right\|_2^2 .$$

Then update $U^{p+1,0} = U^{p,N_U}$.
(3) Update: $\tau = 0.9\tau$
end while

4. Experimental Results and Applications

The model contains several hyperparameters, which we list here: the dimension of the reduced basis k, the hidden dimension of the shallow network h, the regularization weight λ, and the learning rate τ. For simplicity, let X_{Train}, $X_{Validation}$, and X be the training, validation, and entire available dataset (respectively) and we define their cardinalities by $|X_{Train}|$, $|X_{Validation}|$, and $|X|$. Note that $|X_{Train}| + |X_{Validation}| \leq |X|$, and that typically this holds with strict inequality in most experiments. Since we are randomly sampling a subset from X, we typically use fewer samples than available.

4.1. Experiment 1: U initialization comparison

We provide a comparison between the various initializations for U. In particular, we show that using an approximation to the active subspace, W_1, as our initial guess for U will produce an overall small loss at the end of the training, as compared to the other initializers. We compare the three initializations for U that were discussed in Section 3 on the problem of learning a reduced order model for a nonlinear dynamical system. In particular, define the function $f : \mathbb{R}^3 \to \mathbb{R}^3$ by

$$f(x) = \begin{bmatrix} x_2^3 \\ -\left(\frac{x_1+x_3}{2}\right)^3 - \frac{1}{5}x_2 \\ x_2^3 \end{bmatrix}$$

and the dynamical system to be the autonomous differential equation $\frac{d}{dt}x(t) = f(x(t))$. This example is related to the sparsity-promoting methods for learning unknown governing equations from data, see for example Brunton et al. (2016); Schaeffer (2017); Rudy et al. (2017); Raissi et al. (2018); Wu and Xiu (2019); Sun et al. (2020). However, in this work, we would like to learn both a

reduced basis and a nonlinear governing system approximation, as done in the DMD methodology and in Peherstorfer and Willcox (2016).

In this toy example, the system can be reduced to $f(x) = g(U^T x) = g(y)$ where

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 \\ \frac{1}{\sqrt{2}} & 0 \end{bmatrix}, \quad g(y) = \begin{bmatrix} y_2^3 \\ -\left(\frac{y_1}{\sqrt{2}}\right)^3 - \frac{1}{5} y_2 \\ y_2^3 \end{bmatrix}$$

The dataset was constructed using a randomized approach, similar to Schaeffer et al. (2018, 2020). The dataset is generated by collecting 500 trajectories starting from a uniformly random initial state x^0 centered at $\tilde{x}^0 = [4, 3, -2]^T$ with width 2, i.e. $x^0 \in \mathcal{U}[\tilde{x}^0 - 2, \tilde{x}^0 + 2]$. The trajectories were generated using the RK45 method with 202 equally spaced time-stamps over the time interval $[0, 5]$. The Jacobian can be extracted directly from $f(x)$. The entire data set X consists of $|X| = 101,000$ samples. For training our model, we use 150 trajectories for our training set X_{Train} and 30 trajectories for our validation set $X_{Validation}$ (i.e. $|X_{Train}| = 30,300$, $|X_{Validation}| = 6,060$). We used a batch size of 16 trajectories in the NN approximation subproblem. The hyperparameters for our model were set to: $k = 2$, $h = 8$, $\lambda = 10^{-7}$, and $\tau = 10^{-3}$.

Using 100 randomized trials for training our model, we record our results in Figure 1. In Figure 1, we plot the epochs (which are the inner iterations for the NN approximation subproblem over the entire training process) versus the relative error, which we define as:

$$\text{RelError}_{p,n} = \left(\frac{\frac{1}{|X_{Validation}|} \sum_{x \in X_{Validation}} \left\| f(x) - g_{\theta^{p,n}}((U^{p,0})^T x) \right\|_2^2}{\frac{1}{|X|} \sum_{x \in X} \|f(x)\|_2^2} \right)^{\frac{1}{2}}.$$

The mean-squared errors are normalized separately by the number of points. Comparing the identity and random initializations, which do not use Jacobian-information, we see that the methods have similar relative errors and will plateau at roughly the same relative error (taking another 10^5 epochs). The active subspace initialization leads to more dramatic gains in the initial training process, and overall faster convergence. Note that there are still variations on the reduced basis in the training process, i.e. the final subspace may not directly agree with the active subspace that was used to initialize the training. In all cases, the 'jumps' in the error occur due to the new estimate of $U^{p,0}$. The jumps in the solid purple curve in Figure 1 indicate that the alternating minimization leads to a lower relative error than just using the active subspace as the reduced basis.

