Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2021-0217

Vol. **32**, No. 1, pp. 1-40
July 2022

Nonlinear Reduced DNN Models for State Estimation

Wolfgang Dahmen[1], Min Wang[2,*] and Zhu Wang[1]

[1] *Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA.*
[2] *Department of Mathematics, Duke University, Durham, NC 27708, USA.*

Received 8 November 2021; Accepted (in revised version) 8 April 2022

Abstract. We propose in this paper a data driven state estimation scheme for generating nonlinear reduced models for parametric families of PDEs, directly providing *data-to-state maps*, represented in terms of Deep Neural Networks. A major constituent is a *sensor-induced* decomposition of a model-compliant Hilbert space warranting approximation in problem relevant metrics. It plays a similar role as in a Parametric Background Data Weak framework for state estimators based on Reduced Basis concepts. Extensive numerical tests shed light on several optimization strategies that are to improve robustness and performance of such estimators.

AMS subject classifications: 65N20, 65N21, 68T07, 35J15

Key words: State estimation in model-compliant norms, deep neural networks, sensor coordinates, reduced bases, ResNet structures, network expansion.

1 Introduction

Understanding complex "physical systems" solely through observational data is an attractive but unrealistic objective if one insists on certifiable accuracy quantification. This, in turn, is an essential precondition for prediction capability. In fact, unlike application scenarios where an abundance of data are available, data acquisition for "Physics Informed Learning Task" typically relies on sophisticated sensor technology and is often expensive or even harmful. Therefore, a central task is to develop efficient ways for fusing the information provided by data with *background information* provided by physical laws governing the observed states of interest, typically represented by partial differential equations (PDEs). In principle, this falls into the framework of "Physics Informed Neural Networks" (PINN), however, with some noteworthy distinctions explained next.

*Corresponding author. *Email addresses:* `dahmen@math.sc.edu` (W. Dahmen), `wangmin@math.duke.edu` (M. Wang), `wangzhu@math.sc.edu` (Z. Wang)

The central objective of this note is to explore a machine learning approach to state estimation in the above sense. Our contributions concern two major aspects:

(i) In contrast to typical PINN formulations, we employ loss functions that are *equivalent* to the error of the estimator in a norm that is imposed by the continuous mathematical model. More precisely, this norm corresponds to a *stable variational formulation* of the PDE family. In other words, the generalization error for this loss function measures the accuracy of the estimator in a problem intrinsic norm without imposing any additional regularity properties.

(ii) When employing estimators, represented as Deep Neural Networks (DNNs), one has to accept a significant and unavoidable uncertainty about optimization success. Due to (i), one can at least measure the achieved accuracy at any stage of the optimization. We therefore take this fact as a starting point for a systematic *computational exploration* of a simple optimization strategy that seems to be particularly natural in combination with ResNet architectures.

Regarding (i), the proposed approach is, in principle, applicable to a much wider scope of problems than discussed below. Last but not least, in order to facilitate comparisons with other recovery schemes, specifically with methods that are based on *Reduced Basis* concepts, the numerical experiments focus on elliptic families of PDEs with *parameter dependent* diffusion fields. However, for this problem class we discuss in detail two rather different scenarios, namely diffusion coefficients with an *affine* parameter dependence, as well as *log-normal* parameter dependence. It is well known that the first scenario offers favorable conditions for Reduced Basis methods which have been well studied for this type of models and can therefore serve for comparisons. While in this case nonlinear schemes using neural networks do not seem to offer decisive advantages in terms of achievable certifiable estimation accuracy nor computational efficiency we see an advantage of the DNN approach in the second scenario because it seems that they can be better adapted to the challenges of this problem class.

It should be noted though that the present approach shares some conceptual constituents with so called One-Space methods or PBDW (Parametric Background Data Weak) methods (see [2, 4, 10]). We therefore briefly recollect some related basic ideas in Section 2.4. An important element is to represent the sensor functionals as elements of the trial space $\mathbb{U}$ for the underlying PDE. The $\mathbb{U}$-orthogonal projection to their span, termed "measurement space", provides a natural "zero-order approximation" to the observed state. To obtain an improved more accurate reconstruction, the data need to be "lifted" to the complement space. We view the construction of such a "lifting map" as "learning" the expected "label" associated with a given observation, see Section 2. This, in turn, is based on first projecting "synthetic data" in terms of parameter snapshots, to the $\mathbb{U}$-orthogonal complement of the measurement space. We then extract via SVD from these projected data a sufficiently accurate "effective" complement space that captures corresponding components of the solution manifold with high accuracy in $\mathbb{U}$. The lifting map is then expressed in terms of the coefficients of a $\mathbb{U}$-orthogonal basis of the effective complement space which, in turn, are represented by a neural network. The fact that we learn the coefficients of a $\mathbb{U}$-orthogonal basis allows us to control the accuracy of the estimator in the problem-relevant norm. Thus, the proposed method is based on combining POD based model reduction with neural network regression. In this regard, it shares similar concepts with the recent work in [13] and [9]. However, a major distinction is that in [9,13] the whole parameter-to-solution map is constructed while the present work focuses on generating directly a *data-to-state* map through regression in *sensor coordinates* (see Section 2.3).

Concerning (ii), successfully training a neural network remains a serious issue. Neither can one guarantee to actually exploit the expressive power of a given network architecture, nor does it seem possible to estimate the required computational cost. Therefore, it is more important to measure a given optimization outcome in a problem-relevant metric for making it possible to enable the assessment of the estimation quality obtained in the end. In the second part of the paper we computationally explore a natural training strategy that, as will show, renders optimization more reliable, stable and robust with regard to varying algorithmic parameters like depth, width or learning rates. To counter gradient decay when increasing depth, we opt for ResNet architectures. Moreover, we systematically compare stochastic gradient descent applied to all trainable parameters, termed "plain Gl-ResNN training" to an *expansion strategy* that starts with a shallow network and successively adds further blocks in combination with a *blockwise optimization*. That is, at every stage we optimize only the trainables in a single block while freezing the remaining parameters. In that sense we do not fix any network architecture beforehand but expand it dynamically while monitoring loss-decay, see Subsection 3.2. Section 4 is then devoted to extensive numerical studies comparing plain Gl-ResNN and the expansion strategy for both application scenarios.

2 Problem formulation and conceptual preview

2.1 Parametric PDE models

For a wide scope of applications, the underlying governing laws can be represented by a family of PDEs

$$\mathcal{F}(u,\mathbf{y};f)=0, \quad \mathbf{y}\in\mathcal{Y}, \tag{2.1}$$

with data f and coefficients depending on parameters $\mathbf{y}$ ranging over some compact parameter domain $\mathcal{Y}\subset\mathbb{R}^{d_y}$. Focusing on linear problems, what matters in the present context is to identify first a *stable variational formulation* for (2.1), i.e., to identify a(n infinite-dimensional) *trial-space* $\mathbb{U}$ and a *test-space* $\mathbb{V}$, such that

$$\mathcal{F}(u,\mathbf{y};f)(v)=0=f(v)-(\mathcal{B}_{\mathbf{y}}u)(v), \quad v\in\mathbb{V}, \tag{2.2}$$

is well-posed, meaning that the bilinear form

$$b(u,v;\mathbf{y})=(\mathcal{B}_{\mathbf{y}}u)(v), \quad u\in\mathbb{U},\ v\in\mathbb{V}, \tag{2.3}$$

satisfies the conditions of the Babuška-Nečas Theorem (continuity-, inf-sup-, surjectivity-condition), [12].

When $\mathbf{y}$ traverses $\mathcal{Y}$ the respective solutions $u(\mathbf{y})$ mark the viable states of interest and form what is often called the *solution manifold*

$$\mathcal{M} := \{u(\mathbf{y}) : \mathcal{F}(u(\mathbf{y}),\mathbf{y};f) = 0,\ \mathbf{y} \in \mathcal{Y}\} \subset \mathbb{U}, \tag{2.4}$$

which is in most relevant cases a compact subset of $\mathbb{U}$. The central task considered in this work is to recover, from a moderate number of measurements, a state $u \in \mathbb{U}$ under the prior $u \in \mathcal{M}$.

Stable formulations are available for a wide scope of problems to which the following approach then applies. To be specific, we focus as a guiding example second order elliptic problems

$$\mathcal{F}(u,\mathbf{y};f) = f - \mathcal{B}_{\mathbf{y}}u := f + \operatorname{div}(a(\cdot;\mathbf{y})\nabla u), \quad \text{in } \Omega \subset \mathbb{R}^{d_x}, \quad u|_{\partial\Omega} = 0, \tag{2.5}$$

where f is a forcing term. Examples are Darcy's equation for the pressure in groundwater flow or electron impedance tomography. Both involve second order elliptic equations as core models and *parameter dependent* diffusion coefficients that describe permeability or conductivity, respectively, where in the latter case the model is complemented by Robin-type boundary conditions. A large parameter dimension d_y reflects substantial model complexity. In a probabilistic framework, the parametric representations of the coefficients could arise, for instance, from Karhunen-Loève expansions of a random field that represent numerically "unresolvable" features. In this case the number of parameters is ideally *infinite* and parameter truncation causes additional model bias.

For (2.5) a proper choice of trial- and test-spaces is $\mathbb{U} = \mathbb{V} = H_0^1(\Omega)$, provided that one has *uniform ellipticity*, namely there exist constants $0 < r \le R < \infty$ such that

$$r \le a(\cdot,\mathbf{y}) \le R, \quad \text{in } \Omega, \mathbf{y} \in \mathcal{Y}. \tag{2.6}$$

For other problem types, such as convection dominated problems or time-space formulations for parabolic problems, one may have to choose $\mathbb{V}$ differently from $\mathbb{U}$, see e.g. [7,8].

Later in our numerical examples we consider two types of diffusion coefficients:

(S1) *Piecewise constant coefficients:* Given $\{\Omega_j\}_{j\in I}$ a non-overlapping partition of Ω, $\bigcup_{j\in I}\Omega_j = \Omega$, the diffusion coefficient in the model (2.5) is assumed to be of the following form:

$$a(x;\mathbf{y}) = a_0(x) + \sum_{j\in I} y_j \chi_j(x), \tag{2.7}$$

where y_j, the j-th component of $\mathbf{y}$, obeys a uniform distribution and $\chi_j(x)$ is the characteristic function that takes the values 1 in Ω_j and 0 in its complement $\Omega\setminus\Omega_j$. This corresponds to the affine parameter representation (2.7). We assume for simplicity $a_0(x) = 1$ and $y_j \sim U[-1/2,1/2]$ so that the diffusion coefficient is piecewise constant while the uniform ellipticity (2.6) of the problem is guaranteed with a moderate condition of the variational formulation.

(S2) *Lognormal case:* Suppose the diffusion coefficient has the following parameterized form:

$$a(x;\mathbf{y}) = a_0(x) + a_1 e^{z(x,\mathbf{y})}, \tag{2.8}$$

where a_0 is a continuous non-negative function on $\overline{\Omega}$, a_1 is a positive constant and $z(x,\mathbf{y})$ is a zero-mean Gaussian random field

$$z(x,\mathbf{y}) = \sum_{j=1}^{\infty} \sqrt{\mu_j}\xi_j(x)\eta_j(\mathbf{y}), \quad x \in \Omega \quad \text{and} \quad \mathbf{y} \in \mathcal{Y}.$$

Here, $\{\eta_j\}_{j\geq 1}$ form an orthogonal system over $\mathcal{Y}$ where the components of $\mathbf{y}$ are i.i.d. $\mathcal{N}(0,1)$, and the sequence $\{(\mu_j,\xi_j)\}_{j\geq 1}$ are real eigenpairs of the covariance integral operator

$$(Cv)(x) = \int_D c(x,x')v(x')dx'$$

associated with the Matérn model

$$c(x,x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (2\sqrt{\nu}r)^{\nu} K_{\nu}(2\sqrt{\nu}r),$$

in which σ^2 is the marginal variance, $\nu > \frac{1}{2}$ is the smoothness parameter of the random field, Γ is the Gamma function,

$$r = \sqrt{\frac{(x_1 - x_2')^2}{\lambda_{x_1}^2} + \frac{(x_2 - x_2')^2}{\lambda_{x_2}^2}},$$

λ_{x_1} and λ_{x_2} are the correlation lengths along x_1- and x_2-coordinates, and K_ν the modified Bessel function of second kind.

At the discrete level, to generate realizations of the stationary Gaussian process over the grid points of Ω, we use the circulant embedding approach in [14,17]. Its main idea is to embed the covariance matrix to a block circulant matrix so that FFT can be applied for a fast evaluation.

The reason for considering these two scenarios lies in the following principal distinctions: In scenario (S1) the choice of $\mathcal{Y}$ ensures that the diffusion parameter always satisfies (2.6) so that

$$a(u,v;\mathbf{y}) := \int_\Omega a(\cdot;\mathbf{y}) \nabla u \cdot \nabla v dx = f(v), \quad v \in \mathbb{U}, \tag{2.9}$$

is stable over $\mathbb{U} = H_0^1(\Omega) = \mathbb{V}$ and possesses for each $\mathbf{y} \in \mathcal{Y}$ a unique weak solution $u = u(\mathbf{y}) \in \mathbb{U}$. However, the condition of (2.9) depends on R/r and deteriorates when this quotient grows. In scenario (S2) this latter aspect is aggravated further because (2.6) holds only with high probability. Moreover, in contrast to scenario (S1), the diffusion coefficient no longer is affine in $\mathbf{y} \in \mathcal{Y}$ which is known to pose challenges to the construction of certifiable Reduced Bases, see e.g. [11].

2.2 Sensors and data

In addition to the model assumption that an observed state $u \in \mathbb{U}$ (nearly-) belongs to $\mathcal{M}$ we wish to utilize external information in terms of measurements or data. Throughout this note we will assume that the number m of measurements $\mathbf{o} = (o_1, \cdots, o_m)^\top \in \mathbb{R}^m$ of an unknown state u is fixed and of moderate size. We will always assume that the data are produced by *sensors*

$$o_i = \ell_i(u), \quad i = 1, \cdots, m, \quad \boldsymbol{\ell} := (\ell_1, \cdots, \ell_m) \in (\mathbb{U}')^m, \tag{2.10}$$

i.e., we assume, for simplicity, in what follows the ℓ_i to be *bounded linear functionals*.

Estimating u from such data is ill-posed, already due to a possible severe under-sampling. Under the assumption that u (at least nearly) satisfies the PDE for *some* parameter $\mathbf{y} \in \mathcal{Y}$ one may also ask for such a parameter $\mathbf{y}$ that explains the data best. Since the *parameter-to-solution map* $\mathbf{y} \to u(\mathbf{y})$ is not necessarily injective, this latter parameter estimation problem is typically even more severely ill-posed and nonlinear.

