ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Higher-order Quasi-Monte Carlo Training of Deep Neural Networks

M. Longo and S. Mishra and T. K. Rusch and Ch. Schwab

Research Report No. 2020-57
September 2020

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Funding ERC: 770880 COMANFLO

Higher-order Quasi-Monte Carlo Training of Deep Neural Networks

M. Longo, S. Mishra, T. K. Rusch and Ch. Schwab

September 5, 2020

Abstract

We present a novel algorithmic approach and an error analysis leveraging Quasi-Monte Carlo points for training deep neural network (DNN) surrogates of Data-to-Observable (DtO) maps in engineering design.

Our analysis reveals higher-order consistent, *deterministic* choices of training points in the input data space for deep and shallow Neural Networks with holomorphic activation functions such as tanh.

These novel training points are proved to facilitate higher-order decay (in terms of the number of training samples) of the underlying generalization error, with consistency error bounds that are free from the curse of dimensionality in the input data space, provided that DNN weights in hidden layers satisfy certain summability conditions.

We present numerical experiments for DtO maps from elliptic and parabolic PDEs with uncertain inputs that confirm the theoretical analysis.

1 Introduction

Many computational problems with PDEs require the evaluation of *data-to-observables maps* (DtOs for short) (functionals, quantities of interest) of the generic form,

$$\text{Given Data } y \in Y \text{compute observable } g(y). \tag{1}$$

Here, the observable $g : Y \subset \mathbb{R}^d \to \mathbb{R}^{N_{\rm obs}}$, a is a function of some prescribed regularity, that depends on the solution of an underlying operator equation subject to input data $y \in Y$. Such observables arise for instance in uncertainty quantification (UQ) of PDEs, where Y denotes a data space. We focus on Y as a bounded subset of euclidean space parametrizing input data for the PDE model. Other examples include optimal control and design for PDEs, with Y being the control (or design) space. Note that this very generic definition of observables in (1) also includes the solution field by letting Y be the space (space-time) domain.

Computing *observables* of the form (1) requires one to numerically approximate PDEs and possibly, use quadratures to approximate integrals. Given that currently available PDE solvers such as finite element or finite volume methods can be computationally expensive, solving *many query* problems such as UQ, inverse problems and optimal control (design) for very-high dimensional parameter space Y in (1), that require a large number of calls to the underlying PDE solver, can be prohibitively expensive.

Surrogate models [15] provide a possible pathway for reducing the computational cost of such *many query* problems for PDEs. Surrogates such as reduced order models [36] and Gaussian process regression [38], build a *surrogate approximation* $\hat{g} : Y \mapsto \mathbb{R}^{N_{\rm obs}}$ such that $\hat{g} \approx g$ in a suitable sense. As long as the surrogate is sufficiently accurate and the cost of evaluating the

surrogate is significantly less than the cost of numerically evaluating the underlying DtO map g (1) with the same level of fidelity, one can expect the surrogate model to be more computationally efficient than so-called "high-fidelity" PDE solvers. These deliver, for example by discretization of the PDE with discretization parameter $\delta \in (0, 1]$, a one-parameter family of approximate forward maps $\{g_\delta\}_{0<\delta\leq 1}$, which is assumed to be consistent with g in the sense that

$$\lim_{\delta \downarrow 0} g_\delta = g \quad \text{uniformly with respect to } Y \,. \tag{2}$$

Here, the discretization parameter δ could, e.g., be a stepsize $\Delta t > 0$, a FE/FD meshwidth $h > 0$ or the reciprocal of a spectral order.

Building surrogates $\hat{g}$ to g with certified fidelity uniform with respect to the set of input data Y can be challenging. Accordingly, during the past decade computational science and engineering has witnessed the arrival of mathematical and computational frameworks aiming at generating such surrogates computationally, and to quantify the corresponding emulation error mathematically. These frameworks go by the name of "Reduced Basis (RB) methods" or "Model Order Reduction (MOR) techniques". They aim at *computational determination of low-dimensional subspaces* X_N of the vector space X containing the response of the PDE model of interest. We refer to the surveys [36, 23] and the references there for details and theory. In MOR and RB, the phase of building $\hat{g}$ is usually referred to as "offline phase" and is a) usually quite costly and b) is based on executing certain greedy searches on numerical approximations g_δ of the PDE of interest.

Deep neural networks (DNNs) (e.g. [18]; specifically, here the term "DNN" will denote a so-called feed-forward NN) are concatenated, repeated compositions of affine maps and scalar non-linear activation functions. In recent years, DNNs emerged as another powerful tool in computational science with well-documented success in a variety of tasks such as image classification, text and speech recognition, robotics and protein folding [25]. Their mathematical structure allows the interpretation of MOR and RB as particular instances (see, e.g., [39] for a development of this point of view in the context of parametric dynamical systems). Given their *universality*, i.e., the ability to approximate ("*express*" in the terminology of the deep learning community) large classes of functions (e.g. [35] and the references there), and their high approximation rates on regular maps (e.g. [4, 33, 32] and the references there), DNNs are increasingly being used in various contexts for the numerical approximation of DtO maps for PDEs [37, 19, 3].

In particular, recent papers such as [27, 26, 30, 28] have proposed using DNNs for building surrogates for *observables* of PDEs and applying these surrogates to accelerate UQ for PDEs [27, 26] and PDE constrained optimization [28]. These articles use DNNs within the paradigm of *supervised learning* i.e. select a *training set* $\mathcal{S} \subset Y$ and use a (stochastic) gradient descent algorithm to find tuning parameters (weights and biases) that provide the smallest mismatch between the underlying DtO map g and the resulting neural network on this training set.

It is standard in machine learning [29] to choose independent and identically distributed random points in Y to constitute the training set. However, as pointed out in [27, 30] and references therein, the so-called *generalization gap*, i.e. the difference between the *generalization error* or *population risk* (see (14) for a precise definition) and the computable *training error* or *empirical risk* for *trained* DNNs scales, at best, as $1/\sqrt{N}$ (in the root mean square sense), with $N = \#(\mathcal{S})$ being the number of training points (samples). We refer again to [29] and references therein for sharper estimates on the generalization gap.