4.2. Experiment 2: Comparing methods using NACA0012

This experiment demonstrates the robustness of our model in the scarce-data setting as compared to other approaches. In this test, we find a reduced basis and surrogate model for the drag coefficient associated with the NACA0012 airfoil with respect to 18-shape parameters. The drag coefficient was computed using Stanford University Unstructured (SU2) computational fluid dynamics code Economon et al. (2016); Hokanson and Constantine (2018). The results and comparisons are plotted in Figure 2. We compare our model to the degree-5 polynomial ridge regression model from Hokanson and Constantine (2018) using subspaces of dimensions one, three, and five. In addition, we compare our approach with a Gaussian process regression Rasmussen and Williams (2005); Pedregosa et al. (2011) the LASSO problem (with cross-validation) with a dictionary of monomials

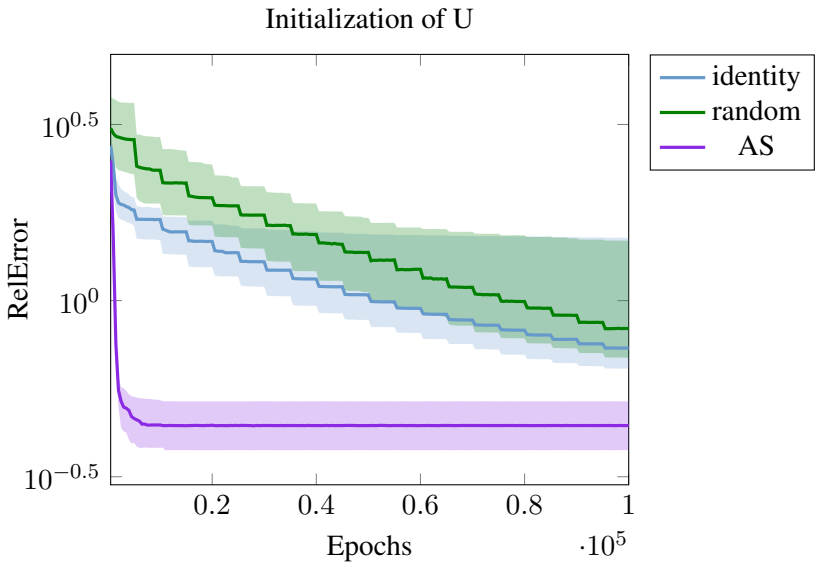

Figure 1: Experiment 1: The plot compares the relative error over the training epochs for the three initialization schemes described in Section 3: identity initialization (in blue), random initialization (in green), and the active subspace initialization (in purple). The median value over 100 trials is displayed as the solid line, and the shaded region encloses the 25^{th} to 75^{th} percentile. Note that the active subspace initialize reduces the error faster than the other schemes.

up to degree 3 Pedregosa et al. (2011); and a standard quadratic regression problem. The Gaussian process, LASSO, and quadratic regression do not perform model reduction on the input parameters. For each model, we ran 100 trials using randomly chosen data points for the training set using the sizes $\{10, 25, 50, 100, 250, 500, 1000\}$ (except for the quadratic model, in which we only use $\{250, 500, 1000\}$ since it becomes ill-conditioned for smaller sets). For each trial, we calculate their relative errors over the entire dataset X, which is defined by:

$$\text{RelError}_X = \left(\frac{\sum_{x \in X} \left\| f(x) - \tilde{f}(x) \right\|_2^2}{\sum_{x \in X} \|f(x)\|_2^2} \right)^{\frac{1}{2}}$$

where $\tilde{f}$ represents the trained model for a given approach. For our model, we used $P = 10$ outer iterations, batch size of 16 in the NN approximation subproblem, and the remaining hyperparameters were set to: $h = 8$, $\lambda = 10^{-7}$, and $\tau = 10^{-3}$.