A common approach to regularize both estimation tasks is *Bayesian inversion*. An alternative is to fix from the start a presumably good enough discretization, say in terms of a large finite element space $\mathbb{U}_h$ to then solve (a large scale optimization) problem

$$\min_{\mathbf{y} \in \mathcal{Y}, u_h \in \mathbb{U}_h} \left\{ \|\mathbf{o} - \boldsymbol{\ell}(u_h)\|^2 + \lambda \|\mathcal{F}_h(u_h, \mathbf{y}; f)\|^2 + \mu \|R_h(u_h)\|^2 \right\}, \tag{2.11}$$

where $\|\cdot\|$ is just the Euclidean norm, λ and μ are weight parameters and the last summand represents a regularization term which is needed since $d_y + \dim \mathbb{U}_h$ is typically much larger than the number of measurements m. Questions arising in this context are: How to choose λ, μ and the regularization operator R_h; perhaps, more importantly, should one measure deviation from measurements and the discrete residual – closeness to the model – in the *same* metric?

Moreover, for each new data instance $\mathbf{o}'$ one has to solve the same large-scale problem again. Thus, employing a *reduced model* [15,16] for approximating the *parameter-to-solution map* $\mathbf{y} \mapsto u(\mathbf{y})$ would serve two purposes, namely mitigating under-sampling and speeding forward simulations. Specifically, we apply similar concepts as in [2, 4, 6, 10] to base state estimation on *reduced modeling*, as described next.

2.3 Sensor-coordinates

Rather than tying a discretization directly to the estimation task, and hence to a specific regularization, as in (2.11), reduced order modeling methods such as the Reduced Basis Methods (RBM) first prepare in an *offline phase* a reduced order model that requires the bulk of computation. It typically takes place in a "truth-space" $\mathbb{U}_h \subset \mathbb{U}$, usually a finite element space of sufficiently large dimension that is expected to comply with envisaged estimation objectives (which could be even adjusted at a later stage). In particular, the reduced order model can be adapted to the solution manifold and the sensor system. As such it is only used indirectly in the recovery process and should be viewed as representing "computing in $\mathbb{U}$". A first important ingredient of the methods in [2, 4, 6, 10] is to *Riesz-lift* the functionals in $\boldsymbol{\ell}$ from $\mathbb{U}'$ to $\mathbb{U}$, thereby subjecting them to the same metric as the states u. The obtained Riesz-representers $\phi_i \in \mathbb{U}$ of $\ell_i \in \mathbb{U}'$ then span an m-dimensional subspace $\mathbb{W} \subset \mathbb{U}$, referred to as *measurement space*. Thus, $\mathbb{U}$-orthogonal projections $P_{\mathbb{W}}u$ of some $u \in \mathbb{U}$ to $\mathbb{W}$ encode the same data-information about u as $\mathbf{o} = \boldsymbol{\ell}(u)$. This induces the decomposition

$$\mathbb{U} = \mathbb{W} \bigoplus \mathbb{W}^{\perp}, \quad u = P_{\mathbb{W}}u + P_{\mathbb{W}^{\perp}}u, \tag{2.12}$$

so to speak representing any state u in "sensor coordinates" $w = P_{\mathbb{W}}u$, and "labels" $w^{\perp} = P_{\mathbb{W}^{\perp}}u$.

In these terms, recovering a state u from its measurement $w = P_{\mathbb{W}}u$, means to approximate the label $P_{\mathbb{W}^{\perp}}u \in \mathbb{W}^{\perp}$. Thus, the state estimation can be viewed as seeking a map

$$A : w \mapsto A(w), \quad A(w) = w + B(w), \tag{2.13}$$

where, in principle, $B : \mathbb{W} \to \mathbb{W}^{\perp}$ could be any map that hopefully exploits the fact that $w = P_{\mathbb{W}}u$ for some $u \in \mathcal{M}$ in an effective way.

2.4 Affine recovery map

The methods in [2, 4, 10] determine the *lifting-map* $B : \mathbb{W} \to \mathbb{W}^{\perp}$ as a *linear* or *affine* map, termed as "One-Space-Methods", and in [6] as a *piecewise affine* map combining One-Space concepts with parameter domain decomposition and model selection. As shown in [4], any affine map B is characterized by an *affine* subspace $\mathbb{U}_n = \bar{u} + \widetilde{\mathbb{U}}_n \subset \mathbb{U}$, where $\bar{u}$ is a suitable chosen offset state and $\widetilde{\mathbb{U}}_n$ is a linear space of dimension n, for which the estimator $A = A_{\mathbb{U}_n}$ satisfies

$$u^*(w) := A_{\mathbb{U}_n}(w) = \operatorname{argmin}\{u \in w + \mathbb{W}^{\perp} : \|u - P_{\mathbb{U}_n}u\|_{\mathbb{U}}\}. \tag{2.14}$$

Moreover, $u^*(w)$ can be computed efficiently as a linear least-squares problem in $\mathbb{U}_n$ followed by a simple correction in $\mathbb{W}$. Uniqueness is ensured if

$$\mu(\widetilde{\mathbb{U}}_n, \mathbb{W}) := \sup_{v \in \mathbb{U}_n} \frac{\|v\|_{\mathbb{U}}}{\|P_{\mathbb{W}}v\|_{\mathbb{U}}} < \infty, \tag{2.15}$$

which is the case if and only if $\mathbb{U}_n \cap \mathbb{W}^{\perp} = \{0\}$. $\mu(\widetilde{\mathbb{U}}_n, \mathbb{W})$ is actually computable as one over the smallest singular value of the cross-Gramian of a basis for $\mathbb{U}_n$ and $\mathbb{W}$. Hence, it has a geometric interpretation because it relates to the *angle* between the spaces $\widetilde{\mathbb{U}}_n$ and $\mathbb{W}$, tending to infinity when this angle approaches $\pi/2$. This affects estimation accuracy directly since

$$\sup_{u \in \mathcal{M}} \|u^*(P_{\mathbb{U}_n}u) - u\|_{\mathbb{U}} \le \mu(\widetilde{\mathbb{U}}_n, \mathbb{W})\varepsilon_n, \quad \text{when} \quad \operatorname{dist}(\mathcal{M}, \mathbb{U}_n)_{\mathbb{U}} := \sup_{u \in \mathcal{M}} \inf_{z \in \mathbb{U}_n} \|u - z\|_{\mathbb{U}} \le \varepsilon_n. \tag{2.16}$$

This is actually best possible when using, as sole prior, the knowledge that $u\in\mathcal{M}$ also belongs to the convex set $\mathcal{K}(\mathbb{U}_n,\varepsilon_n):=\{u\in\mathbb{U}:\inf_{z\in\mathbb{U}_n}\|u-z\|_{\mathbb{U}}\le\varepsilon_n\}$, [2]. Thus, overall estimation accuracy involves a competition between the approximation property of the space $\mathbb{U}_n$ and "visibility" from $\mathbb{W}$. In fact, $\mathbb{U}_n$ can have at most dimension $n=m$ since otherwise $\mu(\widetilde{\mathbb{U}}_n,\mathbb{W})=\infty$. Hence, the rigidity of an affine space severely limits estimation quality. In fact, it is shown in [6] that restricting to such convex priors will generally fail to meet natural estimation benchmarks which calls for employing *nonlinear reduced models*.

In this regard, the following comments will provide some orientation. One-space-methods appear to nevertheless perform very well for problems where the solution manifold $\mathcal{M}$ has rapidly decaying Kolmogorov n-widths

$$d_n(\mathcal{M})_{\mathbb{U}}:=\inf_{\dim\mathbb{V}_n\le n}\operatorname{dist}(\mathcal{M},\mathbb{V}_n)_{\mathbb{U}}. \tag{2.17}$$

For elliptic problems (2.5) this is known to be the case, even in high parameter dimensional regimes, when (2.6) holds and the diffusion coefficients depend *affinely* on the parameters, as is the case in scenario (S1), (2.7). Approximations from judiciously chosen linear spaces can then be very effective. Moreover, affine parameter-dependence is also instrumental in the *greedy* construction methods of Reduced Bases for rate-optimal reduced linear models which play a central role in one-space-methods, [1, 5–7, 11]. It is therefore interesting to see how the nonlinear estimators proposed in the present paper compare with affine recovery methods in scenarios where the latter are known to perform well. For this reason we include scenario (S1) in our numerical tests.

That said, the effectivity of one-space-methods depends in a rather sensitive way on the above favorable conditions, namely rapid decay of n-widths, uniform ellipticity, and affine parameter dependence. We therefore include scenario (S2) with log-normal parameter dependence. In fact, affine dependence no longer holds, the behavior of n-widths is much less clear, and the coefficient field may nearly degenerate, challenging the validity of (2.6).

2.5 A regression framework

To avoid monitoring pointwise errors in high-dimensions we opt now for a mean-square accuracy quantification which responds in a less sensitive way to high parameter dimensionality and blends naturally into a learning context. A natural probabilistic model could be based on a probability measure μ on $\mathbb{U}$ with support on $\mathcal{M}$. $u=(w=P_{\mathbb{W}}u,\ w^\perp=P_{\mathbb{W}^\perp}u)$, is then viewed as a random variable. The optimal estimator would then be the nonlinear map

$$A^*(w)=w+B^*(w), \tag{2.18}$$

where the conditional expectation $B^*(w)=\int_{w+\mathbb{W}^\perp}P_{\mathbb{W}^\perp}u\,d\mu(u|w)$ is the *regression function* minimizing

$$\int_{\mathbb{U}}\|u-A(P_{\mathbb{W}}u)\|_{\mathbb{U}}^2d\mu(u)=\int_{\mathbb{W}}\int_{w+\mathbb{W}^\perp}\|w^\perp-B(w)\|_{\mathbb{U}}^2d\mu(u|w)d\mu(w) \tag{2.19}$$

over all mappings of the form $A(w)=w+B(w)$, $B:\mathbb{W}\to\mathbb{W}^\perp$.

The central goal in what follows is to construct numerical estimators that approximate B^* well. Note that this approximation should take place in $\mathbb{U}$ for the estimator to respect the natural problem metrics.

3 State estimation algorithm

3.1 Computational setting

We describe next in more details how to set up a learning problem in the problem compliant norm $\|\cdot\|_{\mathbb{U}}$. We adhere to the sensor-induced decomposition (2.12). Of course, the underlying Riesz-lifts of the measurement functionals $\boldsymbol{\ell}$ cannot be computed exactly but need to be approximated within controllable accuracy. In the spirit of Reduced Basis methodology, we employ a sufficiently large "truth space" $\mathbb{U}_h\subset\mathbb{U}$, which we choose here as a conforming finite element space. The scheme is based on the following steps.

(1) The Measurement Space: As a major part of the offline stage, one then solves the m

Galerkin problems

$$(\tilde{\phi}_h^i,v_h)_{\mathbb{U}}=\ell_i(v_h),\quad \forall v_h\in\mathbb{U}_h,\quad \text{and}\quad i=1,\cdots,m, \tag{3.1}$$

providing (approximate) *Riesz representers* $\tilde{\phi}_h^i\in\mathbb{U}_h$ of the linear functionals ℓ_i. Then define

$$\mathbb{W}_h:=\texttt{span}\{\phi_h^1,\cdots,\phi_h^m\}, \tag{3.2}$$

where the $\{\phi_h^1,\cdots,\phi_h^m\}$ result from orthonormalizing the lifted functionals $\{\tilde{\phi}_h^1,\cdots,\tilde{\phi}_h^m\}$. Hence,

$$P_{\mathbb{W}_h}u:=\sum_{i=1}^m(u,\phi_h^i)_{\mathbb{U}}\phi_h^i, \tag{3.3}$$

is the orthogonal projector from $\mathbb{U}_h$ onto $\mathbb{W}_h$ that encodes the information provided by the sensor system as elements of the trial space $\mathbb{U}_h$.

Specifically, given an observation vector $\boldsymbol{\ell}(u)\in\mathbb{R}^m$ of some observed state $u\in\mathbb{U}$, we can determine $P_{\mathbb{W}_h}(u)$ as follows. Let $\tilde{\boldsymbol{\Phi}}$, $\boldsymbol{\Phi}\in\mathbb{U}_h^m$ denote the column vectors, obtained by lining up the functions $\tilde{\phi}_h^i,\phi_h^i$, $i=1,\cdots,m$. Moreover let $\mathbf{C}\in\mathbb{R}^{m\times m}$ denote the (lower triangular) matrix that realizes the change of bases $\mathbf{C}\tilde{\Phi}=\Phi$. One readily checks that

$$\mathbf{w}:=\mathbf{C}\ell(u_h)=\left((u_h,\phi_h^1)_{\mathbb{U}},\cdots,(u_h,\phi_h^m)_{\mathbb{U}}\right)^\top, \tag{3.4}$$

i.e., the $\mathbb{U}$-orthogonal projection of u to the measurement space $\mathbb{W}_h$ is given by

$$P_{\mathbb{W}_h}u=\sum_{i=1}^m w_i\phi_h^i=:\boldsymbol{\Phi}^\top\mathbf{w}, \tag{3.5}$$

providing the best approximation to an observed state from $\mathbb{W}_h$. This can therefore be viewed already as a "zero-order" reconstruction of the observed state in $\mathbb{W}_h \subset \mathbb{U}$.

Note that the problems (3.1) are always *elliptic* Galerkin problems, regardless of the nature of (2.1). The Riesz-lift (as a mapping from $\mathbb{U}'$ to $\mathbb{U}$) has condition number equal to one. Therefore, these preparatory offline calculations are stable and often a posteriori error estimates allow one to assess the accuracy of Riesz-lifts, controlled by the choice of the truth space $\mathbb{U}_h \subset \mathbb{U}$. This should not be confused with the accuracy of $P_{\mathbb{W}_h} u$ with respect to u which is primarily limited by the number and type of sensors.

So far, we have ignored noise of the data $\mathbf{o} = \boldsymbol{\ell}(u)$. Noise in the observation vector $\mathbf{o}$ carries over to noise in the coefficients $\mathbf{w}$ from (3.4) possibly inflated by the condition of the transformation matrix $\mathbf{C}$, see [6] for a more detailed discussion of this issue.

Our subsequent numerical experiments refer to the model problem (2.5) where the sensors are given by the local averages of u over subdomains $B_i \subset \Omega$

$$\ell_i(u(x,\mathbf{y})) = \frac{1}{|B_i|} \int_{B_i} u(x,\mathbf{y})\, dx, \tag{3.6}$$

with local neighborhoods B_i, and where the diffusion coefficients are piecewise constants on a 4×4 checkerboard partition of Ω, see (2.7). Thus the parametric dimension is 16.