Given this slow decay of generalization error in terms of the number N of i.i.d random training samples, one possibly needs a large number of samples to achieve a desired level of DNN fidelity. In emulation of DtOs from PDEs, training data is generated by N-fold calling the underlying PDE solver (with discretization error $|g - g_\delta|$ well below the DNN target fidelity).

Thus, a potentially prohibitively large number N of calls to the 'high-fidelity' forward PDE solver could preclude efficient training and surrogate modeling, see, e.g. [27] and references therein.

In this reasoning, we must distinguish between the **exact DtO map** g (which is, generally, not numerically accessible) and its **numerical approximations** $\{g_\delta : 0 < \delta \leq 1\}$. For our results to hold for the exact forward map g, we require the discretization error δ in the numerical approximation g_δ of the DtO map g to be uniformly smaller than the target DNN emulation fidelity $\hat{\varepsilon} > 0$ of the DNN $\hat{g}$ approximating g. I.e., we require

$$\sup_{y \in Y} |g(y) - g_\delta(y)| \leq \delta \overset{!}{\leq} \hat{\varepsilon} := \sup_{y \in Y} |g(y) - \hat{g}(y)| \,. \tag{3}$$

In order to alleviate prohibitive DNN training cost, one could consider more sophisticated training designs $\mathcal{S}$. The authors of [27, 30] propose using *low-discrepancy sequences*, such as Sobol' and Halton sequences used in *quasi-Monte Carlo* (QMC) quadrature algorithms [5], as training points. In [30], the authors prove that as long as the underlying map (1) is of bounded Hardy-Krause variation, one can prove that the generalization gap for supervised DNN with QMC training points, decays (upto a logarithmic correction) as N^{-1} i.e. linearly in the number of training samples. Furthermore, these training points lead to deterministic bounds on the generalization gap that are inherently more robust and easier to verify than probabilistic root mean square bounds with random training points. Numerical examples, presented in [27] and [30] demonstrate the increased efficiency of using QMC points for training, when compared to random points.

However, as is well known, the logarithmic correction stemming from the Koksma-Hlawka inequality for QMC, *depends exponentially* on the underlying dimension d of the parameter space Y. Consequently, the proposed deep learning algorithm with (for example) sets $\mathcal{S}$ chosen as Sobol' training points, suffers from the *curse of dimensionality*. This is also demonstrated in numerical experiments in [30] where the deep learning algorithm based on QMC training points, outperforms the one with random training points, only for problems in moderately high dimensions. It is natural to ask if one can find training sets $\mathcal{S}$ for DNNs which overcome this curse of dimensionality, while still possessing a faster rate of decay than the use of i.i.d random training points. In particular, if one can find training point designs $\mathcal{S}$ which ensure higher than linear rate of decay of the generalization gap, independent of high parameter space dimension.

It turns out that recent developments in *Quasi-Monte Carlo integration* algorithms, namely the design of *higher order QMC* (HoQMC) rules [14, 10, 16] can provide positive answers to the above questions. In particular, in the context of numerical integration, these rules lead to a dimension-independent, superlinear decay of the quadrature error as long as the integrand appearing in the loss function is *holomorphic* with holomorphy domains quantified in terms of the coordinate and the integrand dimension; see, e.g., [6, 14].

The loss function being (an integral of) a difference between the map $y \mapsto g(y)$ and a DNN surrogate $y \mapsto \hat{g}(y)$, this entails holomorphy requirements of both, the DtO map, as well as of the DNN surrogates. We point out that a large number of PDEs, particularly of the elliptic and parabolic type possess solutions (and observables) whose DtO maps are holomorphic in the parameter space. Moreover, in [6] the authors introduced two sufficient criteria that ensure holomorphy in a variety of cases, including UQ for non-linear PDEs and shape holomorphy (e.g. [6, 8, 21, 2] and the references here).

Motivated by the higher-order QMC rules in [14, 10, 16], in this article we make the following *contributions*:

- We develop several *novel DNN training strategies*, based on the use of *deterministic, higher order Quasi-Monte Carlo (QMC) point designs* as DNN training points, in order to emulate Data-to-Observables maps for systems governed by parametric PDEs.

- We prove, under *quantified holomorphy hypotheses on the DtO map to be emulated by the DNN and on the scalar activation function of the DNN*, that *for any input dimension* d,

the generalization gap of the resulting trained DNN decays superlinearly with respect to the number N of training points, and independent of the dimension of data space. I.e., we prove a bound $O(N^{-\alpha})$ with $\alpha \geq 2$ and N being the number of training points, with the constant implied in $O()$ being independent of the dimension d of the input parameter domain. Thus, the proposed deep learning algorithm can achieve significantly lower errors than the one based on random i.i.d training points, while still being free of the curse of dimensionality.

- We present a suite of numerical experiments for data-to-observables which arise from parametric PDEs with uncertain input data to illustrate the theory. We also show numerical experiments which strongly indicate that several hypotheses on holomorphy and sparsity in our results appear to be necessary, while others seem to be artifacts of our proofs based on complex-variable techniques.

The rest of the paper is organized as follows. The deep learning algorithm is presented in Section 2 and is analyzed in Section 3. Several illustrative numerical experiments are presented in Section 4 and the contributions of the current article and possible extensions are discussed in Section 5.

2 Deep Learning on higher-order Quasi-Monte Carlo training points

In the present section we briefly recapitulate elements from Quasi-Monte Carlo integration, as they pertain to the proposed higher-order lattice integration schemes which we subsequently use for defining the DNN loss function. In Section 2.2 we define the architectures of DNNs that we consider. Section 2.3 then introduces the computable loss functions which we use in DNN training. and outline our proposed "Deep learning with Higher-order Quasi-Monte Carlo points (DL-HoQMC)" algorithm.