We use the quadratic model as a baseline since it has no dimension reduction and seems to obtain accurate results when given a sufficient number of samples (see also Hokanson and Constantine (2018)). This success may be attributed to the model having access to all 18 input variables, and having 190 free parameters to tune in the data-abundant setting. In the data-scarce setting, we see that our model outperforms the others, supporting our claim of it being a robust alternative in this data regime. As we increase the training set size, the reduced basis of dimensions 3 and 5 begin

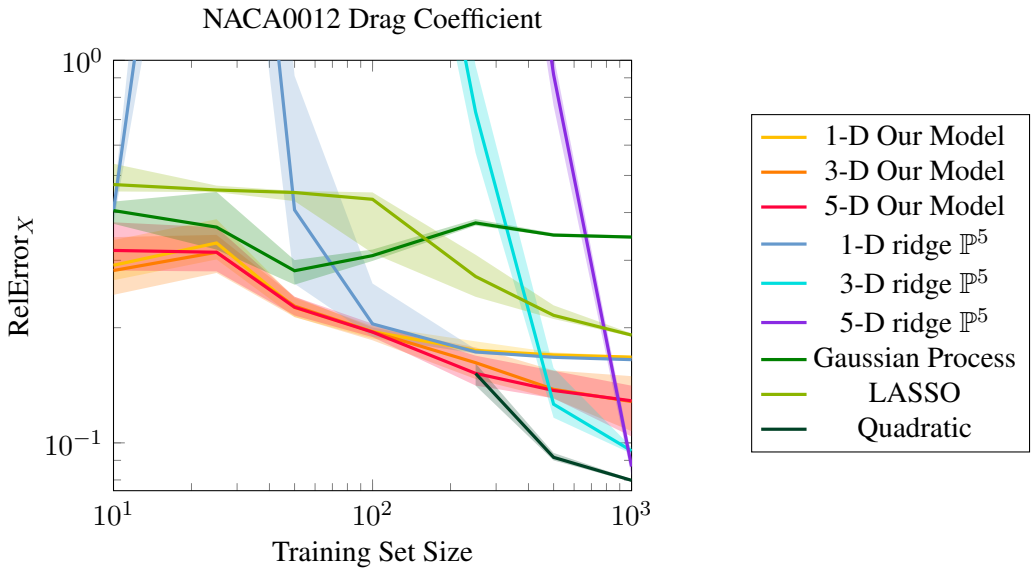

Figure 2: Experiment 2: Comparison of various methods for approximating the NACA0012 drag coefficient with respect to 18 shape parameters. Our method is labelled as "k-D Our Model" where the reduced basis dimensions are $k = 1, 3, 5$. The "k-D ridge" curves correspond to the degree-5 polynomial ridge regression model from Hokanson and Constantine (2018) using active subspaces of dimensions $k = 1, 3, 5$. We also compare to the Gaussian process regression Rasmussen and Williams (2005); Pedregosa et al. (2011), the LASSO problem with a dictionary of monomials up to degree 3 Pedregosa et al. (2011); and a standard quadratic regression problem. The median relative error is displayed as the solid curves, and the shaded region encloses the 25^{th} to 75^{th} percentile (using 100 random trials).

to outperform the 1D model. The LASSO model is also robust, but does not reduce the input dimension since sparsity is imposed in the representation (monomials) themselves and not the input space.

Note also that the computational cost of evaluating each of these models roughly scales with their respective number of free parameters. Due to its shallow network structure, our model excels in this regard. To illustrate, we compare the two ROM architectures presented here: ignoring the dimension reduction step (which both models share), our 5D network approximation uses 57 free parameters while the 5D polynomial ridge regression approximation uses 252. This is also smaller than the deep networks mentioned in Section 1.