Specifically, in subsequent experiments we realize those by taking the average value of the FE solutions at the four points of a small square B_i enclosing the sensor location. Denoting these four points by $\{x_{p_j}\}_{j=1}^4$, we have

$$\ell_i(u_h(\mathbf{y})) = \frac{1}{4} \sum_{j=1}^{4} u_h(x_{p_j},\mathbf{y}), \tag{3.7}$$

entering the right hand side of (3.1). A numerical illustration of the selected normalized basis, $\phi_h^1, \phi_h^6, \phi_h^{11}, \phi_h^{16}$, is given in Fig. 1, where the normalized basis vectors are obtained by the SVD of the matrix formed by the finite element coefficient vectors of the lifted functionals with respect to the inner product weighted by $(v_h, v_h)_{\mathbb{U}}$.

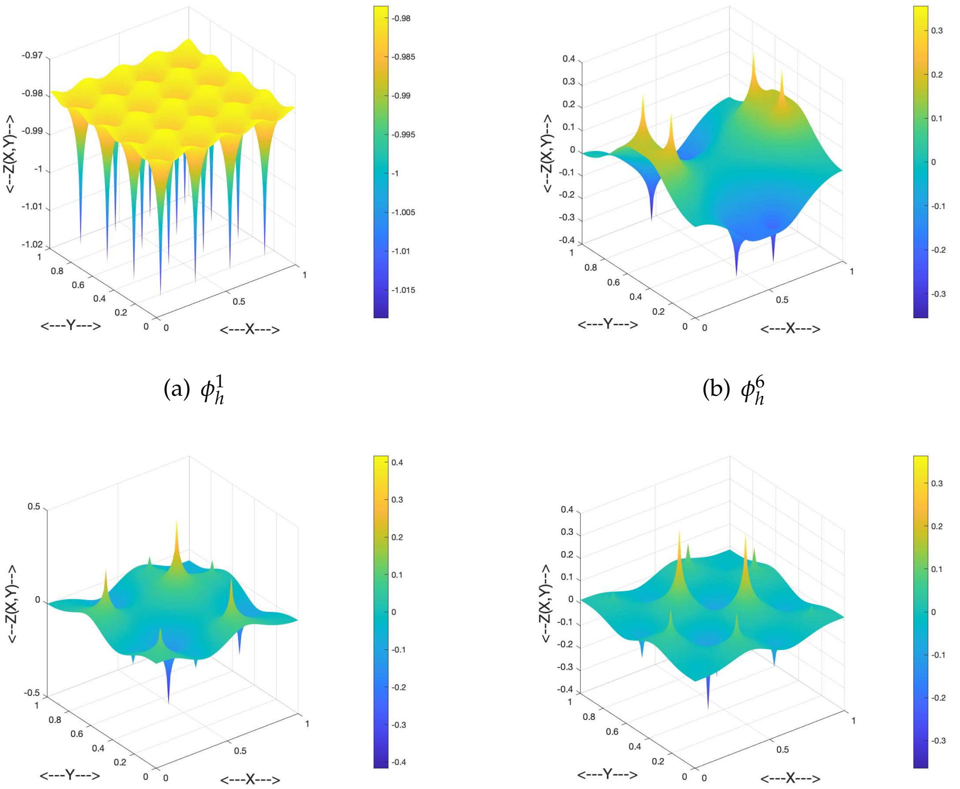
(2) Generating Synthetic Data: We randomly pick a set of parameter samples $\mathbf{y}^s$, for $s = 1, \cdots, \widehat{N}$, and employ a standard solver to compute the FE solutions $u_h^s = u_h(x;\mathbf{y}^s) \in \mathbb{U}_h$ as "snapshots" at the selected parameter values. These FE solutions provide the high-fidelity "truth" data to be later used for training the estimator towards minimizing the regression objective (2.19). Corresponding synthetic measurements then take the form

$$\mathbf{w}^j = (w_{h,1}^j, \cdots, w_{h,m}^j)^\top, \quad w_h^j := P_{\mathbb{W}_h} u_h^j = \sum_{i=1}^{m} w_{h,i}^j \phi_h^i, \quad j = 1, \cdots, \widehat{N}, \tag{3.8}$$

as detailed by (3.4) and (3.5).

In addition we need the "training labels" providing complement information

$$z_h^s = (I - P_{\mathbb{W}_h}) u_h^s, \quad s = 1, \cdots, \widehat{N}. \tag{3.9}$$



(a) ϕ_h^1

(b) ϕ_h^6

(c) ϕ_h^{11}

(d) ϕ_h^{16}

Figure 1: Example basis functions of the measurement space $\mathbb{W}_h$.

(3) Approximate $\mathbb{W}_h^\perp$: To extract the essential information provided by z_h^s, we perform next a singular value decomposition to the resulting point-cloud of finite-element coefficient vectors $\mathbf{z}_h^s \in \mathbb{R}^{N_h}$. Suppose we envisage an overall estimation target tolerance $\eta > 0$. We then retain only those k left singular vectors corresponding to singular values larger than or equal to a value $\tilde{\eta}$ which is typically less than η, for the following reason. First, incidentally, the SVD and the decay of singular values indicate whether the complement information of $\mathcal{M}$ can be adequately captured by a linear space of acceptable size within some target tolerance. While the H^1-norms of the z_h^s are uniformly bounded, the coefficients $\mathbf{z}_h^s$ in their respective finite element representation convey accuracy only in L_2, and so does the truncation of the singular values. Strictly speaking, employing standard inverse inequalities, one should take $\tilde{\eta} \le h\eta$, where h is the mesh-size in $\mathbb{U}_h$.

We next $\mathbb{U}$-orthonormalize the finite element functions corresponding to the retained left singular vectors, arriving at a $\mathbb{U}$-orthonormal basis $\mathbf{\Psi} = \{\psi_h^1, \cdots, \psi_h^k\} \subset \mathbb{W}_h^\perp$. We then define

$$\widetilde{\mathbb{W}}_h^{\perp} := \mathtt{span}\{\psi_h^1,\cdots,\psi_h^k\} \tag{3.10}$$

as our *effective complement space* that accommodates the training labels c_h^s given by

$$c_h^s(x) := P_{\mathbb{W}_h^{\perp}} u_h^s = \sum_{i=1}^{k} c_i^s \psi_h^i = \mathbf{\Psi}^\top \mathbf{c}^s, \quad c_i^s := (u_h^s, \psi_h^s)_{\mathbb{U}}, \quad s = 1,\cdots,\widehat{N}. \tag{3.11}$$

Note that c_h^s is essentially a compression of z_h^s.

In brief, after projecting the snapshot data into the complement of the measurement space, we determine a set of orthogonal basis ψ_h^j that optimally approximates the data in the sense that

$$\min_{\{\psi_h^1,\dots,\psi_h^k\}} \frac{1}{\widehat{N}} \sum_{s=1}^{\widehat{N}} \left\| z_h^s - \sum_{j=1}^{k} \left(z_h^s, \psi_h^j \right)_{\mathbb{U}} \psi_h^j \right\|_{\mathbb{U}}^2$$

subject to the conditions $(\psi_h^j, \psi_h^i)_{\mathbb{U}} = \delta_{i,j}$, $1 \le i,j \le k$, where $\delta_{i,j}$ is the Kronecker delta function. A numerical illustration of the normalized basis of $\widetilde{\mathbb{W}}_h^{\perp}$, $\psi_h^1, \psi_h^6, \psi_h^{11}, \psi_h^{16}$, is shown in Fig. 2.

Remark 3.1. Approximation in $\mathbb{U}$ is now realized by training coefficient vectors in the Euclidean norm of $\mathbb{R}^k$ because

$$\Big\| \sum_{j=1}^{k} a_j \psi_h^j \Big\|_{\mathbb{U}}^2 = \sum_{j=1}^{k} a_j^2 =: \|\mathbf{a}\|_2^2, \quad \mathbf{a} \in \mathbb{R}^k. \tag{3.12}$$

In other words, the estimator respects the intrinsic problem metrics which is a major difference from common approaches involving neural networks.

As argued earlier, we wish to construct a nonlinear map that recovers from data $w \in \mathbb{W}_h$ a state $w + B(w) \in w + \widetilde{\mathbb{W}}_h^{\perp}$ for an appropriate $B: \mathbb{W}_h \to \widetilde{\mathbb{W}}_h^{\perp}$. More specifically, for any $w_h = \sum_{i=1}^{m} w_i \phi_h^i =: \mathbf{w}^\top \mathbf{\Phi} \in \mathbb{W}_h$, the envisaged mapping B has the form

$$B(w_h) = \mathbf{w}_\perp(\mathbf{w})^\top \mathbf{\Psi}, \tag{3.13}$$

where

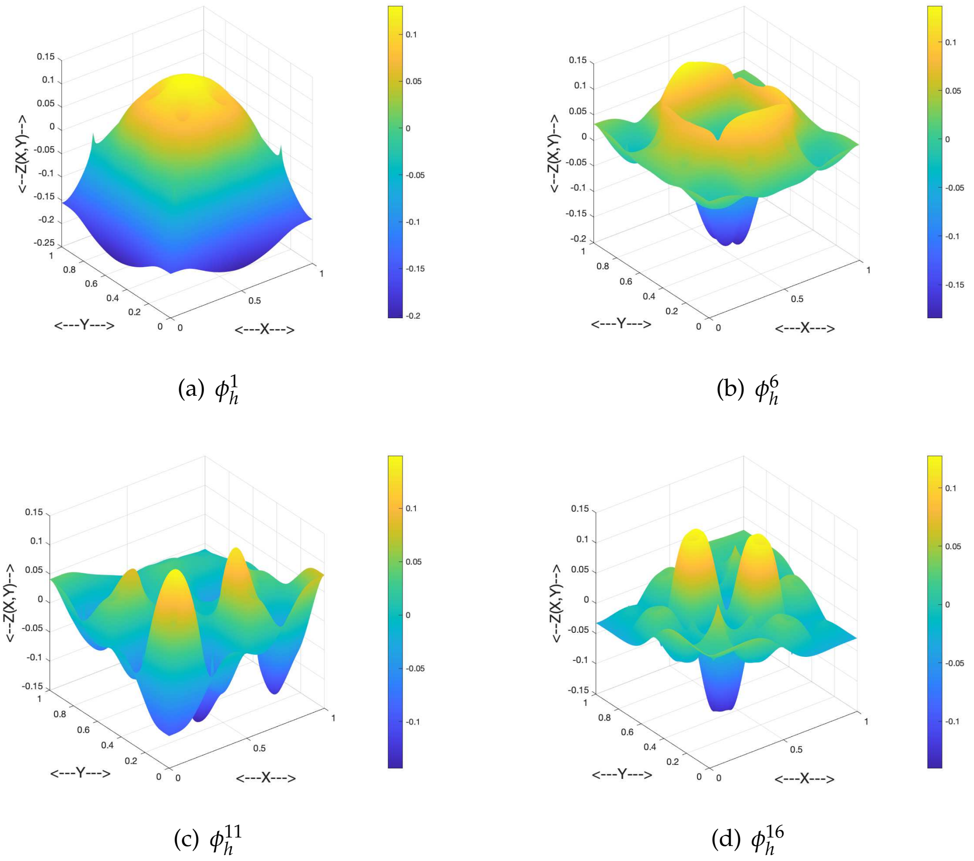
$$\mathbf{w}_\perp(\mathbf{w}) = (w_{\perp,h}^1(\mathbf{w}),\cdots,w_{\perp,h}^k(\mathbf{w}))^\top \in \mathbb{R}^k, \tag{3.14}$$

and

$$\mathbf{w}_\perp(\mathbf{w}) = \mathcal{N}\mathcal{N}(\mathbf{w}) \tag{3.15}$$

will be represented as a neural network with input data $\mathbf{w} \in \mathbb{R}^m$ and output dimension k.

(4) Loss Function and Training the Neural Network (NN): We randomly select a subset $\mathcal{U}_{train} = \{u_h^1,\cdots,u_h^{N_{train}}\}$ from the "truth" FE data $\mathcal{U}_{sample} = \{u_h^s : s = 1,\cdots,\widehat{N}\}$, generated in step (1), for training purpose and leave the rest $\mathcal{U}_{ghost} := \mathcal{U}_{sample} \backslash \mathcal{U}_{train}$ for testing. The



(a) ϕ_h^1

(b) ϕ_h^6

(c) ϕ_h^{11}

(d) ϕ_h^{16}

Figure 2: Example basis functions of the complement space $\widetilde{\mathbb{W}}_h^{\perp}$.

associated coefficient vectors $\{\mathbf{w}^1,\cdots,\mathbf{w}^{N_{train}}\}$ and $\{\mathbf{c}^1,\cdots,\mathbf{c}^{N_{train}}\}$ are used as the training data for minimizing the natural empirical loss analogous to (2.19)

$$\mathtt{Loss}=R(\Theta):=\frac{1}{N_{train}}\sum_{t=1}^{N_{train}}\|\mathbf{c}^t-\mathcal{NN}(\mathbf{w}^t;\Theta)\|_2^2, \tag{3.16}$$

where we assume in what follows that the $\mathcal{NN}$ depends on a collection Θ of *trainable* or *hyper-parameters*, over which $\mathtt{Loss}$ is to be minimized.

Once the training of $\mathcal{NN}$ is completed, for any new measurement $\boldsymbol{\ell}(u(\cdot,\mathbf{y}))$ with coefficient vector $\mathbf{w}\in\mathbb{R}^m$, defined by (3.4), (3.5), the observed state $u(\cdot,\mathbf{y})$ is approximated by

$$u\approx\mathbf{\Phi}_h^{\top}\mathbf{w}+\mathbf{\Psi}_h^{\top}\mathcal{NN}(\mathbf{w}).$$

A natural question would be now to analyze the performance of the estimator obtained by minimizing the loss. This could be approached by employing standard machine learning concepts like Rademacher complexity in conjunction with more specific assumptions on the network structure and the solution manifold. We postpone this task to forthcoming work and address in the remainder of this paper instead a more elementary issue, namely "optimization success" which is most essential for a potential merit of the proposed schemes.

3.2 A closer look at step (4)

It is a priori unclear which specific network architecture and which budget of trainable parameters is appropriate for the given recovery problem. Even if such structural knowledge were available, it is not clear whether the expressive potential of a given network class can be exhausted by the available standard optimization tools largely relying on stochastic gradient descent (SGD) concepts. Aside from formulating a model-compliant regression problem, our second primary goal is to explore numerical strategies that, on the one hand, render optimization stable and less sensitive on algorithmic settings whose most favorable choice is usually not known in practice. On the other hand, we wish to incorporate and test some simple mechanisms to *adapt* the network architecture, inspired by classical "nested iteration" concepts in numerical analysis. Corresponding simple mechanisms can be summarized as follows: (a) A ResNet architecture with its skip-connections is known to mitigate "gradient damping" which impedes the adjustment of trainable parameters in lower blocks when using deep networks. (b) Each block in a ResNet can be viewed as a perturbation of the identity and may therefore be expected to support a stable incremental accuracy upgrade. (c) Instead of "plain training" of an ResNet, where a stochastic gradient descent is applied to all trainable parameters simultaneously, we study an iterative training strategy that optimizes only single blocks at a time while freezing trainable parameters in all other blocks. Part of the underlying rationale is that training a shallow network is more reliable and efficient than training a deep network. We discuss these issues next in more detail.