2.1 Higher-order Quasi-Monte Carlo rules

We consider two classes of higher order QMC quadrature rules in this article. Either class of rules is derived from first order digital nets construction of Polynomial lattices, as originally introduced by Niederreiter in [31]. We briefly recapitulate the essentials. In the following, let $b \geq 2$ be a prime number, $\mathbb{F}_b$ be the finite field with b elements, $\mathbb{F}_b[x]$ be the set of all polynomials over $\mathbb{F}_b$ and $\mathbb{F}_b((x^{-1}))$ be the set of all formal Laurent series of the form $\sum_{i=t}^{\infty} a_i x^{-i}$, $t \in \mathbb{Z}$, a_i in $\mathbb{F}_b$.

We can identify an integer $0 \leq n < b^m$ given by the b-adic expansion $n = n_0 + n_1 b + \cdots + n_{m-1} b^{m-1}$ and $n_0, \ldots, n_{m-1} \in \{0, 1, \ldots, b-1\}$, with its corresponding polynomial $n(x) \in \mathbb{F}_b[x]$ given by $n(x) = n_0 + n_1 x + \cdots + n_{m-1} x^{m-1}$, where we now view $n_0, \ldots, n_{m-1}$ as elements of $\mathbb{F}_b$.

Definition 2.1 (Polynomial lattice rule)**.** *Let $m \geq 2$ be an integer and $p \in \mathbb{F}_b[x]$ be a polynomial with $\deg(p) = m$. Let $\boldsymbol{q} = (q_1, \ldots, q_d)$ be a vector of polynomials over $\mathbb{F}_b$ with degree $\deg q_j < m$. We define the map $v_m : \mathbb{F}_b((x^{-1})) \to [0, 1)$ by*

$$v_m\left(\sum_{i=t}^{\infty} a_i x^{-i}\right) = \sum_{i=\max(1,t)}^{m} a_i b^{-i},$$

for $t \in \mathbb{Z}$. For $0 \leq n < b^m$, we put

$$y_n = \left(v_m\left(\frac{n(x)q_1(x)}{p(x)}\right), \ldots, v_m\left(\frac{n(x)q_d(x)}{p(x)}\right)\right) \in [0, 1)^d.$$

Then the set $P_m(\boldsymbol{q}, p) = \{y_0, y_1, \ldots, y_{b^m-1}\}$ is called a polynomial lattice point set and a quadrature rule $Q_{b^m,d}$, using this point set is called a polynomial lattice rule.

The following class of higher order lattice rule was proposed in [11] and developed in [9]. It mainly relies on an asymptotic expansion of the quadrature error that allows to apply Richardson extrapolation to the sequence of quadrature rules $(Q_{b^m,d})_{m\in\mathbb{N}}$, when applied to integrands of sufficient regularity.

Definition 2.2 (Extrapolated Polynomial lattice)**.** *Let $\alpha \geq 2$ be a natural number and $Q_{b^{m-\tau+1},d}, \tau \in \{1,\dots,\alpha\}$ be a set of Polynomial lattice rules with associated lattice point sets $P_{m-\tau+1}(\boldsymbol{q}_{m-\tau+1}, p_{m-\tau+1})$ respectively.*

We define, for suitable coefficients $a_\tau^{(\alpha)}$ defined in [11, Lemma 2.9], the quadrature rule

$$Q_{b^m,d}^{(\alpha)} = \sum_{\tau=1}^{\alpha} a_\tau^{(\alpha)} Q_{b^{m-\tau+1},d}$$

and we call it Extrapolated Polynomial Lattice *(EPL) rule of order α and lattice cardinality $N = b^{m-\alpha+1} + \ldots + b^m$.*

An alternative definition of higher order QMC rules is based on so-called *digit interlacing polynomial lattices*, as explained in the following definition, [12, 10, 14, 16].

Definition 2.3 (Interlaced Polynomial lattice)**.** *Let $\alpha \geq 2$ be a natural number and let $\mathcal{D}_\alpha \colon [0,1)^{d\alpha} \to [0,1)^d$ be defined by*

$$\mathcal{D}_\alpha(x_1,\dots,x_{d\alpha}) = (\mathcal{D}_\alpha(x_1,\dots,x_\alpha),\dots,\mathcal{D}_\alpha(x_{(d-1)\alpha+1},\dots,x_{d\alpha})),$$

and satisfying for any $x = \left(\sum_{i=1}^{\infty} x_{1,i}b^{-i},\dots,\sum_{i=1}^{\infty} x_{\alpha,i}b^{-i}\right) \in [0,1)^\alpha$, with $x_{\tau,i} \in \mathbb{F}_b$, $\forall \tau = 1,\dots,\alpha, \forall i \in \mathbb{N}$ and such that $(x_{\tau,i})_{i>i_0}$ is not constant equal to $b-1$ after any index $i_0 \in \mathbb{N}$,

$$\mathcal{D}_\alpha(x) = \sum_{i=1}^{\infty}\sum_{\tau=1}^{\alpha} x_{\tau,i} b^{-(\alpha(i-1)+\tau)} \in [0,1).$$

Then, a quadrature rule using $\mathcal{D}_\alpha(P_m((q_1,\dots,q_{\alpha d}),p))$ as point set is called Interlaced Polynomial Lattice *(IPL) rule of order α and cardinality $N = b^m$.*

In the next sections we will always work with the natural choice of basis $b = 2$ so that digit operations become bit operations in the numerical computations. Conversely to classical QMC point sets as Sobol' and Halton sequences, EPL and IPL are known to achieve dimension-independent error bounds (e.g. [12, 14, 10, 11]). The main reason for such improvement is that their construction can be done in a problem-dependent manner, exploiting the varying importance of the individual components in the vector $y = (y_1,\dots,y_d) \in Y$. For this purpose we define a set of positive *weights* $\boldsymbol{\gamma} := (\gamma_{\mathfrak{u}})_{\mathfrak{u}\subseteq\{1,\dots,d\}}$, that quantify the relative importance of the variables in the set $\mathfrak{u} \subseteq \{1,\dots,d\}$. In our discussion we will need QMC weights in SPOD[1] form,

$$\gamma_{\mathfrak{u}} := \sum_{\boldsymbol{\nu}\in\{1:\alpha\}^{|\mathfrak{u}|}} |\boldsymbol{\nu}|! \prod_{j\in\mathfrak{u}} \left(2^{\delta(\nu_j,\alpha)}\beta_j^{\nu_j}\right), \qquad (\beta_j)_j \in \ell^1(\mathbb{N}) \tag{4}$$

that allow for a wider class of applications compared to product weights $\gamma_{\mathfrak{u}} := \prod_{j\in\mathfrak{u}} \beta_j$, $\mathfrak{u} \subseteq \{1,\dots,d\}$. The weights play a key role in the *Component-By-Component* (CBC) construction of a generating vector $\boldsymbol{q}$ of a polynomial lattice. The details of the CBC construction and their fast version using FFT can be found in [12] for IPL rules and in [9] for EPL rules.