4.3. Experiment 3: Dependence on Basis Dimension and Hidden Dimension

The number of free parameters in our model depends on the dimension of the reduced basis, denoted by k, and the hidden dimension of the two-layer neural network, denoted by h. In this experiment, we apply our method to the NACA0012 airfoil example and measure the error as a function of various values for both k and h. Specifically, we vary $k \in \{1, 2, 3\}$ and $h \in \{8, 64, 256\}$, run

100 trials using randomly chosen data points, and compare the results in the first table in Table 1. All other hyperparameters are fixed: $P = 10$ outer iterations were used, training and validation set size is 50, batch size is 16, $\lambda = 10^{-7}$, and learning rate $\tau = 10^{-3}$. We see that as the subspace dimension increases, the number of hidden layers needed to resolve the function also increases. Outside of the case $(k, h) = (1, 256)$, where over-fitting may be occurring, the results seem to improve as h increases.

4.4. Experiment 4: Comparison with a Bowtie Network

similar network structure can be achieved by using a bowtie network. We compare our model to a shallow bowtie neural network model defined by $\tilde{f}_{\tilde{\theta}} : \mathbb{R}^m \to \mathbb{R}^n$, where

$$\tilde{f}_{\tilde{\theta}}(x) = A_2(\text{ReLU}(A_1(A_0 x) + b_1))) + b_2.$$

The main difference between the bowtie network and our model is that the Grassmann layer is replaced by a fully connected linear layer with $A_0 : \mathbb{R}^m \to \mathbb{R}^k$ and without a bias term. The remaining layers are defined as described in Section 2.2: the first fully connected linear layer is defined by the matrix $A_1 : \mathbb{R}^k \to \mathbb{R}^h$ and bias $b_1 \in \mathbb{R}^h$ and the second fully connected linear layer is defined by the matrix $A_2 : \mathbb{R}^h \to \mathbb{R}^n$ and bias $b_2 \in \mathbb{R}^n$. The total number of trainable parameters, i.e. the size of $\tilde{\theta} \in R^{\tilde{d}}$, is $\tilde{d} = km + h(k + 1) + n(h + 1)$. Note that this is larger than the number of free parameters in our model, since we have the added constraint. We compare the bowtie model with the same set of dimensional parameters: $k \in \{1, 2, 3\}$ and $h \in \{8, 64, 256\}$. The bowtie model was optimized over 50,000 epochs (the same total number of epochs/inner iterations used in the NN approximation subproblem for the experiment in Section 4.3), with the remaining hyperparameters were fixed: the training and validation set size was 50, the batch size was 16, $\lambda = 10^{-7}$, and $\tau = 10^{-3}$. We ran 100 trials using randomly chosen data points, and the results are summarized in the second table in Table 1.

The bowtie network produces about $1.2\times$ larger relative errors than our model. For fixed k, we see that in Table 1, as h increases the bowtie model can produce worse results. This is likely due to the lack of structure being imposed onto the system. The diagonal elements of the first table indicate that our model's accuracy improves as with the hidden dimension and basis dimension, which is not the case for the bowtie model. This has two benefits; the first is of practical importance, namely, that the user does not have to manually optimize the hyperparameters. The second is in terms of interpretability, the subspace approximation is likely preserving the dominate features and thus producing an overall model which better fits the dataset.

4.5. Experiment 5: Applications to Model Reduction for ONERA and HyshotII

In this experiment, we show that our approach can be applied to other datasets. Specifically, we compute a reduced order model for the drag coefficient associated with the ONERA-M6 wing with respect to 50-shape parameters Lukaczyk et al., and for the normalized integral of pressure over the end of the scramjet engine of the HyShot II vehicle with respect to 7-input parameters Constantine et al. (2015b). For the ONERA-M6 dataset, we have $|X| = 297$, $|X_{Train}| = 50$, and $|X_{Validation}| = 50$. This dataset includes derivative-information, thus we use the active subspace initialization for U. As in Experiments 3 (Section 4.3) and 4 (Section 4.4), we vary the same set of dimensional parameters: $k \in \{1, 2, 3\}$ and $h \in \{8, 64, 256\}$ and measure the relative error. The

	Our Model			Bowtie Model		
	h=8	**h=64**	**h=256**	**h=8**	**h=64**	**h=256**
k=1	0.411	0.411	0.484	0.469	0.474	0.468
k=2	0.408	0.373	0.370	0.497	0.481	0.523
k=3	0.403	0.374	0.331	0.486	0.508	0.505