Neural Network Setting: Specifically, we consider the following *Residual Neural Network* (ResNet) structure with i blocks:

$$\mathcal{N}\mathcal{N}^{[i]}(\mathbf{x};\theta^{[0]},\theta^{[1]},\cdots,\theta^{[i]}):=\mathcal{N}^{[i]}\left(\mathcal{N}\mathcal{N}^{[i-1]}(\mathbf{x};\theta^{[0]},\theta^{[1]},\cdots,\theta^{[i-1]});\theta^{[i]}\right), \tag{3.17}$$

with each block defined by

$$\begin{aligned}
&\mathcal{N}^{[1]}(\mathbf{x};\theta^{[0]},\theta^{[1]}):=W_3^{[1]}\sigma\big(W_2^{[1]}\sigma(W_1^{[1]}\mathbf{x}+b_1^{[1]})+b_2^{[1]}\big)+W^{[0]}\mathbf{x},\\
&\mathcal{N}^{[i]}(\mathbf{x},\theta^{[i]}):=W_3^{[i]}\sigma\big(W_2^{[i]}\sigma(W_1^{[i]}\mathbf{x}+b_1^{[i]})+b_2^{[i]}\big)+\mathbf{x},\quad i=2,3,\cdots,
\end{aligned}$$

where $\theta^{[i]}=\{W_1^{[i]},W_2^{[i]},W_3^{[i]},b_1^{[i]},b_2^{[i]}\}$ for $i=1,2,\cdots$ and $\theta^{[0]}=\{W^{[0]}\}$, $\sigma(\cdot)$ is some pointwise nonlinear function. In the numerical experiments of Section 4, we specifically take the activation function σ to be tanh. See Fig. 3 for an descriptive diagram of the ResNet structure.

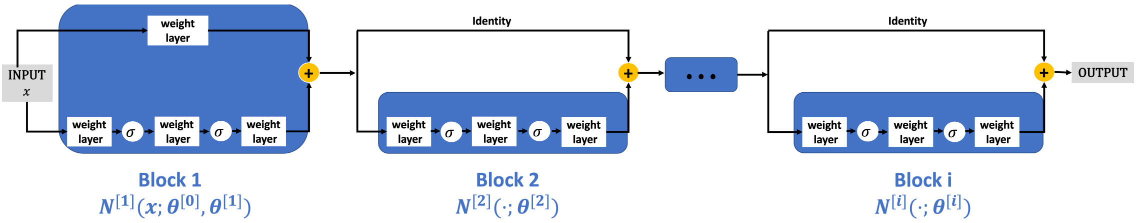


Figure 3: ResNet with i blocks.

An Expansion Strategy: As indicated earlier, we will not approximate the recovery map using a fixed network architecture. We rather start with training a single hidden layer until saturation. We then proceed approximating residual data with another shallow network, and iterate this process which results in a ResNet structure. The basic rationale is similar to "full multigrid" or "nested iteration" in numerical analysis.

To describe the procedure in more detail, let

$$\mathcal{D}:=\{\mathbf{w}^i : i=1,\cdots,N^{train}\}\times\{\mathbf{c}^i : i=1,\cdots,N^{train}\}$$

denote the set of training data, as described before. In view of (3.12), we employ simple mean-squared loss functions. In these terms we may rewrite (3.16) as

$$\mathcal{L}_{\mathcal{D}}[f]:=\frac{1}{\#\mathcal{D}}\sum_{(\mathbf{w},\mathbf{c})\in\mathcal{D}}\|\mathbf{c}-f(\mathbf{w})\|_2^2. \tag{3.18}$$

In these terms, the procedure can be described as follows: At the initial step f is taken as a shallow network $\mathcal{NN}^{[1]}(\cdot;\theta^{[0]},\theta^{[1]})$ and solve first the optimization problem:

$$\textbf{OP-1:}\qquad \min_{(\theta^{[0]},\theta^{[1]})\in\Theta^{[1]}}\quad \mathcal{L}_{\mathcal{D}}[\mathcal{NN}^{[1]}(\cdot;\theta^{[0]},\theta^{[1]})],$$

where $\Theta^{[1]}$ denotes the budget of hyper-parameters used at this first stage, encoding in particular, the widths. **OP-1** is treated with SGD-based optimizers until the loss stagnates or reaches a (local) minimum. The resulting hyper-parameters are denoted by $(\theta_*^{[0]},\theta_*^{[1]})$.

In a next step, we expand the neural network by introducing a new block and solve

$$\textbf{OP-2:}\qquad \min_{(\theta^{[0]},\theta^{[1]},\theta^{[2]})\in\Theta^{[2]}}\quad \mathcal{L}_{\mathcal{D}}[\mathcal{NN}^{[2]}(\cdot;\theta^{[0]},\theta^{[1]},\theta^{[2]})],$$

over an extended global budget of hyper-parameters $\Theta^{[2]}$. Moreover, **OP-2** itself is solved by sweeping over sub-problems as follows. Fixing the values of the trained parameters $\theta^{[1]}=\theta_*^{[1]}$ and $\theta^{[0]}=\theta_*^{[0]}$ we apply gradient descent first only over the newly added trainable parameters $\theta^{[2]}$. That is, we solve,

$$\textbf{OP-2}^*\textbf{:}\qquad \min_{\theta^{[2]}}\quad \mathcal{L}_{\mathcal{D}}[\mathcal{NN}^{[2]}(\cdot;\theta_*^{[0]},\theta_*^{[1]},\theta^{[2]})].$$

Due to the ResNet structure, upon defining the data

$$\mathcal{D}_1 := \{ (\mathcal{NN}^{[1]}(\mathbf{w};\theta_*^{[0]},\theta_*^{[1]}), \mathbf{c} - \mathcal{NN}^{[1]}(\mathbf{w};\theta_*^{[0]},\theta_*^{[1]})) \mid (\mathbf{w},\mathbf{c}) \in \mathcal{D}\},$$

this is equivalent to optimizing

$$\min_{\theta^{[2]}} \quad \mathcal{L}_{\mathcal{D}_1}[\mathcal{NN}^{[2]}(\cdot;\theta^{[2]})]$$

with a $\mathcal{NN}^{[1]}$-projected input and a residual from **OP-1** as output.

We can thus expect $\mathcal{NN}^{[2]}$ to perform better than $\mathcal{NN}^{[1]}$ in terms of learning the map between the observation $\mathbf{w}$ and state-labels $\mathbf{c}$ because adding an approximate residual presumably increases the overall accuracy of the estimator.

Once a new block has been added, the parameters in preceding blocks presumably could be further adjusted. This can be done by repeating a sweep over all blocks, i.e., successively optimizing each block while freezing the hyper-parameters in all remaining blocks.

The retained ghost-samples in $\mathcal{U}_{ghost}$ can now be used to assess the generalization error. If this is found unsatisfactory, the current ResNet can be expanded by a further block, leading to analogous optimization tasks **OP-k**. Moreover, for $k > 2$, the advantages of training shallow networks can be exploited by performing analogous block-optimization steps freezing the hyper-parameters in all but one block. In addition, we are free to design various more elaborate sweeping strategies to cover all trainable parameters need to be updated. For instance, one could add a round of updates toward all trainable parameters at the end of the training process. We refer to Section 5.4 for related experiments.

Another way of viewing this process is to consider the infinite optimization problem

$$\textbf{OP:} \qquad \min_{\theta \in \mathbb{R}^\infty} \mathcal{L}_D[\mathcal{NN}(\cdot;\theta)],$$

where $\mathcal{NN}(\cdot;\theta) := \lim_i \mathcal{NN}^{[i]}(\cdot;\theta^{[0]},\theta^{[1]},\cdots,\theta^{[i]})$ is an idealized ResNet with an infinite number of blocks. Then the above block expanding scheme can be interpreted as a greedy algorithm for approximately solving the infinite problem **OP**

$$\textbf{OP-i} = \qquad \min_{\theta^{[0]},\theta^{[1]},\cdots,\theta^{[i]}} \mathcal{L}_D[\mathcal{NN}(\cdot;\theta^{[0]},\theta^{[1]},\cdots,\theta^{[i]},\mathbf{0})],$$

successively increasing i until the test on ghost-samples falls below a set tolerance.

In Section 6, we will first present the accuracy of the recovered solutions with the **nonlinear** map learned with a ResNet compared to solutions obtained with the Reduced Basis Method (RBM). We show that in the piece-wise constant case, ResNet can reach similar accuracy as RBM. In log-normal case, while the RBM can not be applied to obtain reasonable solutions, we show that the ResNet can still be used to obtain a nonlinear solution with L_2 lifting (see Section 4.3).

4 Numerical results

The above description of a state estimation algorithm is so far merely a skeleton. Concrete realizations require fixing concrete algorithmic ingredients such as batch sizes, learning rates, and the total number of training steps (whose meaning will be precisely explained later in this section). The numerical experiments reported in this section have two major purposes: (I) shed some light on the dependence of optimization success on specific algorithmic settings, in particular, regarding two different principal training strategies. The first one represents standard procedures and applies Stochastic Gradient Descent (SGD) variants *to all trainable parameters*, defining a given DNN with ResNet architecture. Schemes of this type differ only by various algorithmic specifications listed below and will be referred to as Gl-ResNN and "training" then refers to the corresponding *global* optimization. The second strategy does not aim at optimizing a fixed network but intertwines optimization with a *blockwise network expansion*. This means that initially only a shallow network is trained which is subsequently expanded in a stepwise manner by additional blocks in a ResNet architecture. Such schemes are referred to by Exp-ResNN. The corresponding training, referred to as *blockwise training*, applies SGD only to the currently newly added block, freezing all parameters in preceding blocks. Comparisons between Gl-ResNN and Exp-ResNN concern achievable generalization error accuracies, stability, robustness with regard to algorithmic settings, and efficiency. (II) compare the performance of neural networks with estimators that are based on Reduced Basis concepts for Scenario (S1) of piecewise constant affine parameter representations of the diffusion coefficients, introduced in Section 2.1. (S1) is known to be a very favorable scenario for Reduced Basis methods that have been well studied for this kind of problems exhibiting excellent performance, [4,6].

Furthermore, we explore the performance of the ResNet-based estimator for scenario (S2), involving log-normal random diffusion parameters. In this case the performance of Reduced Basis is much less understood and hard to certify. Corresponding experiments are discussed in Section 4.3. To ease the description of the experiment configurations, we list the notations in Table 1.

4.1 Numerical set up

ResNets defined as in (3.17) with activation functions tanh are used for all experiments. These networks will be optimized with the aid of the Proximal Adagrad algorithm. This is essentially a variant of SGD which is capable of adapting learning rates per parameter. We further provide in Section 5.6 a comparison between Adagrad with the Adam algorithm to justify this choice.

To ensure all experiment results are fair comparisons, for each set of experiments, we will fix the number of total training steps (T). "Training step" means one iteration in Adagrad algorithm, which we sometimes also refer to as *one update step* of the trainable parameters. In particular, if the ResNet is trained in a block-wise sense (Exp-ResNN),

Table 1: Abbreviations.

	Abbreviation	**Explanation**	**Comments**
	ResNet	Residual neural network	General architecture is defined as in (3.17). Activation function $\sigma=\tanh$.
Training Scheme	Gl-ResNN	A deep network with ResNet architecture is trained by standard SGD methods applied in a global fashion to the full collection of trainable parameters.	
	Exp-ResNN	Deep networks with ResNet architecture are generated and trained by successively expanding the current network by a new block, confining SGD updates to a single block at a time	
ResNet Configurations	B	Number of blocks	
	W	Width of hidden layers	Random parameter initialization drawn from a Gaussian distribution
	O	Output dimension of $\mathbf{c}$	Number of orthogonal basis taken in $\widetilde{\mathbb{W}}_h^{\perp}$
Training Hyper-parameters	l	l^1 regularization weight of trainable learning parameters	0.00001 if not specified
	b	Batch size	100 if not specified
	lr	Learning rate	
	T	Number of total training steps	
Data Type	`Train`	Training data type	
	pwc	Piecewise constant case training set (S1)	10,000 samples subject to 16 uniformly distributed sensors
	log-normal	Log-normal case training data (S2)	1,000 samples when 16 sensors; 5,000 samples when 49 sensors
	`sen`	Number of uniformly distributed sensors	
	POD-H^1	Dimension reduction of $\widetilde{\mathbb{W}}_h^{\perp}$ with POD in H^1 sense	
	POD-L_2	Dimension reduction of $\widetilde{\mathbb{W}}_h^{\perp}$ with POD in L_2 sense	

then each block will be attributed an equal number of T/B steps to update the trainable parameters in this block. The choice of T will be specified in the loss history figures as well as in the error tables for each set of experiments.

With Exp-ResNN it is, in principle, possible to apply in the course of the training process SGD repeatedly to blocks that had been added at an earlier stage of the expansion process. In most numerical examples, only one round of parameter updating will be applied to the newly added last block. These blocks will not be revisited at a later time. The only exception takes place in Section 5.4, where global updates are carried out in addition to the block-wise training for the ResNet. Here we wish to see whether Exp-ResNN provides favorable initial guesses for a subsequent Gl-ResNN. We often refer to any arrangement of the order of blockwise updates and revisiting blocks as *training schedule*.

For measuring errors, we wish to estimate the relative analogue to the ideal regression risk (2.19):

$$\mathcal{E}:=\left(\frac{\mathbb{E}_{\mathbf{y}\in\mathcal{Y}_{test}}||u(\mathbf{y})-u_{pred}(\mathbf{y})||_{\mathbb{U}}^2}{\mathbb{E}_{\mathbf{y}\in\mathcal{Y}_{test}}||u(\mathbf{y})||_{\mathbb{U}}^2}\right)^{\frac{1}{2}},$$

where $u_{pred}:=\mathbf{\Phi}_h\mathbf{w}+\mathbf{\Psi}_h\mathbf{c}_{pred}$. In particular, if the norm is taken to be the problem compliant norm $||\cdot||_{\mathbb{U}}$, due to Remark 3.1, evaluating its empirical counter part:

$$\hat{\mathcal{E}}=\left(\frac{\sum_s||\mathbf{c}^s-\mathbf{c}^s_{pred}||_2^2}{\sum_s||\mathbf{w}||_2^2+||\mathbf{c}^s||_2^2}\right)^{\frac{1}{2}},$$

just requires computing Euclidean norms for the predicted coefficients $\{\mathbf{c}^s_{pred}\}$. Of course, we expect that sufficiently large sample sizes provide accurate estimates

$$\hat{\mathcal{E}}\approx\mathcal{E}.$$

If one would like to evaluate the accuracy of the recovered solution $u_{pred}(\mathbf{y})$ in a norm that is different from the natural norm ($||\cdot||_{\mathbb{U}}$), quadrature in the truth space $\mathbb{U}_h$ is required to approximately evaluate the respective norm of the functions.