The CBC construction for EPL rules proves to be slightly cheaper in terms of operations: $\mathcal{O}\big((\alpha+d)N\log N + \alpha^2 d^2 N\big)$ for EPL rules versus $\mathcal{O}\big(\alpha d N \log N + \alpha^2 d^2 N\big)$ for IPL, both requiring

[1] SPOD: "Smoothness-driven, Product and Order Dependent", see [12].

$\mathcal{O}(\alpha dN)$ memory. On the other hand, IPL rules seem to give slightly better outcomes as shown in [11], leading to an overall comparable performance. One advantage of EPL over IPL, is that they allow for computable, asympotically exact, a-posteriori error estimates [9].

We will consider either class of point sets as the training sets of our deep learning algorithm that we describe below.

2.2 Deep Neural networks

We consider the following form of deep neural networks (DNNs) in this paper. Let $\sigma\colon \mathbb{R}^k \to \mathbb{R}^k$ be a nonlinear activation function for $k \in \mathbb{N}$, $L \in \mathbb{N}$ and a collection of weights $W^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$, and biases $b^{(\ell)} \in \mathbb{R}^{d_\ell}$ for $\ell = 1, \ldots, L$. We shall refer to the integer $L \geq 1$ as *depth of the DNN* and we denote the layer widths tuple $\mathfrak{d} := \{d_0, d_1, ..., d_L\} \in \mathbb{N}^{L+1}$ as *the architecture of the DNN*; that is, $d_0 = d \in \mathbb{N}$ is the input dimension and $d_L = N_{\text{obs}}$ denotes the output dimension in the definition of the underlying observable (1). We collect the set of parameters of the DNN in

$$\Theta := \Big\{ (W^{(\ell)}, b^{(\ell)}) \in \mathbb{R}^{d_\ell \times d_{\ell-1}} \times \mathbb{R}^{d_\ell} \colon \ell = 1, \ldots, L \Big\}. \tag{5}$$

Notice that Θ depends on the sequence $\mathfrak{d}$ which we do not indicate notationally. Also, we do not impose any further conditions (such as sparsity, clipping or quantization) on the weight matrices $W^{(\ell)}$ or on the bias vectors $b^{(\ell)}$ in (5).

For any $(W, b) \in \mathbb{R}^{d \times d'} \times \mathbb{R}^d$, $d, d' \in \mathbb{N}$, let $f_{W,b}\colon \mathbb{R}^{d'} \to \mathbb{R}^d$ denote nonlinear map which is defined by

$$f_{W,b}(y) := \sigma(Wy + b)\ . \tag{6}$$

For $\theta \in \Theta$, define the Neural network map $\phi_\theta^L : \mathbb{R}^d \to \mathbb{R}^{d_L}$ as

$$\phi_\theta^L(y) := W^{(L)}(f_{W^{(L-1)},b^{(L-1)}} \circ \cdots \circ f_{W^{(1)},b^{(1)}}(y)) + b^{(L)}\ . \tag{7}$$

We observe that the form (7) of the neural network corresponds to that of a fully-connected feed-forward multi-layer perceptron [18]. This form is very general and other specific types of neural networks, such as convolutional neural networks (CNNs), or sparsely connected DNNs, can be realized by imposing constraints on the structure of the weight matrices in Θ which are used in (7).

2.3 Loss functions

As we adopt the paradigm of supervised learning [18], DNNs of the form (7) will be *trained* to determine parameters $\theta \in \Theta$ in concrete applications. I.e., a parameter vector θ has to be found numerically such that the mismatch between the ground truth (underlying map g (1)) and the DNN is minimized over the *training set*. To this end, we define suitable *loss functions to quantify the mismatch* between the map g and its DNN surrogate.

In this context, we define for any suitable polynomial lattice point (or IPL) set $P_m(\boldsymbol{q}, p)$ of cardinality 2^m

$$\tilde{\mathcal{E}}_{T,m}(\theta) := \left(\frac{1}{2^m} \sum_{y \in P_m(\boldsymbol{q},p)} \left| g(y) - \phi_\theta^L(y) \right|^2 \right)^{1/2}, \quad \theta \in \Theta. \tag{8}$$

Next, we differentiate between two cases. First, for IPL training points, we consider the following partial loss function,

$$J(\theta) := \Big(\tilde{\mathcal{E}}_{T,m}(\theta) \Big)^2, \quad \theta \in \Theta. \tag{9}$$

Second, for the EPL training points, we need to work with suitable extrapolation of the partial loss functions

$$\mathcal{E}_T^{(\alpha)}(\theta) := \left| \sum_{\tau=1}^{\alpha} a_\tau^{(\alpha)} \left(\tilde{\mathcal{E}}_{T,m-\tau+1}(\theta) \right)^2 \right|^{1/2}, \theta \in \Theta, \tag{10}$$

with the coefficients $a_\tau^{(\alpha)}$ as in Definition 2.2. For notational simplicity, we confine ourselves to $\alpha = 2$, that reads $a_1^{(2)} = 2, a_2^{(2)} = -1$ so that

$$\mathcal{E}_T^{(2)}(\theta) = \left| 2\tilde{\mathcal{E}}_{T,m}^2(\theta) - \tilde{\mathcal{E}}_{T,m-1}^2(\theta) \right|^{1/2}. \tag{11}$$

We hasten to add, however, that all results generalize verbatim to higher digit interlacing order $\alpha > 2$, resp. to higher extrapolation orders. While the use of IPLs naturally results in positive coefficients in the loss function, EPLs, being obtained by Richardson type extrapolation formulas, and thus involve alternating signs of coefficients. In numerical DNN training, solving the optimization problem with alternating sign linear combinations can be computationally delicate. Hence, we define the loss function for the EPL training points by the following upper bound on (11),

$$J(\theta) = 2\tilde{\mathcal{E}}_{T,m}^2(\theta) + \tilde{\mathcal{E}}_{T,m-1}^2(\theta). \tag{12}$$

Note that the loss function (12) can be readily generalized for any $\alpha > 2$, replacing the coefficients $a_\tau^{(\alpha)}$ in (10) by their absolute values.