Table 1: Experiment 3 and 4: This table contains the relative error in the drag coefficient associated with the NACA0012 airfoil with respect to 18-shape parameters, comparing our model using various dimensional parameters and a bowtie network of the same size. Specifically, the relative error being shown is the median of the minimum relative error calculated over 100 trials. The dimension of the reduced basis is denoted by k and the hidden dimension of the two-layer neural network is denoted by h.

remaining hyperparameters were set to: $P = 10$ outer iterations, batch size of 16, hidden dimension $h = 8$, $k = 1$, $\lambda = 10^{-7}$, and $\tau = 10^{-3}$. We ran 100 trials using randomly chosen data points, and the results are summarized in Table 2. We see that for large hidden dimension (i.e. $h = 256$) this model begins to overfit, likely due to the size of the training set. However, similar to the results of Experiment 3 (Section 4.3), the model does improve as the pair (h, k) increases.

For the HyShotII dataset, we show that a one-dimension subspace can be used as an accurate approximation for the 7D parameter system as was done in Constantine et al. (2015a). For this dataset, we have $|X| = 52$, $|X_{Train}| = 10$, and $|X_{Validation}| = 10$. This dataset does not include derivative information, so we used random initialization for U. This system has built in "noise" due to the possible errors in the numerical solvers used to generate the dataset. Using the hyperparameters: $P = 10$ outer iterations, batch size of 16, hidden dimension $h = 8$, $k = 1$, $\lambda = 10^{-7}$, and $\tau = 10^{-3}$, our model produces an approximation with a relative error of 0.211. Using these hyperparameters, we ran 100 trials using randomly chosen data points. We observed that for the HyshotII test, overfitting was observed when we used $h \geq 64$. From the various trials, we found that we only need a hidden dimension of $h = 8$, since the training set is very small. This also results in a more compact, and possibly more interpretable, approximation.

	ONERA-M6		
	h=8	**h=64**	**h=256**
k=1	0.115	0.121	0.150
k=2	0.110	0.088	0.096
k=3	0.113	0.076	0.079

Table 2: This table contains the relative error in the drag coefficient associated with the ONERA-M6 wing with respect to 50-shape parameters Lukaczyk et al. using our model with hidden dimension h and a reduced basis dimension k. Specifically, the relative error being shown is the median of the minimum relative error calculated over 100 trials. The dataset sizes are: $|X| = 297$, $|X_{Train}| = 50$, and $|X_{Validation}| = 50$.

5. Conclusion

We proposed an approach for constructing reduced order models using shallow neural networks with structured layers. The first fully connected layer is constrained to lie on the Grassmann manifold in order to map the input data onto a low dimensional subspace. Experimental results on the NACA0012 airfoil show that this constrained layer produces more robust results than a fully connected layer with the same dimensions. When gradient-information is available, the method is initialized using the active subspace method, which outperforms the standard initializers. In addition, it was shown that training the reduced basis jointly with the neural network produces smaller error than using a pre-trained, fixed basis. The method is applied to model reduction of nonlinear dynamical systems and aerospace engineering problems. In several examples, even though the number of samples is relatively small, our method is able to produce meaningful and (relatively) accurate surrogate models. Since the initial layer can dramatically decreases the dimension of the input space, this approach is easy to train and has smaller complexity than a neural network in the ambient dimension. This is beneficial in applications where the model must be queried many times. In addition, the mixture of a shallow and low-complexity neural network with the added constrained layers leads to more interpretable models than a standard neural network.

Acknowledgments

The authors would like to acknowledge the support of AFOSR, FA9550-17-1-0125 and the support of NSF CAREER grant #1752116. The authors would like to thank Jeffrey Hokanson for his help.

References

Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. *arXiv preprint arXiv:2005.03180*, 2020.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Charles K. Chui and Hrushikesh N. Mhaskar. Deep nets for local manifold learning. *Frontiers in Applied Mathematics and Statistics*, 4:12, 2018. ISSN 2297-4687. doi: 10.3389/fams.2018.00012. URL https://www.frontiersin.org/article/10.3389/fams.2018.00012.