4.2 The piecewise constant case (S1)

We first consider the aforementioned diffusion problem (2.5) with $f=1$ and a piecewise constant diffusive parameter within $\Omega=[0,1]^2$. More specifically, we consider a non-overlapping, 4×4 uniform decomposition $\{\Omega_j\}_{j=1}^{16}$ of Ω. The diffusion coefficient is a constant on each Ω_j as defined in (2.7) .

4.2.1 Gl-ResNN vs. Exp-ResNN

The first group of experiments concerns a general performance comparison between the ResNet expansion strategy – in short Exp-ResNN, outlined above and a global update strategy which updates the whole network with the same architecture, termed Gl-ResNN in what follows. By "performance" we mean training efficiency as well as corresponding achieved training and generalization losses. Specifically, we consider first the case where $m=16$ sensors are placed uniformly in Ω (see Fig. 4) and the corresponding measurements are evaluated by averaging the solution at the four vertices of a square of side 0.001 centered at the sensor location, see (3.7). Thus, the number of sensors equals in this case the parametric dimension so that there is a chance that the measurements determine the state uniquely.

We use in total 10,000 snapshots represented in the truth-space that serve as synthetic data, 500 of which are used for testing purposes. To draw theoretical conclusions a larger amount of test data would be necessary. However, intense testing has revealed that

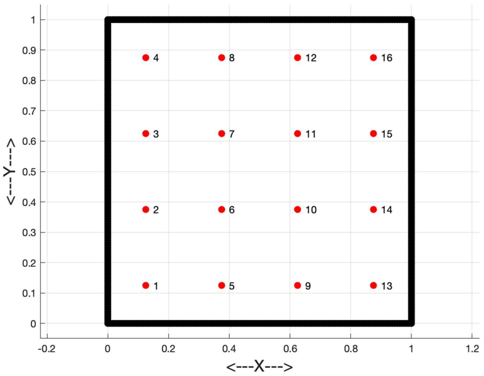


Figure 4: 16 uniformly distributed sensors.

larger test sizes have no significant effect on the results in the scenarios under consideration. Based on computations, using these data we find that $k=28$ basis functions suffice to reserve 99.5% of the H^1-energy in $\widetilde{\mathbb{W}}_h^{\perp}$, represented by the Hilbert-Schmidt norm of a full orthonormal basis in $\widetilde{\mathbb{W}}_h^{\perp}$. The case studies documented by subsequent figures are referenced as follows: "*pwc*" refers to "piecewise constant diffusion coefficients" in scenario (S1); "POD-H^1" indicates that the sensors have been Riesz-lifted to $H^1(\Omega)$ which accommodates the measurement space $\mathbb{W}$. The SVD truncation threshold is chosen to ensure accuracy in $H^1(\Omega)$; recall also that "sen16" means that the recovery is based on data from 16 sensors.

The numerical results, shown in Fig. 5, indicate that the training loss resulting from Exp-ResNN decays faster than the standard Gl-ResNN training in this case. The relative generalization errors for both approaches after 12×10^5 steps of training are displayed in Table 2, where the generalization errors are evaluated on the test set of 500 samples. The result shows that Exp-ResNN outperforms Gl-ResNN in a sense detailed later below. In particular, the training loss for Exp-ResNN drops faster to a saturation level which can be achieved by Gl-ResNN only at the expense of a significantly larger training effort. We also compare the estimated generalization error in H^1 with the achieved (expectedly smaller) error in the weaker L_2-norm.

One observes though that increasing network depth, i.e., employing a higher number of ResNet blocks does not increase accuracy significantly in either model. This indicates that a moderate level of nonlinearity suffices in this scenario. This is not surprising considering the moderate number of POD basis functions needed to accurately capture complement information.

To conclude, our findings can be summarized as follows: When using larger widths, e.g. 200, Exp-ResNN leads to a faster convergence and a somewhat smaller generalization error in comparison with plain Gl-ResNN training. In fact, while in Exp-ResNN the generalization error at least does not increase when increasing network complexity, plain Gl-ResNN shows a degrading performance reflecting increasing difficulties in realizing

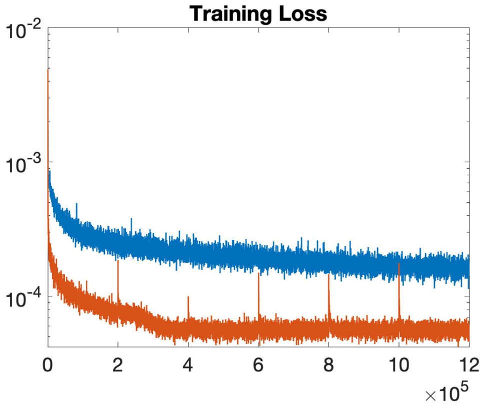
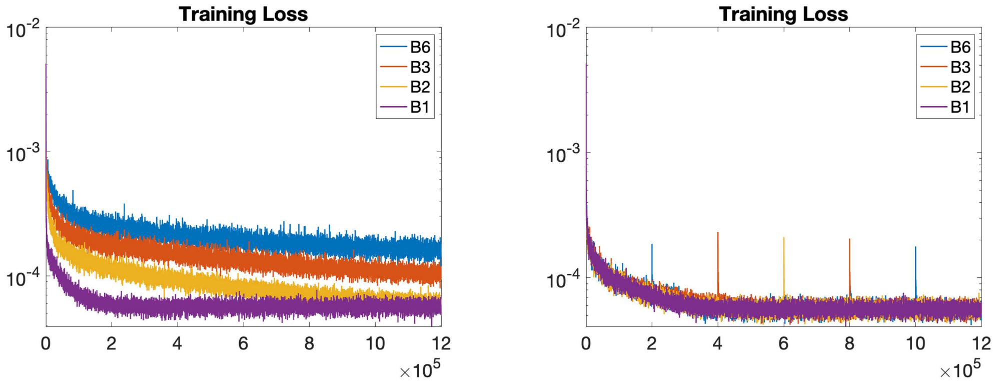


Figure 5: Exp-ResNN(red)/Gl-ResNN(blue) training loss comparison. Train-*pwc*-sen16-POD-H^1, B6-W200-O28-lr0.03.



(a) Gl-ResNN training loss

(b) Exp-ResNN training loss

Figure 6: Training loss of Gl-ResNN/Exp-ResNN v.s. different number of blocks. Train-*pwc*-sen16-POD-H^1, W200-O28-lr0.03.

expressive potential. On the other hand, for smaller widths, such as 20, the performance of both variants is comparable. It should be noted that already a single block and width $=20$ achieves an "empirical accuracy level" that is improved only slightly by more complex networks (compare Table 2 with Table 3). A significant increase in the number of trainable parameters has not resulted in a significant decrease of training and generalization losses, indicating that the global update strategy does not exhaust the expressive power of the underlying networks. This may rather indicate that larger neural network complexity widens a plateau of local minima of about the same magnitude in the loss landscape while parameter choices realizing higher accuracy remain isolated and very hard to find. Of course, this could be affected by different (more expensive) modelities in running parameter updates which incidentally would change the implicit regularization mechanism.

Table 2: Generalization error v.s. different number of ResNet blocks (fixed total training steps 12×10^5). `Train`-*pwc*-sen16-POD-H^1, Exp-ResNN/Gl-ResNN-W200-O28-lr0.03.

		$\hat{\mathcal{E}}$		Relative L_2 Error of u_{pred}	
# of blocks	#of trainables	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
1	49,648	9.76%	9.76%	3.01%	3.01%
2	101,248	9.77%	10.13%	3.01%	3.13%
3	152,848	9.76%	13.53%	3.01%	4.09%
6	307,648	9.78%	16.85%	3.01%	5.27%

Table 3: Generalization error v.s. different number of ResNet blocks (fixed total training steps 12×10^5). `Train`-*pwc*-sen16-POD-H^1, Exp-ResNN/Gl-ResNN-W20-O28-lr0.03.

		$\hat{\mathcal{E}}$		Relative L_2 Error of u_{pred}	
# of blocks	#of trainables	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
1	1,768	9.76%	9.76%	3.01%	3.01%
2	3,328	9.76%	9.76%	3.01%	3.01%
3	4,888	9.76%	9.77%	3.01%	3.01%
6	9,568	9.76%	9.76%	3.01%	3.01%

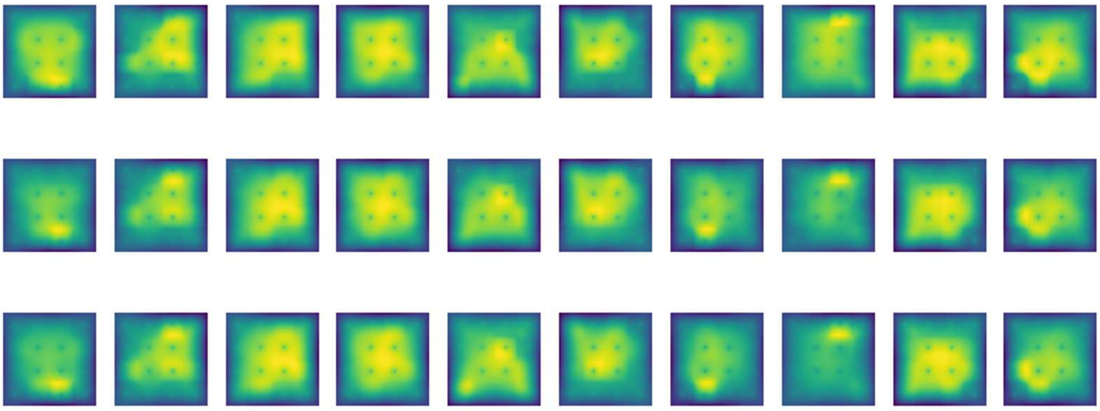


Figure 7: `Train`-*pwc*-sen16-POD-H^1, B6-W200-O28-lr0.03. Upper row: reference projected solution $z_h^s = P_{\widetilde{\mathbb{W}}_h^\perp} u_h^s$; Middle row: Prediction of z_h^s with Gl-ResNN; Lower row: Prediction of z_h^s with Exp-ResNN.

4.3 Log-normal case

As indicated earlier, uniform ellipticity in conjunction with affine parameter dependence of the diffusion coefficients offers very favorable conditions for the type of affine space recovery schemes described in Section 2.4. In particular, affine parameter dependence as well as rapidly decaying Kolmogorov n-widths (2.17) are quite important for methods, resorting to Reduced Bases, to work well, while neural networks are far less dependent on these preconditions. Therefore, we turn to scenario (S2) which is more challenging in both regards.

Specifically, we consider the case with $a_0=0$ and $a_1=1$ in (2.8). Recall that we aim to learn the map form $\mathbf{w}\to\mathbf{c}$ where $\mathbf{c}$ is the POD coefficient vector of the solution in $\mathbb{W}_h^{\perp}$.

First, since the diffusion coefficients no longer depend affinely on the parameters $\mathbf{y}$, rigorously founded error surrogates are no longer computable in an efficient way. This impedes the theoretical foundation as well as the efficiency of methods using Reduced Bases and therefore provides a strong motivation exploring alternate methods. Second, it is not clear whether the solution manifold $\mathcal{M}$ still have rapidly decaying n-widths, so that affine spaces of moderate dimension will not give rise to accurate estimators. In fact, computing the SVD of the snapshot projections $\{z_h^1,\cdots,z_h^{\widehat{N}}\}$, based on the Riesz representations of the measurement functionals in $\mathbb{U}=H^1$, shows only very slowly decaying singular values. This indicates that $P_{\mathbb{W}_h^{\perp}}(\mathcal{M})$ cannot be well approximated by a linear space of moderate dimension. Thus, when following the above lines, we would have to seek coefficient vectors $\mathbf{c}$ in a space of dimension comparable to $\widehat{N}$, which renders training Gl-ResNN prohibitive. Finally, the diffusion coefficients may near-degenerate degrading uniform ellipticity. This raises the question whether H^1 is still an appropriate space to accommodate a reasonable measurement space $\mathbb{W}$ which is at the heart of the choice of sensor coordinates (2.12). These adverse effects are reflected by the numerical experiments discussed below.

Therefore, we choose in scenario (S2) $\mathbb{U}=L_2(\Omega)$ which means we are content with a weaker metric for measuring accuracy. As a consequence, the representation of the functionals $\boldsymbol{\ell}(u)$ in $L_2(\Omega)$ is the L_2-orthogonal projection of these functionals to the truth-space $\mathbb{U}_h$ which then, as before, span the measurement space $\mathbb{W}_h\subset\mathbb{U}_h$. As indicated earlier, we give up on quantifiable gradient information but facilitate a more effective approximation of $P_{\mathbb{W}_h^{\perp}}(\mathcal{M})$ where the projection is now understood in the L_2-sense. In fact, for $\widehat{N}=1000$ snapshot samples, with the original H^1-Riesz-lifting, one needs the 1000 dominant POD modes to sustain 99.5% of the H^1 energy. Instead, only 21 dominating modes are required to realize the same accuracy in $L_2(\Omega)$. All results presented in this section will be subject to this change. Nevertheless, for comparison we record below in each experiment also the relative H^1-error which, as expected, is larger by an order of magnitude.

4.3.1 Exp-ResNN vs. Gl-ResNN (16 sensors)

We are interested to see whether, or under which circumstances, the advantages of Exp-ResNN over plain Gl-ResNN training persists also in scenario (S2) where several problem characteristics are quite different. We consider similar test conditions as before, namely 16 uniformly distributed sensors. In total, 1000 snapshots are collected providing 1000 synthetic data points, of which 950 are used for training while 50 are reserved for evalua-

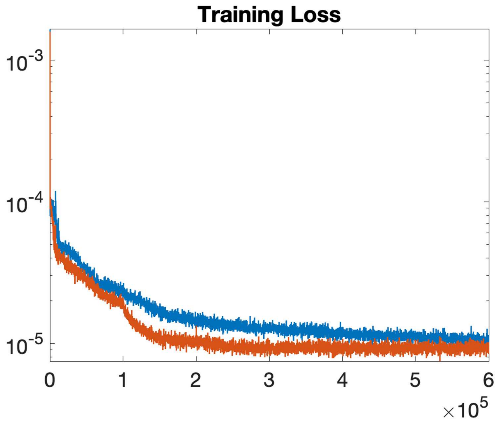
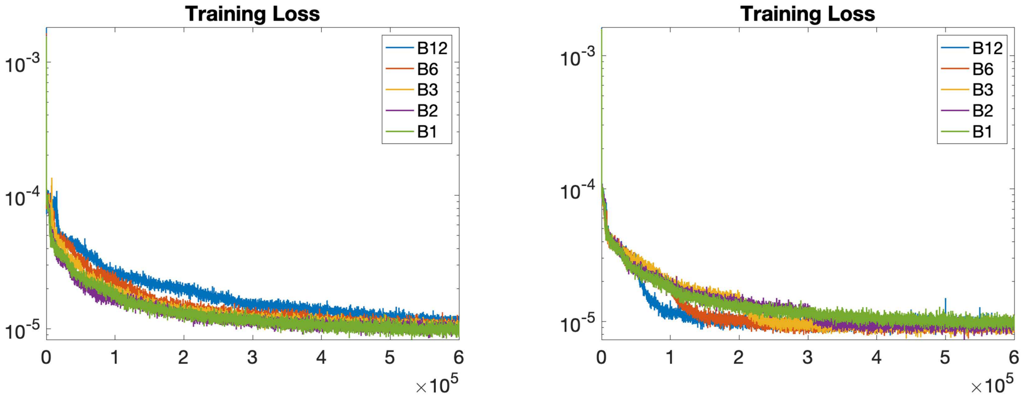


Figure 8: Exp-ResNN(red)/Gl-ResNN(blue) training loss comparison. Train-*log-normal*-sen16-POD-L_2, B6-W20-O21-lr0.02.