The goal of the training process in supervised learning is to find the parameter vector θ, for which the loss functions (9) or (12) are minimized. It is common in machine learning [18] to regularize the minimization problem for the loss function, i.e. we seek to find

$$\theta^* = \arg\min_{\theta \in \Theta} \left(J(\theta) + \lambda \mathcal{R}(\theta) \right). \tag{13}$$

Here, J is defined by either (9) (for IPL training points) or (12) (for EPL training points) and $\mathcal{R}$ is a *regularization* (penalization) term. A popular choice is to set $\mathcal{R}(\theta) = \|\theta_W\|_q^q$, with θ_W denoting the concatenated vector of all weights in (7) and either $q = 1$ (to induce sparsity) or $q = 2$. The parameter $0 \leq \lambda \ll 1$ balances the regularization term with the actual loss J.

The above minimization problem amounts to finding a minimum of a possibly non-convex function over a very high-dimensional parameter space. We follow standard practice in machine learning by either (approximately) solving (13) with a full-batch gradient descent algorithm or variants of mini-batch stochastic gradient descent (SGD) algorithms such as ADAM [24].

For notational simplicity, we denote the (approximate, local) minimum weight vector in (13) as θ^* and the underlying deep neural network $\phi_{\theta^*}^L$ will be our neural network surrogate for the underlying map g. The proposed algorithm for computing this neural network is summarized below.

Deep learning with Higher-order Quasi-Monte Carlo points (DL-HoQMC)

Inputs: Underlying map g (1), higher-order QMC training points such as EPL or IPL points, hyper-parameters and architecture of neural network (7) with depth L

Goal: Find neural network $\phi_{\theta^*}^L$ for approximating the underlying map g.

Step 1: Choose the training set $\mathcal{S}$ either as IPL or EPL QMC points. Evaluate $g(y)$ for all $y \in \mathcal{S}$ by a suitable numerical method.

Step 2: For an initial value of the weight vector $\overline{\theta}$, evaluate the neural network $\phi_{\overline{\theta}}^L$ (7), the loss function (13) and its gradients to initialize the (stochastic) gradient descent algorithm.

Step 3: Run a stochastic gradient descent algorithm till an approximate local minimum θ^* of (13) is reached. The map $\phi_{\theta^*}^L$ is the desired neural network approximating the map g.

3 Analysis of the DL-HoQMC algorithm

The objective of our analysis of the DL-HoQMC algorithm would be to estimate the so-called *generalization error* of this algorithm which is defined as $\mathcal{E}_G = \mathcal{E}_G(\theta^*)$, with

$$\mathcal{E}_G(\theta) = \left(\int_Y \left| g(y) - \phi_\theta^L(y) \right|^2 dy \right)^{1/2}, \quad \theta \in \Theta. \tag{14}$$

Throughout the rest of this work we adhere to the convention that the input data is appropriately scaled to the box $Y := [0,1]^d$ of Lebesgue measure $|Y| = 1$. Hence, the expressions (8), (10) are QMC quadrature approximations of $\mathcal{E}_G(\theta)$.

As is customary in machine learning [29, 1], we will estimate the generalization error in terms of *computable* training errors such as (8) for the IPL training points and (10) for the EPL training points. The key is to realize that the training errors (8) and (10) are the QMC quadratures for the integral in (14) defining generalization error. Thus, higher-order QMC approximation results (e.g. [13] and the references there and, for the presently proposed QMC integrations, [14, 12]) can be brought into play to estimate the so-called *generalization gap* i.e. difference between quadrature error and computable training errors.

Although the DL-HoQMC algorithm can be applied for approximating any underlying map g, it is clear from the higher-order QMC theory that *dimension independent higher-order approximation* results can only be obtained for integrands in loss functions that exhibit sufficient regularity with explicit, quantified dependence on the coordinate dimension. Our starting point in determining the appropriate function class for the underlying map as well as the approximating neural network is the *weighted unanchored Sobolev space* $\mathcal{W}_{d,\alpha,\boldsymbol{\gamma},1,\infty}$, which is defined by the set of integrand functions $F \in C^\infty([0,1]^d)$ equipped with the norm

$$\|F\|_{d,\alpha,\boldsymbol{\gamma},1,\infty} := \sup_{\mathfrak{u}\in\{1:d\}} \frac{1}{\gamma_\mathfrak{u}} \sum_{\mathfrak{v}\subseteq\mathfrak{u}} \sum_{\boldsymbol{\nu}_{\mathfrak{u}\setminus\mathfrak{v}}\in\{1:\alpha\}^{|\mathfrak{u}\setminus\mathfrak{v}|}} \int_{[0,1]^{|\mathfrak{v}|}} \left| \int_{[0,1]^{d-|\mathfrak{v}|}} \partial_y^{(\boldsymbol{\nu}_{\mathfrak{u}\setminus\mathfrak{v}},\alpha_\mathfrak{v})} F(y)\, \mathrm{d}y_{\mathfrak{v}^c} \right| \mathrm{d}y_\mathfrak{v}, \tag{15}$$

for a set of weights $\boldsymbol{\gamma}$ to be determined. Inspecting (15), it transpires that an error analysis will require estimates of higher order derivatives of input-output maps of DNNs in terms of (bounds on) NN weights, biases and activations. With the focus on deep NNs, we found the use of the multivariate chain rule in bounding $\|\hat{g}\|_{d,\alpha,\boldsymbol{\gamma},1,\infty}$ prohibitive due to the compositional structure of DNNs. Instead, we opt on using complex variable techniques based on quantified holomorphy to that end. Being derivative-free and preserved under composition, it appears naturally adapted to the analysis of DNNs. Using complex variable techniques mandates *holomorphic activation functions* in the DNNs, though.