Paul G Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM, 2015.

Paul G Constantine and Alireza Doostan. Time-dependent global sensitivity analysis with active subspaces for a lithium ion battery model. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(5):243–262, 2017.

Paul G. Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: Applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4), Jan 2014. ISSN 1095-7197. doi: 10.1137/130916138. URL http://dx.doi.org/10.1137/130916138.

Paul G Constantine, Michael Emory, Johan Larsson, and Gianluca Iaccarino. Exploiting active subspaces to quantify uncertainty in the numerical simulation of the HyShot ii scramjet. *Journal of Computational Physics*, 302:1–20, 2015a.

Paul G Constantine, Carson Kent, and Tan Bui-Thanh. Accelerating markov chain monte carlo with active subspaces. *SIAM Journal on Scientific Computing*, 38(5):A2779–A2805, 2016.

Paul G. Constantine, Armin Eftekhari, Jeffrey Hokanson, and Rachel A. Ward. A near-stationary subspace for ridge approximation. *Computer Methods in Applied Mechanics and Engineering*, 326, Nov 2017. ISSN 0045-7825. doi: 10.1016/j.cma.2017.07.038. URL http://dx.doi.org/10.1016/j.cma.2017.07.038.

P.G. Constantine, M. Emory, J. Larsson, and G. Iaccarino. Exploiting active subspaces to quantify uncertainty in the numerical simulation of the hyshot ii scramjet. *Journal of Computational Physics*, 302:1 – 20, 2015b. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2015.09.001. URL http://www.sciencedirect.com/science/article/pii/S002199911500580X.

Tiangang Cui, James Martin, Youssef M Marzouk, Antti Solonen, and Alessio Spantini. Likelihood-informed dimension reduction for nonlinear inverse problems. *Inverse Problems*, 30(11):114015, 2014.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

Thomas Daniel, Fabien Casenave, Nissrine Akkari, and D. Ryckelynck. Model order reduction assisted by deep neural networks (rom-net). *Advanced Modeling and Simulation in Engineering Sciences*, 7, 12 2020. doi: 10.1186/s40323-020-00153-6.

Thomas D. Economon, Francisco Palacios, Sean R. Copeland, Trent W. Lukaczyk, and Juan J. Alonso. Su2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 2016. doi: 10.2514/1.J053813. URL https://doi.org/10.2514/1.J053813.

Massimo Fornasier, Karin Schnass, and Jan Vybiral. Learning functions of few arbitrary linear parameters in high dimensions. *Foundations of Computational Mathematics*, 12(2):229–262, 2012.

J.S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.02.037. URL https://www.sciencedirect.com/science/article/pii/S0021999118301190.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. ISSN 0036-8075. doi: 10.1126/science.1127647. URL https://science.sciencemag.org/content/313/5786/504.

Jeffrey M. Hokanson and Paul G. Constantine. Data-driven polynomial ridge approximation using variable projection. *SIAM Journal on Scientific Computing*, 40(3), Jan 2018. ISSN 1095-7197. doi: 10.1137/17m1117690. URL http://dx.doi.org/10.1137/17M1117690.

Philip Holmes, John L Lumley, Gahl Berkooz, and Clarence W Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

Jennifer L Jefferson, James M Gilbert, Paul G Constantine, and Reed M Maxwell. Active subspaces for sensitivity analysis and dimension reduction of an integrated hydrologic model. *Computers & Geosciences*, 83:127–138, 2015.

Jennifer L Jefferson, Reed M Maxwell, and Paul G Constantine. Exploring the sensitivity of photosynthesis and stomatal resistance parameters in a land surface model. *Journal of Hydrometeorology*, 18(3):897–915, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Remi R Lam, Olivier Zahm, Youssef M Marzouk, and Karen E Willcox. Multifidelity dimension reduction via active subspaces. *SIAM Journal on Scientific Computing*, 42(2):A929–A956, 2020.

Ker-Chau Li. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327, 1991.

Hugo F. S. Lui and William R. Wolf. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *Journal of Fluid Mechanics*, 872:963–994, Jun 2019. ISSN 1469-7645. doi: 10.1017/jfm.2019.358. URL http://dx.doi.org/10.1017/jfm.2019.358.