(a) Gl-ResNNtraining loss

(b) Exp-ResNNtraining loss

Figure 9: Training loss of Gl-ResNN/Exp-ResNN v.s. different number of blocks. Train-*log-normal*-sen16-POD-L_2, W20-O21-lr0.02.

tion in this case. The dimension of observational data $\mathbf{w}$ is then $m=16$, while according to the preceding remarks, the effective complement space dimension, accommodating the coefficients $\mathbf{c}$, is $k=21$.

The history of training losses is shown in Fig. 8, where we observe again that the training of Exp-ResNN is more efficient than that of Gl-ResNN. Specifically, when a new block is introduced in Exp-ResNN, the training loss decays more rapidly (see the corner of the loss curve in Fig. 8 at step 1×10^5). Thus, the expansion strategy is clearly beneficial in this case. Moreover, perhaps not surprisingly, we observe that Gl-ResNN suffers from a slow down in loss decay when training a larger number of parameters simultaneously (Fig. 6(a)). By contrast, for the block by block optimization in Exp-ResNN and fixed width, the number of simultaneously updated parameters stays constant so that even for

Table 4: Generalization error v.s. different number of ResNet blocks (fixed total training steps 6×10^5). `Train`-*log-normal*-sen16-POD-L_2, Exp-ResNN/Gl-ResNN-W20-O21-lr0.02.

		$\hat{\mathcal{E}}$		relative H^1 Error of u_{pred}	
# of blocks	#of trainables	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
1	1,516	7.89%	7.89%	47.61%	47.61%
2	2,796	7.65%	7.94%	47.33%	48.26%
3	4,076	7.60%	8.00%	47.08%	48.31%
6	7,916	7.60%	8.12%	47.08%	48.65%
12	15,596	7.76%	8.42%	47.42%	49.28%

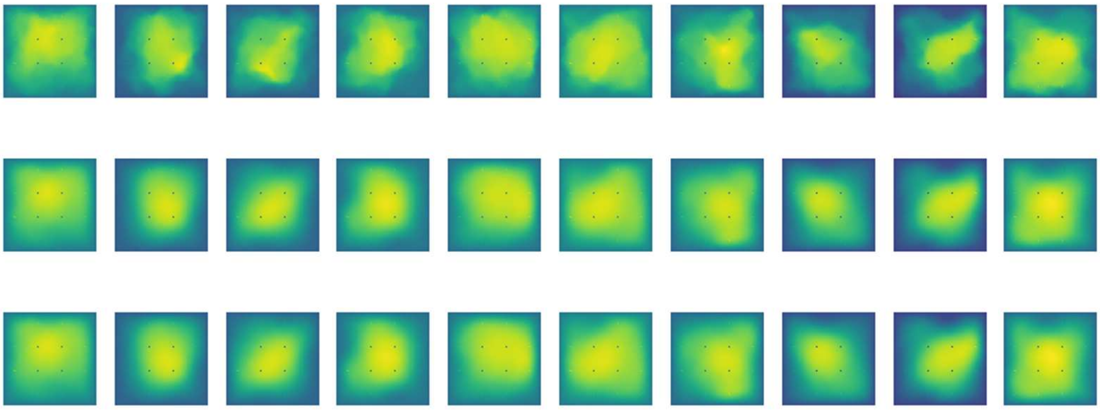


Figure 10: `Train`-*log-normal*-sen16-POD-L_2, B6-W20-O21-lr0.02. Upper row: reference projected solution $z_h^s = P_{\mathbb{W}_h^\perp} u_h^s$; Middle row: Prediction of z_h^s with Gl-ResNN; Lower row: Prediction of z_h^s with Exp-ResNN.

larger (deeper) networks, one reaches a similar level of training loss at a smaller number of updates (see Fig. 6(b)). Correspondingly, a slightly better overall accuracy of Exp-ResNN can be observed in terms of the generalization error (Table 4). One should keep in mind though that we have allotted a fixed budget of training steps to all variants in this experiment. Thus, increasing depth, reduces the training effort spent on each block which may explain the relatively large generalization error obtained for B12. Hence, when favoring accuracy improvements at the expense of more training steps, Exp-ResNN offers a clearly better potential while a global training seems to be rather limited. One the other hand, the results in Table 4 also indicate that not much gain in accuracy should be expected in this test case by using more than one or two blocks.

In summary, Exp-ResNN appears to offer advantages in training neural networks with larger depth compared with plain Gl-ResNN. The relatively coarse information provided by the 16 measurement data seems to leave more room for an enhanced nonlinearity of deeper networks to capture the manifold component in $\mathbb{W}^\perp$. This will be seen in

Section 4.3.2 to change somewhat when a larger number of sensors increases the accuracy of the "zero-order approximation" provided by the projection $P_{\mathbb{W}_h^\perp}$.

4.3.2 Exp-ResNN vs. Gl-ResNN (49 sensors)

In this subsection, we consider 49 uniformly distributed sensors for measurements. In total, 6000 snapshots are collected. Among them, 5950 samples are used for training while 50 are reserved for evaluation. The dimension of the latent space accommodating $\mathbf{c}^s$ is now $k=22$ after applying SVD and keeping 99.5% energy in the L_2 sense.

However, in this case, from Fig. 11, we can see that although the training loss of Exp-ResNN decays faster compared to Gl-ResNN at the beginning, both ended up at a

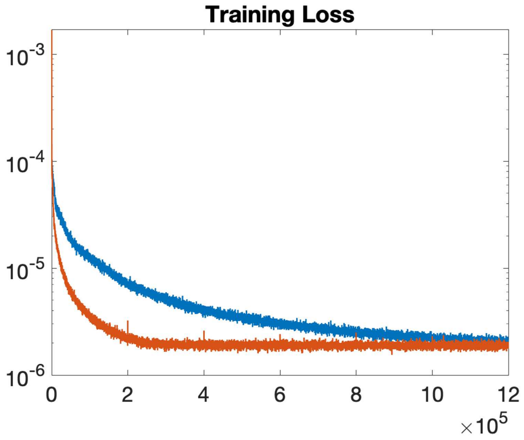
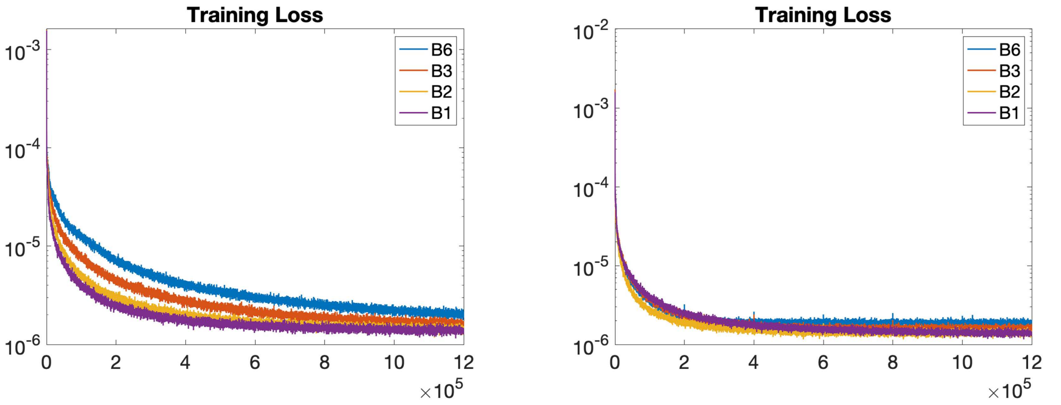


Figure 11: Exp-ResNN(red)/Gl-ResNN(blue) training loss comparison. Train-*log-normal*-sen49-POD-L_2, B6-W20-O22-lr0.03.



(a) Gl-ResNNtraining loss

(b) Exp-ResNNtraining loss

Figure 12: Training loss of Gl-ResNN/Exp-ResNN v.s. different number of blocks. Train-*log-normal*-sen49-POD-L_2, W20-O22-lr0.03.

Table 5: Generalization error v.s. different number of ResNet blocks (fixed total training steps 12×10^5). Train-*log-normal*-sen49-POD-L_2, Exp-ResNN/Gl-ResNN-W20-O22-lr0.03.

		$\hat{\mathcal{E}}$		H^1 Error of u_{pred}	
# of blocks	#of trainables	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
1	2,983	3.08%	3.08%	52.66%	52.66%
2	4,258	3.10%	3.13%	52.64%	52.71%
3	5,578	3.31%	3.34%	52.82%	53.06%
6	9,538	3.60%	3.69%	53.27%	53.90%

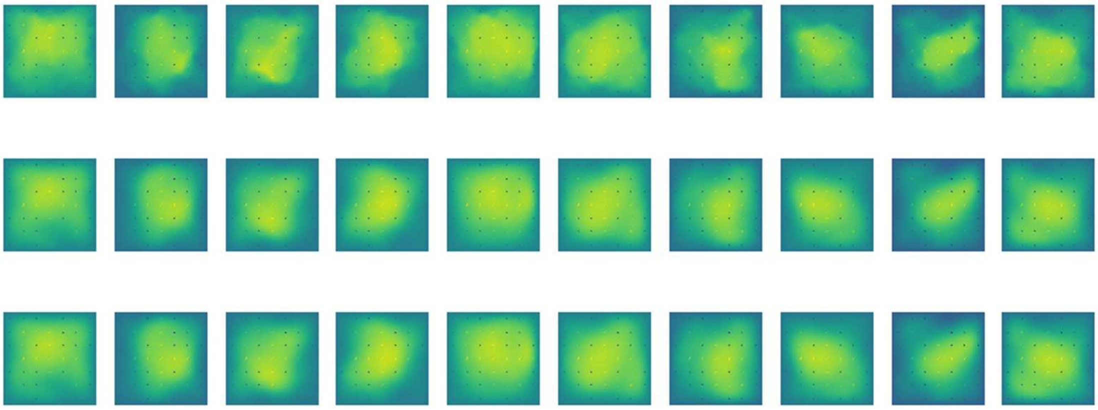


Figure 13: Upper row: reference projected solution $z_h^s = P_{\mathbb{W}_h^\perp} u_h^s$; Middle row: Prediction of the z_h^s with Gl-ResNN; Lower row: Prediction of z_h^s with Exp-ResNN.

similar level. We also do not observe significant changes in decay rates of the loss when employing the expansion strategy Exp-ResNN (see Fig. 11 at step 6×10^5). In fact, the generalization error of Exp-ResNN is only slightly smaller than that of Gl-ResNN (Table 5). In addition, it is seen that, within the fixed total number of training steps, the best performance is already achieved using a shallow Gl-ResNN. This indicates that, within the achievable accuracy range, the map of interest is close to a linear one, given that the "zero-order" approximation $P_{\mathbb{W}_h} u$ is now already rather accurate. However, we do notice that when applying a block by block training strategy in Exp-ResNN, while the difference in generalization error is small, the savings in training are huge because only one block is trained at a time, and thus the number of parameters under training is fixed. Thus, for deep neural networks, such a sequential training scheme is expected to be beneficial compared to updating all parameters simultaneously.

5 Robustness of Exp-ResNN with regard to algorithmic settings

In this section we further examine how sensitively Exp-ResNN depends on various algorithmic settings such as different learning rates or neural network architecture. We also wish to see whether extra steps of optimization over all trainable parameters can improve the overall performance, especially the generalization accuracy of Exp-ResNN. To be consistent, we stick to the example in Section 4.3.1, where a log-normal case is considered and 16 uniformly placed sensors give rise to the space $\mathbb{W}_h$ and the corresponding complement $\mathbb{W}_h^T$ in the truth space $\mathbb{U}_h$.

5.1 Sensitivity to learning rate

We first check how different learning rates affect the performance of Exp-ResNN in comparison with Gl-ResNN. In particular, the different learning rates considered are merely an initialization of the learning rates that are applied during the training process. The optimizer of our choice (AdaGrad, the adaptive gradient algorithm) is a modified stochastic gradient descent algorithm which will automatically adjust the learning rate per parameter as the training proceeds. Corresponding findings can be summarized as follows (see Fig. 14 and Table 6):

- The training loss of Exp-ResNN converges faster for a wide range of *constant initial* learning rates.
- The generalization errors for Exp-ResNN are smaller.

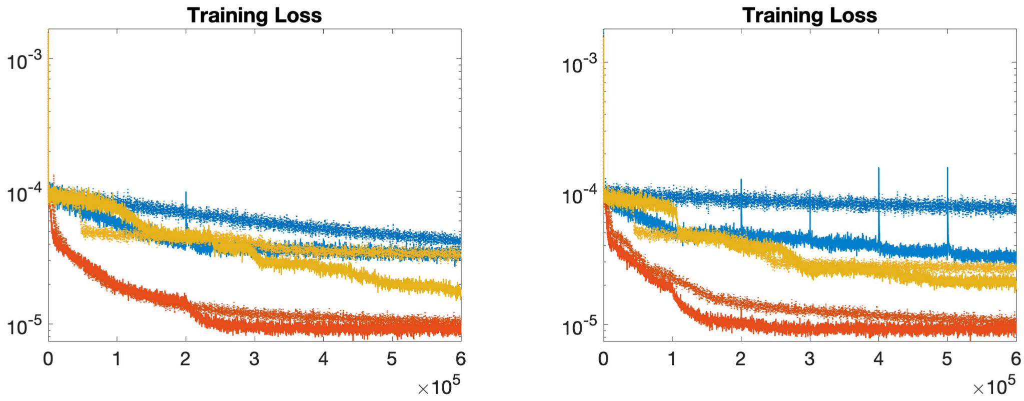
Table 6: Generalization error v.s. different learning rates (fixed total training steps 6×10^5). Train-*log-normal*-sen16-POD-L_2,W20-O21.