Our goal is to find conditions on the underlying map g and on the weights $W^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and biases $b^{(\ell)} \in \mathbb{R}^{d_\ell}, \ell \in 1 \dots L$ of the DNN (7), which ensure that the integrand in (14) i.e. $\left| g - \phi_\theta^L \right|^2 \in \mathcal{W}_{d,\alpha,\boldsymbol{\gamma},1,\infty}$ uniformly with respect to the input dimension d. A sufficient condition relies on the concept of $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy with $0 < p < 1$ in the sense of [14, Theorem 3.1]. We define this concept below.

Definition 3.1 ($(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy on polytubes)**.** *[14, 6] Let X be a Banach space over $\mathbb{C}$, $\varepsilon > 0$, $0 < p < 1$ and let $\boldsymbol{\beta} \in \ell^p(\mathbb{N})$ be a non-negative sequence in $\mathbb{R}$. Define the* polytubes *$\mathcal{T}_{\boldsymbol{\rho}} = \bigtimes_{j\geq 1} \mathcal{T}_{\rho_j}$, where*

$$\mathcal{T}_{\rho_j} = \{ z \in \mathbb{C} \colon \operatorname{dist}(z, [-1,1]) < \rho_j - 1 \} \qquad \rho_j \in (1, \infty).$$

We say that a $\boldsymbol{\rho} = (\rho_j)_{j\in\mathbb{N}} \in (1,\infty)^{\mathbb{N}}$ is $(\boldsymbol{\beta}, \varepsilon)$-admissible, if there holds

$$\sum_{j\geq 1} (\rho_j - 1)\beta_j \leq \varepsilon \qquad \rho_j \in (1, \infty). \tag{16a}$$

Define $U := [-1,1]^{\mathbb{N}} \subset \mathcal{T}_{\boldsymbol{\rho}}$. A map $\phi\colon U \to X$ is called $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic if:

1. *for all $\boldsymbol{\rho}$ that is $(\boldsymbol{\beta}, \varepsilon)$-admissible, ϕ admits holomorphic extension with respect to each variable on the polytube $\mathcal{T}_{\boldsymbol{\rho}}$, and*
2. *there exists a family of open sets $\mathcal{O}_{\boldsymbol{\rho}} \supset \overline{\mathcal{T}_{\boldsymbol{\rho}}}$ and a constant $C_\varepsilon > 0$ independent of $\boldsymbol{\rho}$ such that there holds the uniform bound*

$$\sup_{z \in \mathcal{O}_{\boldsymbol{\rho}}} \|\phi(z)\|_X \leq C_\varepsilon. \tag{16b}$$

Definition 3.1 will be, in our case, accommodated by mapping $U = [-1,1]^{\mathbb{N}}$ to the domain $[0,1]^{\mathbb{N}}$, by means of an affine (in particular holomorphic) change of variables $\xi\colon y \mapsto (y+1)/2$. Moreover, the infinite dimensional parameter space is an artifact to obtain expression rate bounds that are free from the curse of dimensionality. We define the underlying map $\bar{g} \circ \xi : U \to \mathbb{R}^{N_{\text{obs}}}$ as a function with infinitely many parameters and we view the quantity of interest as its truncated version via

$$g \circ \xi((y_1, \ldots, y_d)) = \bar{g} \circ \xi((y_1, \ldots, y_d, 0, 0, \ldots)), \quad (y_1, \ldots, y_d) \in [-1,1]^d\,. \tag{17}$$

That is, we anchor the parameters after d to the center of their domain. Note that holomorphy, thus analyticity of the integrand, also allows to recover super-polynomial convergence (with respect to N) of the training error to the generalization error by training, for example, on tensorized Gauss-points $\Gamma_{n,d} = \big\{(y_1, \ldots, y_d) \in Y = [0,1]^d\colon y_i \in \Gamma_{n,1},\ \forall i\big\}$, where $\Gamma_{n,1}$ denote the Gauss quadrature points in the interval $[0,1]$. However, the implied convergence of $\mathcal{O}(\exp(-rN^{1/d}))$, with $r > 0$ independent of N and of d, deteriorates quickly as the parameter dimension d increases and results in practically infeasible training designs, for even moderate values of d. QMC training designs $\mathcal{S}$ only afford algebraic rates of convergence in terms of $N = \#(\mathcal{S})$ which are free from the curse of dimensionality, though.

In order to investigate the holomorphy of neural networks (7), we formally extend $f_{W,b}$ in (6) for complex sequences $z = y + i\eta \in \mathbb{C}^{\mathbb{N}}$, $y, \eta \in \mathbb{R}^{\mathbb{N}}$ and semiinfinite ("sequence") arrays $W = \big(W_{ij}\big)_{\substack{1 \leq i \leq d \\ 1 \leq j < \infty}} \in \mathbb{R}^{d \times \mathbb{N}}$, and we write

$$f_{W,b}(z) = \sigma(Wz + b).$$

We remark that we retain the network parameters W, b real valued. Verifications of the $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy of DNNs is considered in the following two sections.

3.1 Quantified Holomorphy of Shallow Neural Networks

We start the study of holomorphy of DNNs by first considering the shallow ($L = 2$ in (7)) but possibly wide neural network. Let $R > 0$, we use the notation

$$\mathcal{S}_R = \{z \in \mathbb{C}\colon |\Im(z)| < R\} \tag{18}$$

for the *strip* of width $2R$ around the real axis and $\mathcal{S}_R^d$ the d-fold cartesian product of $\mathcal{S}_R$ with itself. Then we have the following proposition.