Trent W. Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J. Alonso. *Active Subspaces for Shape Optimization*. doi: 10.2514/6.2014-1171. URL https://arc.aiaa.org/doi/abs/10.2514/6.2014-1171.

Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. In *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

Thomas O'Leary-Roseberry, Umberto Villa, Peng Chen, and Omar Ghattas. Derivative-informed projected neural networks for high-dimensional parametric maps governed by pdes, 2020.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85): 2825–2830, 2011. URL http://jmlr.org/papers/v12/pedregosa11a.html.

Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Sujith Ravi. Projectionnet: Learning efficient on-device deep networks using neural projections, 2017.

Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dans Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641(1):115–127, 2009.

Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

Trent Michael Russi. *Uncertainty quantification with experimental data and complex system models*. PhD thesis, UC Berkeley, 2010.

Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473 (2197):20160446, 2017.

Hayden Schaeffer, Giang Tran, and Rachel Ward. Extracting sparse high-dimensional dynamics from limited data. *SIAM Journal on Applied Mathematics*, 78(6):3279–3295, 2018.

Hayden Schaeffer, Giang Tran, Rachel Ward, and Linan Zhang. Extracting structured dynamical systems using sparse optimization with very few samples. *Multiscale Modeling & Simulation*, 18 (4):1435–1461, 2020.

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

Uri Shaham, Alexander Cloninger, and Ronald R. Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2018. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2016.04.003. URL https://www.sciencedirect.com/science/article/pii/S1063520316300033.

Yifan Sun, Linan Zhang, and Hayden Schaeffer. Neupde: Neural network based ordinary and partial differential equations for modeling time-dependent data. In *Mathematical and Scientific Machine Learning*, pages 352–372. PMLR, 2020.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016. URL http://jmlr.org/papers/v17/16-177.html.

Kailiang Wu and Dongbin Xiu. Numerical aspects for approximating governing equations using data. *Journal of Computational Physics*, 384:200–221, 2019.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103 – 114, 2017. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2017.07.002. URL http://www.sciencedirect.com/science/article/pii/S0893608017301545.

W. Zhu, Q. Qiu, J. Huang, R. Calderbank, G. Sapiro, and I. Daubechies. Ldmnet: Low dimensional manifold regularized neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2743–2751, June 2018. doi: 10.1109/CVPR.2018.00290.

Appendix A. Proofs

A.1. Lemma 1

Proof The proof generalizes the scalar case, see Lemma 2.2 from Constantine et al. (2014). For completeness, we prove it here.

For any $i \in [n]$, each component of f can be written as:

$$f_i(x) = f_i(WW^T x) = f_i(W_1 W_1^T x + W_2 W_2^T x) = f_i(W_1 y + W_2 z),$$

and so by the chain rule we have

$$\nabla_y f_i(x) = \nabla_y f_i(W_1 y + W_2 z) = W_1^T \nabla_x f_i(W_1 y + W_2 z) = W_1^T \nabla_x f_i(x).$$

It follows that $D_y f(x)^T = W_1^T D_x f(x)^T$, and similarly $D_z f(x)^T = W_2^T D_x f(x)^T$. Therefore,

$$\begin{aligned}
\sum_{i=1}^{n} \mathbb{E}\left[\nabla_y(f_i)^T \nabla_y(f_i)\right] &= \mathbb{E}\left[\operatorname{tr} D_y(f)^T D_y(f)\right] \\
&= \mathbb{E}\left[\operatorname{tr} W_1^T D_x(f)^T D_x(f) W_1\right] \\
&= \operatorname{tr} W_1^T \mathbb{E}\left[D_x(f)^T D_x(f)\right] W_1 \\
&= \operatorname{tr} W_1^T C W_1 \\
&= \operatorname{tr} \Lambda_1 \\
&= \lambda_1 + \cdots + \lambda_k
\end{aligned}$$

The same argument can be applied to the z case. ■

A.2. Theorem 2

Proof The proof generalizes Theorem 3.1 from Constantine et al. (2014), using our result in Lemma 1. For completeness, we prove it here.