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
lr	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
0.002	10.75%	14.50%	54.25%	61.29%
0.02	7.60%	8.00%	47.08%	48.31%
0.2	14.78%	16.66%	60.62%	62.96%

(a) B3

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
lr	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
0.002	11.76%	13.38%	55.66%	60.13%
0.02	7.60%	8.12%	47.08%	48.65%
0.2	14.53%	23.44%	60.80%	76.68%

(b) B6



(a) B3

(b) B6

Figure 14: Training loss of Gl-ResNN(doted)/Exp-ResNN(solid) v.s. different learning rate (blue: lr = 0.2, red: lr = 0.02, yellow: lr = 0.2). `Train`-*log-normal*-sen16-POD-L_2, W20-O21.

Table 7: Generalization error v.s. different neural network width (fixed total training steps 6×10^5). `Train`-*log-normal*-sen16-POD-L_2, O21-lr0.02.

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
Width	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
W20	7.60%	8.00%	47.08%	48.31%
W200	11.41%	25.10%	55.17%	77.85%

(a) B3

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
Width	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
W20	7.60%	8.12%	47.08%	48.65%
W200	12.06%	24.94%	56.24%	75.99%

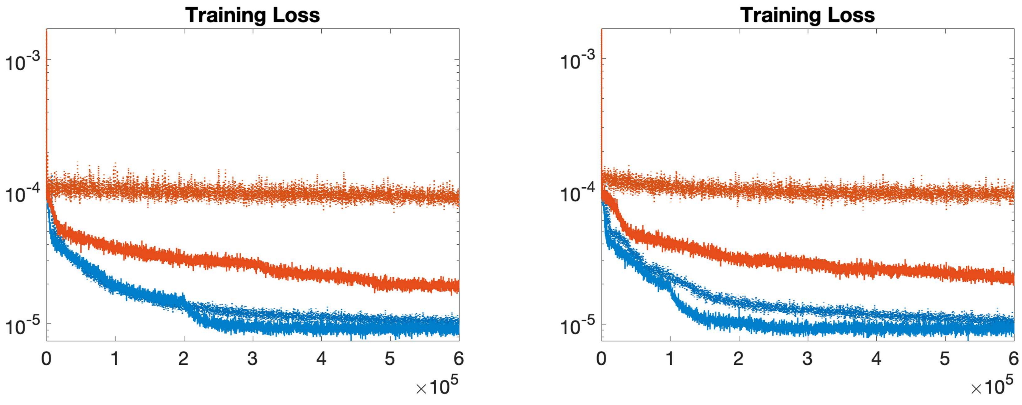
(b) B6

5.2 Dependence on neural network architecture

In this subsection we explore the effect of varying the architecture again for the application in scenario (S2).

5.2.1 Width

As indicated by Fig. 15 and as expected, neural networks of equal depth but larger widths require more training iterations to converge and each update is computationally more intense than for narrow ones. For both narrow and wide neural networks, Exp-ResNN appears to entail a faster decay of the loss during the training process as well as a more accurate prediction.



(a) B3 (b) B6

Figure 15: Training loss of Gl-ResNN(doted)/Exp-ResNN(solid) v.s. different width (blue: W20, red: W200). Train-*log-normal*-sen16-POD-L_2, O21-lr0.02

5.2.2 Dependence on depth

Similar to what can be observed from Fig. 9, we observe that, for the given test problems, deeper Exp-ResNNs can converge at a similar rate as shallower ones. By contrast, perhaps not surprisingly, the convergence of plain Gl-ResNNs will generally slow down as depth increases. In other words, the deeper, the slower is the convergence of Gl-ResNN. see Fig. 16.

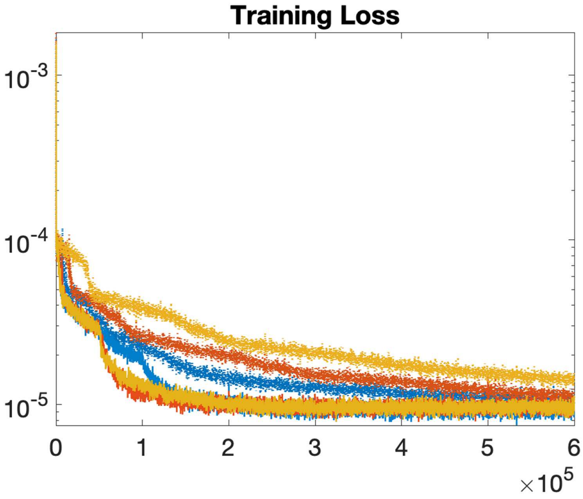


Figure 16: Training loss of Gl-ResNN(doted)/Exp-ResNN(solid) v.s. different depth (# of blocks, blue: B6, red: B12, yellow: B14). Train-*log-normal*-sen16-POD-L_2, W20-O21-lr0.02

Table 8: Generalization error v.s. different number of ResNet blocks (fixed total training steps 6×10^5). `Train`-*log-normal*-sen16-POD-L_2, W20-O21-lr0.02.

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
# of blocks	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
B6	7.60%	8.12%	47.08%	48.65%
B12	7.76%	8.42%	47.42%	49.28%
B14	8.17%	9.40%	47.78%	51.00%

5.3 Duration of training

The above discussion of the influence of width and depth on the learning outcome is based on an assumption that a fixed number of training steps is applied. It is, in general, completely unclear whether such a budget of training steps suffices to exploit the expressive power of the network. We are therefore interested to see how Exp-ResNN and Gl-ResNN compare when lifting the complexity constraints. This is the more interesting as an optimization step on a single block is not quite comparable with a descent step over the whole network. This discrepancy increases of course with increasing depth. As a first step in this direction, we inspect the effect of quadrupling the total number of training steps to 2.4×10^6.

It turns out that earlier findings are confirmed. Eventually, given enough training time and effort, Gl-ResNN can achieve about the same accuracy as Exp-ResNN which indicates that the "achievable" expressivity offered by the Gl-ResNN architecture has been

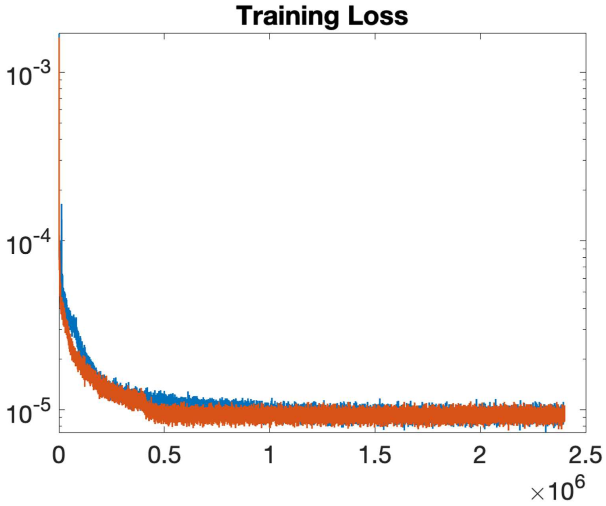


Figure 17: Training loss of Gl-ResNN(red)/Exp-ResNN(blue) for large number of training steps (2.4×10^6). `Train`-*log-normal*-sen16-POD-L_2, B6-W20-O21-lr0.02

Table 9: Generalization error v.s. different number of training steps. Train-*log-normal*-sen16-POD-L_2, B6-W20-O21-lr0.02.

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
Total training steps	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
6×10^5	7.60%	8.12%	47.08%	48.65%
24×10^5	7.75%	7.74%	47.10%	47.09%

exploited by both optimization strategies. There is a slight gain of accuracy in comparison with the previous cap of 6×10^5, namely $0.02\% \sim 0.03\%$.

5.4 Training schedules

In this example, we add an additional 2×10^5 training steps to update all trainable parameter simultaneously in addition to a block-wise training. Compared with a pure block by block training schedule (see Fig. 8), the additional global optimization effort does not seem to improve on the training success but rather worsens it (see Fig. 18).

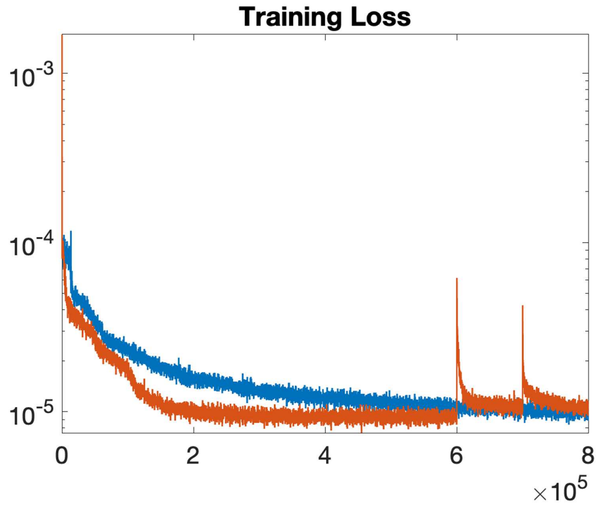


Figure 18: Training loss of Exp-ResNN(red)/Gl-ResNN(blue) with 2×10^5 steps of global updates. Train-*log-normal*-sen16-POD-L_2, B6-W20-O21-lr0.02

.

5.5 Wide neural network subject to long-term training

We check at last whether the neural networks can do better than in previous experiments in terms of approximating the map between $\mathbb{W}_h$ and $\mathbb{W}_h^\perp$ when significantly increasing training time. The results show that earlier findings persist. Table 11 shows that, while

Table 10: Generalization error with/without 2×10^5 global updates. `Train`-*log-normal*-sen16-POD-L_2, B6-W20-O21-lr0.02.

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
Total training steps	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
6×10^5 block-wise updates	7.60%	8.12%	47.08%	48.65%
6×10^5 block-wise updates + 2×10^5 global updates	7.75%	7.99%	47.10%	47.91%

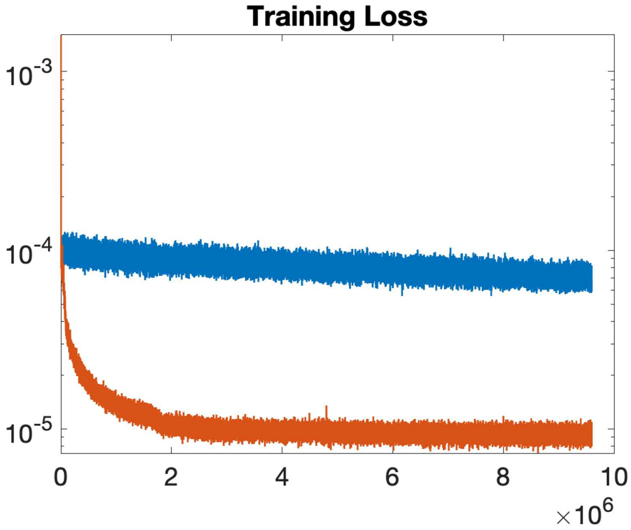


Figure 19: Training loss of Exp-ResNN(red)/Gl-ResNN(blue) for large number of training steps (9.6×10^6) of wide neural network (W200). `Train`-*log-normal*-sen16-POD-L_2, B6-W200-O21-lr0.02

with Exp-ResNN there is no real benefit of larger width, at least estimation quality does not appear to degrade over long training periods. Instead, Gl-ResNN seems to be adversely affected by larger network complexity.

More specifically, Table 11 shows that the smallest generalization error is achieved by the relatively low training effort for narrow networks. Much larger networks, instead, seem to achieve about that same accuracy level only at the expense of a significantly larger training effort, leaving little hope for substantial further accuracy improvements by continued training. In contrast, globally updating corresponding networks achieves the best result again for narrow networks but, in agreement with earlier tests, at the expense of four times as many training steps than for block-wise training. For wide networks even extensive training effort does not seem to reproduce accuracy levels, achieved earlier for smaller networks.

Table 11: Generalization error v.s. different number of training steps. Train-*log-normal*-sen16-POD-L_2, B6-O21-lr0.02.

	$\hat{\mathcal{E}}$		Relative H^1 Error of u_{pred}	
Total training steps	Exp-ResNN	Gl-ResNN	Exp-ResNN	Gl-ResNN
short (6×10^5) & narrow (W20)	7.60%	8.12%	47.08%	48.65%
long (24×10^5) & narrow (W20)	7.75%	7.74%	47.10%	47.09%
short (6×10^5) & wide (W200)	12.06%	24.94%	56.24%	75.99%
long (24×10^5) & wide (W200)	8.82%	23.09%	49.66%	78.19%
super long (96×10^5) & wide (W200)	7.77%	21.83%	47.17%	71.95%

5.6 Optimizer

Here we present a comparison of the Proximal Adagrad with Adam. From Fig. 20 we also observe that the training process of Exp-ResNN is more stable than Gl-ResNN. From Fig. 21 and Table 12, we can further see that both optimizers seem to provide similar results by the end of the training process. But Adam seems to be overall a little less stable. The reason for taking the learning rate for Adam to be 0.001 is because larger rates like 0.01 appear to produce meaningless results.

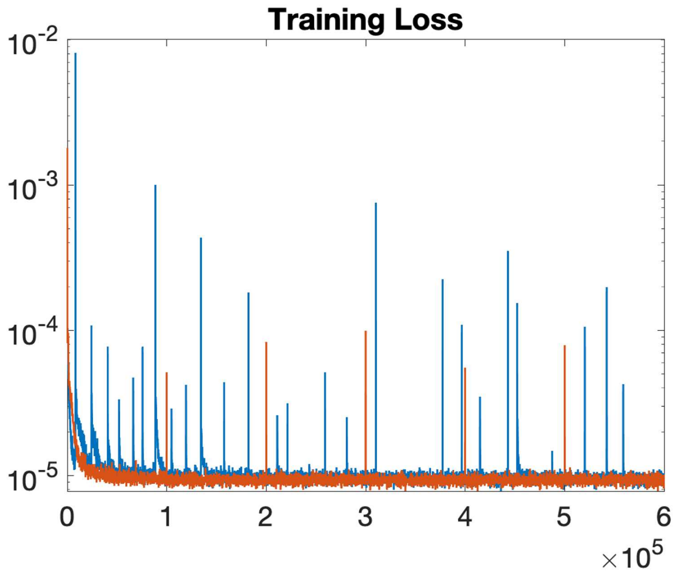
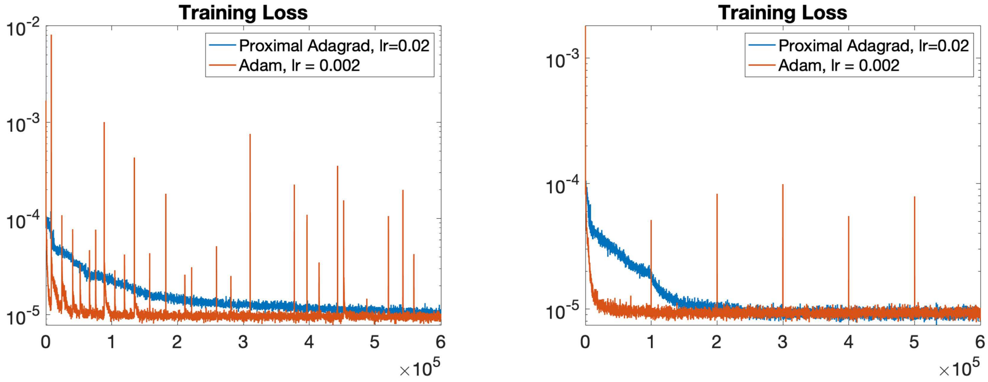


Figure 20: Training history using Adam. Train-*pwc*-sen16-POD-H^1, Exp-ResNN(red)/Gl-ResNN(blue)-W20-O21-lr0.02/0.002.