Proposition 3.2. *Let $R > 0$, $\sigma\colon \mathcal{S}_R^k \to \mathbb{C}^k$ be holomorphic and $0 < \varepsilon < 2R$. Assume given a sequence $\boldsymbol{\beta} = (\beta_j)_{j \in \mathbb{N}} \in \ell^p(\mathbb{N})$, with some $0 < p < 1$, $b \in \mathbb{R}^k$ and $W = (W_{ij})_{\substack{1 \leq i \leq k \\ 1 \leq j < \infty}} \in \mathbb{R}^{k \times \mathbb{N}}$.*

Then, if $\max_{i=1,\ldots,k} |W_{ij}| \leq \beta_j\ \forall j \in \mathbb{N}$ there holds that $f_{W,b} \circ \xi\colon U \to \mathbb{R}^k$ is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic on polytubes.

Proof. Let $\boldsymbol{\rho}$ be a $(\boldsymbol{\beta}, \varepsilon)$-admissible sequence, $z = y + i\eta \in \mathcal{T}_{\boldsymbol{\rho}}$ and let $\tilde{z} := W\xi(z) + b$. Since W, b are real valued there holds

$$\Im(\tilde{z}) = \frac{1}{2} W\eta.$$

Note that $\eta_j = \Im(z_j) < \rho_j - 1$ for all $z \in \mathcal{T}_{\boldsymbol{\rho}}$; hence for all $i = 1, \dots, k$, we obtain

$$|\Im(\tilde{z}_i)| = \frac{1}{2}\left|\sum_{j=1}^{d} W_{ij}\eta_j\right| < \frac{1}{2}\sum_{j=1}^{d} \beta_j(\rho_j - 1) \leq \frac{\varepsilon}{2} < R. \tag{19}$$

Therefore $\tilde{z} \in \mathcal{S}_R^d$ and $f_{W,b} \circ \xi$ is holomorphic on $\mathcal{T}_{\boldsymbol{\rho}}$, so that we proved the first condition of Definition 3.1. To verify the second condition, let $\tilde{\boldsymbol{\rho}}$ be $(\boldsymbol{\beta}, (\varepsilon + 2R)/2)$-admissible and satisfy $\rho_j < \tilde{\rho}_j$, and set $\mathcal{O}_{\boldsymbol{\rho}} = \mathcal{T}_{\tilde{\boldsymbol{\rho}}}$ in Definition 3.1. Hence $\mathcal{T}_{\tilde{\boldsymbol{\rho}}} \supsetneq \overline{\mathcal{T}_{\boldsymbol{\rho}}}$ and

$$\sup_{z \in \mathcal{T}_{\tilde{\boldsymbol{\rho}}}} |f_{W,b}(z)| = \max_{z \in \overline{\mathcal{T}_{\tilde{\boldsymbol{\rho}}}}} |f_{W,b}(z)| \leq C_\varepsilon,$$

where we used that for all $z \in \overline{\mathcal{T}_{\tilde{\boldsymbol{\rho}}}}$, $\tilde{z} = Wz + b$ there holds $|\Im(\tilde{z}_i)| \leq (\varepsilon + R)/2$ for all $i = 1, \dots, k$ and hence $f_{W,b}(z)$ is bounded on this compact set. Note that C_ε depends on the output dimension k but is independent on the input dimension d. □

We now verify the assumptions of Proposition 3.2 for a selection of popular activation functions in the following examples.

Example 3.3. *Let σ given by $\sigma(y) := \left(\frac{1}{1+e^{-y_1}}, \dots, \frac{1}{1+e^{-y_d}}\right)$. The logistic function $\frac{1}{1+e^{-z}}$, $z \in \mathbb{C}$ is meromorphic with poles at $z = \pi i + 2ni\pi, n \in \mathbb{Z}$. In particular it is holomorphic on the open strip $\mathcal{S}_\pi$. Thus, σ is holomorphic on $\mathcal{S}_\pi^d$ by Hartogs Theorem and uniformly bounded on any compact set contained in $\mathcal{S}_\pi^d$.*

Example 3.4. *Let σ given by $\sigma(y) := (\tanh(y_1), \dots, \tanh(y_d))$. The function $\tanh(z)$, $z \in \mathbb{C}$ is meromorphic with poles at $z = \frac{\pi}{2}i + ni\pi, n \in \mathbb{Z}$. In particular it is holomorphic on $\mathcal{S}_{\frac{\pi}{2}}$. Thus, σ is holomorphic on the strip $\mathcal{S}_{\frac{\pi}{2}}^d$ by Hartogs Theorem and uniformly bounded on any compact set contained in $\mathcal{S}_{\frac{\pi}{2}}^d$.*

Example 3.5. *Let σ be the softmax function $\sigma(y) := \left(\frac{e^{y_1}}{\sum_j e^{y_j}}, \dots, \frac{e^{y_d}}{\sum_j e^{y_j}}\right)$. Each component $\frac{e^{z_k}}{\sum_j e^{z_j}}$, $z \in \mathbb{C}^d$ is also meromorphic. In particular, we show that it is holomorphic on the strip $\mathcal{S}_{\frac{\pi}{2}}^d$. In fact, writing $e^{z_k} = e^{y_k}(\cos(\eta_k) + i\sin(\eta_k))$, with $y_k, \eta_k \in \mathbb{R}$ yields*

$$\sum_{k=1}^{d} e^{z_k} = 0 \iff \begin{cases} \sum_{k=1}^{d} e^{y_k}\cos(\eta_k) = 0 \\ \sum_{k=1}^{d} e^{y_k}\sin(\eta_k) = 0 \end{cases},$$

but $\cos(\eta_k) > 0$ for all k due to $z_k \in \mathcal{S}_{\frac{\pi}{2}}$. Thus, $\sum_{k=1}^{d} e^{y_k}\cos(\eta_k) > 0$.