Note that by Equation (3), we have that the conditional expectation is zero, i.e. $\mathbb{E}\left[f_i - g_i \circ W_1^T | y\right] = 0$ for each $i \in [n]$. Therefore, the mean-squared error is controlled by:

$$\begin{aligned}
\mathbb{E}\left[\left\|f - g \circ W_1^T\right\|_2^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\left\|f - g \circ W_1^T\right\|_2^2 | y\right]\right] && (6)\\
&\leq c\,\mathbb{E}\left[\sum_{i=1}^{n} \mathbb{E}\left[\|\nabla_z f_i\|_2^2 | y\right]\right] && (7)\\
&= c\,\mathbb{E}\left[\sum_{i=1}^{n} \mathbb{E}\left[\nabla_z f_i^T \nabla_z f_i | y\right]\right] \\
&= c \sum_{i=1}^{n} \mathbb{E}\left[\nabla_z f_i^T \nabla_z f_i\right] && (8)\\
&= c\left(\lambda_{k+1} + \cdots + \lambda_m\right) && (9)
\end{aligned}$$

where (6) and (8) are by the tower property of conditional expectation, (7) is by the Poincaré inequality, and (9) follows from Lemma 1. ■

A.3. Conditional Expectation and Minimizing Mean-Squared Error

Theorem 3 *Let $f(y, z) : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}$ with $y \in \mathbb{R}^k$, $z \in \mathbb{R}^{m-k}$ be a square integrable function with respect to a joint density function $\pi(y, z)$. Then the conditional expectation $g(y) := \mathbb{E}[f|y]$ is the minimizer of the mean-squared error associated to f. That is, $\mathbb{E}\left[(f-g)^2\right] \leq \mathbb{E}\left[(f-h)^2\right]$ where $h = h(y)$ is any function of y.*

Note that the risk defined in (2) can be rewritten as

$$\min_{g=[g_1,\cdots,g_n]^T} \sum_{i=1}^{n} \mathbb{E}\left[(f_i(W_1 y + W_2 z) - g_i(y))^2)\right]$$

where y, z are the rotated coordinates defined is Section 2.1. Theorem 3 then quickly implies that the function $g = [g_1, \ldots, g_n]^T$ with each g_i defined by (3)–i.e. the conditional expectation of f_i given y–is indeed the minimizer of the risk (2). We prove Theorem 3 now.

Proof Define the marginal density $\pi_Y(y)$ and conditional density $\pi_{Z|Y}(z|y)$ in the usual way. For any function $h = h(y)$ of y, consider its mean-squared error:

$$\begin{aligned}
\mathbb{E}\left[(f-h)^2\right] &= \mathbb{E}\left[(f - g + g - h)^2\right] \\
&= \mathbb{E}\left[(f-g)^2\right] + 2\mathbb{E}\left[(f-g)(g-h)\right] + \mathbb{E}\left[(g-h)^2\right] && (10)
\end{aligned}$$

Given the definition of g, we have that the cross term reduces to zero:

$$\begin{aligned}
\mathbb{E}\left[(f-g)(g-h)\right] &= \int\int (f(y,z) - g(y))(g(y) - h(y))\pi(y,z)\,dz\,dy \\
&= \int \left(\int (f(y,z) - g(y))\pi_{Z|Y}(z|y)\,dz\right)(g(y) - h(y))\pi_Y(y)\,dy && (11)\\
&= 0 && (12)
\end{aligned}$$

where (11) utilizes the representation $\pi(y,z) = \pi_{Z|Y}(z|y)\pi_Y(y)$, and (12) follows from the integral representation of conditional expectation:

$$\begin{aligned}\int (f(y,z) - g(y))\pi_{Z|Y}(z|y)\,dz &= \int f(y,z)\pi_{Z|Y}(z|y)\,dz - g(y)\int_z \pi_{Z|Y}(z|y)\,dz \\ &= \mathbb{E}\left[f|y\right] - g(y) \\ &= 0\end{aligned}$$

Since this and (10) hold for arbitrary $h(y)$, if follows that $\mathbb{E}\left[(f-g)^2\right] \leq \mathbb{E}\left[(f-h)^2\right]$ for any function $h = h(y)$ of y. ■