(a) Gl-ResNN (b) eGl-ResNN

Figure 21: Training history using Adam v.s. Proximal Adagrad. `Train`-*pwc*-sen16-POD-H^1, Gl-ResNN/Exp-ResNN-W20-O21-lr0.02/0.002.

Table 12: Generalization error v.s. different optimizer(fixed total training steps 6×10^5). `Train`-*pwc*-sen16-POD-H^1, eResNet/ResNet-W20-O21-lr0.02/0.002.

		$\hat{\mathcal{E}}$		Relative L_2 Error of u_{pred}	
# of blocks	learning rate	eResNet	ResNet	eResNet	ResNet
Proximal Adagrad	0.02	7.76%	8.42%	47.42%	49.28%
Adam	0.002	7.47%	7.82%	47.14%	47.23%

6 Comparison with affine reduced basis estimators

As discussed earlier in more detail, we have run these experiments with varying choices of learning rates and architecture modifications consistently obtaining essentially the same magnitude of training loss and generalization error. This indicates a certain saturation effect as well as reliability in generating consistent results. Nevertheless, one wonders to what extent the actual expressive power of the networks is at least nearly exhausted and how to gauge the results in comparison with alternate approaches. We have therefore compared the accuracy achieved by Exp-ResNN with results for Affine Space estimators from [5], mentioned earlier in Section 2.4. For the sake of such a comparison we consider the same type of up to 50 randomly distributed sensors depicted in Fig. 22). We generate again 10,000 data points 9000 of which are used for training while the rest is used for evaluation. Based on the experiences gained in previous experiments we have used a B2-W100 Exp-ResNN as a model for learning the observation-to-state mapping. That is, the Gl-ResNN of 2 blocks is trained in the expansion manner. We confine the training of the Exp-ResNN to a fixed number of 100,000 steps. The dependence of the tested generalization error on the number of sensors is shown in Fig. 23. As expected, an increasing number of sensors provides more detailed information on $P_{\widetilde{\mathbb{W}}_h^\perp}(\mathcal{M})$. The

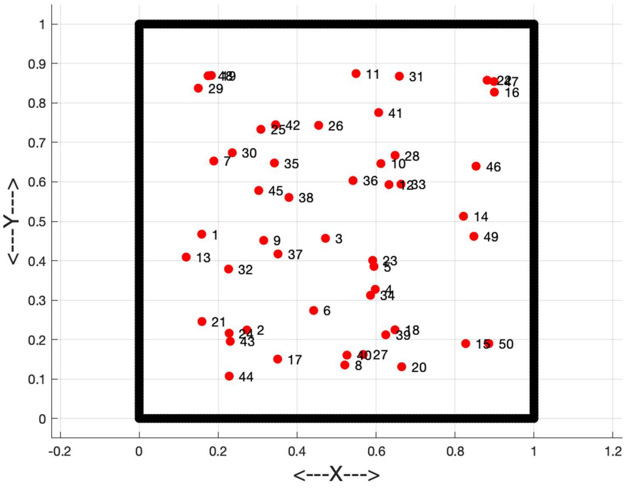


Figure 22: An illustration of 50 random sensors.

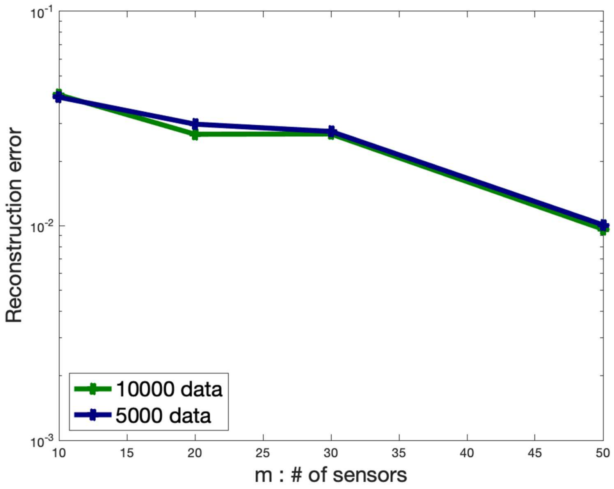
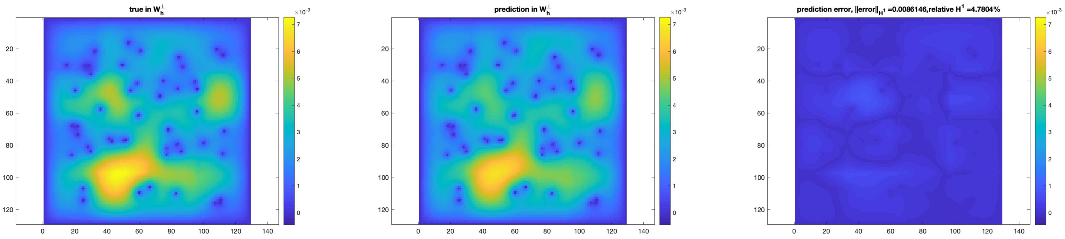


Figure 23: Max reconstruction H^1 errors among 1000 testing samples with Exp-ResNN.

green and blue curve show that further increasing the number of training data has little effect on the achieved accuracy. For the given fixed budget of 10,000 steps the evaluation shows an H^1-error of about 10^{-2}.

The performance of several versions of Affine Space estimators under the same test conditions has been reported in [5]. The computationally most expensive but also most accurate Optimal Affine Space estimator achieves in this experiment roughly an H^1-error of size 3×10^{-3} which is slightly better than the accuracy 8×10^{-3}, observed for Exp-



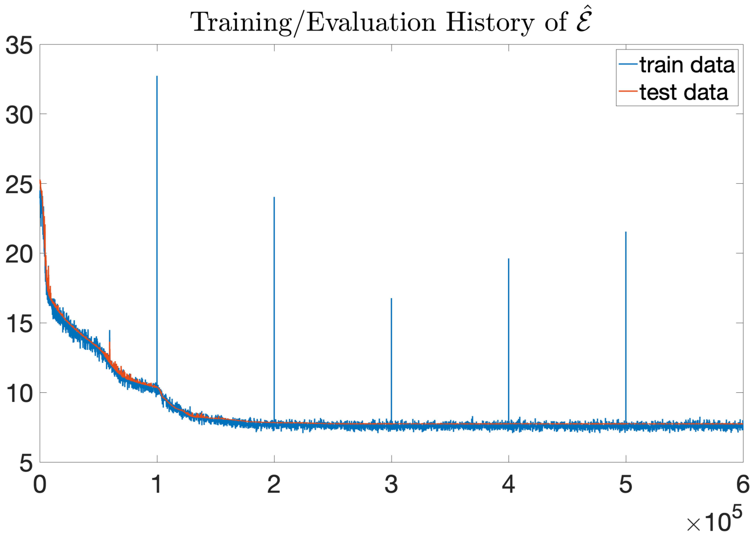
(a) Exact solution (b) Prediction (c) Error

Figure 24: An example of exact vs. Exp-ResNN predicted solution in $\mathbb{W}_h^T$ for 50 random sensors case.

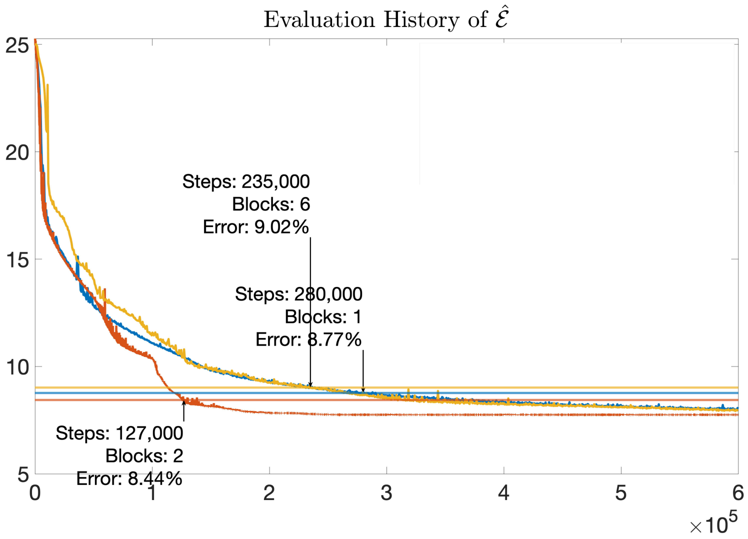
ResNN. However, when lifting the cap of at most 10,000 training steps, the observed maximal H^1-error for Exp-ResNN drops also to 5×10^{-3} in the 50-sensor case, which is at the same level as what the best affine estimator achieves. In summary, it seems that for this type of problems both types of estimators achieve about the same level of estimation accuracy and nonlinearity of the neural network lifting map does not seem to offer substantial advantages for scenario (S1). There is instead a noteworthy difference regarding computational cost in relation to predictable training success. The above example shows that neural networks may have significant disadvantages with regard to optimization success and incurred computational cost in comparison with affine-space recovery schemes where, however, Exp-ResNN shows a consistent level of reliability that avoids degrading accuracy under over-parametrization.

7 Summary

Overall, the experiments for the two application scenarios (S1), (S2) reflect the following general picture. Regardless of a specific training mode accuracy improves rapidly at the beginning and essentially saturates at a moderate network complexity. Beyond that point further improvements require a relatively substantial training effort which may not even be rewarded in the Gl-ResNN mode. Instead, Exp-ResNN usually responds with slight improvements and essentially never with an accuracy degradation. While in a number of cases Exp-ResNN achieves a smaller generalization error than plain Gl-ResNN, by and large, the differences in accuracy are not overly significant. Once the generalization error curve starts flattening, additional increases of network complexity seem to just increase over-parametrization and widen a flat plateau fluctuating around "achievable" local minima. Instead a realization of theoretically possible expressive power seems to remain highly improbable. Aside from an increased robustness with respect to algorithmic settings, the main advantage of Exp-ResNN over Gl-ResNN seems to lie in substantial savings of computational work needed to nearly realize an apparently achievable generalization accuracy. This is illustrated by Fig. 25(b), recording the work needed to achieve



(a) B6: Train/Evaluation Error history



(b) B1/B6: Evaluation Error History. Horizontal lines: 10/9 level of the terminal evaluation error (blue: Gl-ResNN-B1, red: Exp-ResNN-B6, yellow: Gl-ResNN-B6)

Figure 25: `Train`-*log-normal*-sen16-POD-L_2, W20-O21-lr0.02.

10/9 of the smallest generalization error achieved by the respective training modality. It is also interesting to note that the generalization errors at various optimization stages are not much larger than the corresponding relative loss-size, reflecting reliability of the schemes, see Fig. 25(a).

Acknowledgments

This work was supported by National Science Foundation under grant DMS-2012469. We thank the reviewers for their valuable comments regarding the presentation of the material.

References

[1] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk, Convergence rates for greedy algorithms in reduced basis methods, SIAM Journal of Mathematical Analysis 43 (2011): 1457-1472.

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

[3] A. Cohen and R. DeVore, Approximation of high dimensional parametric pdes, Acta Numerica, 2015.

[4] C. Albert, W. Dahmen, R. DeVore, J. Fadili, O. Mula, and J. Nichols, Optimal reduced model algorithms for data-based state estimation, SIAM Journal on Numerical Analysis 58, no. 6 (2020): 3355-3381.

[5] C. Albert, W. Dahmen, R. DeVore, and J. Nichols, Reduced basis greedy selection using random training sets, ESAIM: Mathematical Modelling and Numerical Analysis 54, no. 5 (2020): 1509-1524.

[6] C. Albert, W. Dahmen, O. Mula, and J. Nichols, Nonlinear reduced models for state and parameter estimation, SIAM/ASA Journal on Uncertainty Quantification 10, no. 1 (2022): 227-267.

[7] D. Wolfgang, C. Plesken, and G. Welper, Double greedy algorithms: Reduced basis methods for transport dominated problems, ESAIM: Mathematical Modelling and Numerical Analysis 48, no. 3 (2014): 623-663.

[8] D. Wolfgang, R. Stevenson, and J. Westerdiep, Accuracy controlled data assimilation for parabolic problems, Mathematics of Computation 91.334 (2022): 557-595.

[9] N. D. Santo, S. Deparis, and L. Pegolotti, Data driven approximation of parametrized PDEs by reduced basis and neural networks, Journal of Computational Physics 416 (2020): 109550.

[10] Y. Maday, A.T. Patera, J.D. Penn and M. Yano,A parametrized-background data-weak approach to variational data assimilation: Formulation, analysis, and application to acoustics, Int. J. Numer. Meth. Eng. 102 (2015): 933-965.

[11] G. Rozza, D.B.P. Huynh, and A.T. Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations – application to transport and continuum mechanics, Archives of Computational Methods in Engineering 15, no. 3 (2008): 229-275.

[12] D. Braess, Finite Elements, Theory, Fast Solvers, and Applications in Solid Mechanics, Cambridge University Press, 1997.

[13] K. Bhattacharya, B. Hosseini, N.B. Kovachki and A.M. Stuart, Model reduction and neural networks for parametric PDEs, Arxiv Preprint: https://arxiv.org/pdf/2005.03180.pdf

[14] D. P. Kroese and Z. I. Botev, Spatial process simulation, in: Stochastic Geometry, Spatial Statistics and Random Fields, pp. 369-404. Springer, Cham, 2015.

[15] E. T. Chung, Y. Efendiev, and Y. Li, Space-time GMsFEM for transport equations, GEM-International Journal on Geomathematics 9, no. 2 (2018): 265-292.

[16] E. Chung, Y. Efendiev, Y. Li, and Q. Li, Generalized multiscale finite element method for the steady state linear Boltzmann equation, Multiscale Modeling & Simulation 18, no. 1 (2020): 475-501.

[17] C.R. Dietrich and G.N. Newsam, Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix, SIAM Journal on Scientific Computing, 18(4), 1088-1107, 1997.

[18] D. John, E. Hazan, and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12.7 (2011).

[19] S. Yoram, and J. C. Duchi, Efficient learning using forward-backward splitting, Advances in Neural Information Processing Systems 22 (2009).