Lemma 3.6. *Let X, W be Banach spaces, $\boldsymbol{\beta} \in \ell^p(\mathbb{N})$ and $\varepsilon > 0$. Let $B_X(R) \subseteq X$ be the open ball on X centered at the origin with radius R. Assume that $\phi\colon U \to X$ is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic with uniform bound C_ε on a family of sets $\{\mathcal{O}_{\boldsymbol{\rho}} : \boldsymbol{\rho}$ is $(\boldsymbol{\beta}, \varepsilon)$-admissible$\}$ in the sense of Definition 3.1, with $\mathcal{O}_{\boldsymbol{\rho}} \supset \overline{\mathcal{T}_{\boldsymbol{\rho}}}$ and assume that $h\colon B_X(C_\varepsilon + \delta) \to W$ is holomorphic for some $\delta > 0$.*

Then, $h \circ \phi\colon U \to W$ is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic.

Proof. Fix a $\boldsymbol{\rho}$ that is $(\boldsymbol{\beta}, \varepsilon)$-admissible, and a $z \in \mathcal{O}_{\boldsymbol{\rho}}$ then

$$\|h \circ \phi(z)\|_W \leq \sup_{\|\psi\|_X \leq C_\varepsilon} \|h(\psi)\|_W = \sup_{\psi \in \overline{B_X(C_\varepsilon)}} \|h(\psi)\|_W \leq \hat{C}_\varepsilon$$

since h is bounded on this compact set. Thus

$$\sup_{z \in \mathcal{O}_{\boldsymbol{\rho}}} \|h \circ \phi(z)\|_W \leq \hat{C}_\varepsilon$$

and $\phi(\mathcal{T}_{\boldsymbol{\rho}}) \subseteq B_X(C_\varepsilon + \delta)$ implies that $h \circ \phi$ is well defined on $\mathcal{T}_{\boldsymbol{\rho}}$ and holomorphic as composition of holomorphic functions. □

Proposition 3.7. *Let $q \in 2\mathbb{N}$ be even, $\bar{g} \circ \xi : U \to \mathbb{C}^{d_1}$ be $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic for some $0 < p < 1$ and assume that $f_{W^{(1)}, b^{(1)}}$ satisfies the hypotheses of Proposition 3.2.*

Then, for any shallow neural network $\phi_\theta = W^{(2)} f_{W^{(1)}, b^{(1)}} + b^{(2)}$ there exist $C > 0, J \in \mathbb{N}$ independent of d such that

$$\|(g - \phi_\theta)^q\|_{d, \alpha, \boldsymbol{\gamma}, 1, \infty} \leq C$$

for some SPOD weights $\boldsymbol{\gamma}$ defined by

$$\gamma_{\mathfrak{u}} := \sum_{\boldsymbol{\nu} \in \{1:\alpha\}^{|\mathfrak{u}|}} |\boldsymbol{\nu}|! \prod_{j \in \mathfrak{u}} \left(2^{\delta(\nu_j, \alpha)} \tilde{\beta}_j^{\nu_j}\right) \tag{20}$$

where $\tilde{\beta}_j = 2^{\alpha+2} \|\beta\|_{\ell^1(\mathbb{N})} / \varepsilon \ \forall j \leq J$ and $0 < \tilde{\beta}_j \leq c\beta_j$ if $j > J$. Moreover, the constant $c > 0$ is independent of $d, \boldsymbol{\nu}, \boldsymbol{y}$ and j.

Proof. Since $f_{W^{(1)}, b^{(1)}} \circ \xi$ is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic by Proposition 3.2 and since affine transformations are holomorphic on the entire complex plane, it is easily verified that $(\bar{g} - \phi_\theta) \circ \xi$ is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic for the same sequence $\boldsymbol{\beta}$ and ε.

Furthermore, the assumption that q is an even integer ensures that the map $h \colon \mathbb{R}^{d_1} \to \mathbb{R}$, $x \mapsto |x|^q$ admits holomorphic continuation on $\mathbb{C}^{d_1}$. Hence, Lemma 3.6 and [14, Theorem 3.1, Remark 4.3] imply that there exists a finite set $E := \{1, \ldots, J\} \subset \mathbb{N}$ such that

$$\begin{aligned}
\|(g - \phi_\theta)^q\|_{d, \alpha, \boldsymbol{\gamma}, 1, \infty} &\leq C \sup_{\mathfrak{u} \in \{1:d\}} \frac{1}{\gamma_{\mathfrak{u}}} \sum_{\boldsymbol{\nu} \in \{1:\alpha\}^{|\mathfrak{u}|}} \boldsymbol{\nu}_{\mathfrak{u} \cap E}! \prod_{j \in \mathfrak{u} \cap E} \left(2^{\delta(\nu_j, \alpha)} \tilde{\beta}_j^{\nu_j}\right) |\boldsymbol{\nu}_{\mathfrak{u} \cap E^c}|! \prod_{j \in \mathfrak{u} \cap E^c} \left(2^{\delta(\nu_j, \alpha)} \tilde{\beta}_j^{\nu_j}\right) \\
&\leq C \sup_{\mathfrak{u} \in \{1:d\}} \frac{1}{\gamma_{\mathfrak{u}}} \sum_{\boldsymbol{\nu} \in \{1:\alpha\}^{|\mathfrak{u}|}} |\boldsymbol{\nu}|! \prod_{j \in \mathfrak{u}} \left(2^{\delta(\nu_j, \alpha)} \tilde{\beta}_j^{\nu_j}\right)
\end{aligned}$$

for a C independent of the dimension d but dependent on $\|\boldsymbol{\beta}\|_{\ell^1(\mathbb{N})}$ and ε, which proves the claim upon choosing the weights $\boldsymbol{\gamma}$ as in (20). □

3.2 Quantified Holomorphy of Deep Neural Networks

In this section we prove $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy for a class of DNNs, with $L \geq 3$ in (7), to which Proposition 3.7 can be extended. We verify $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy provided that a summability condition on the network weights holds.

Proposition 3.8. *Let $R, R' > 0$, $\sigma \colon \mathcal{S}_R^{d_\ell} \to \mathcal{S}_{R'}^{d_\ell}$ be holomorphic and let $0 < \varepsilon < 2R$ be given. Assume given a sequence $\boldsymbol{\beta} = (\beta_j)_{j \in \mathbb{N}} \in \ell^p(\mathbb{N})$, with some $0 < p < 1$, and with the network*

parameters $b^{(\ell)} \in \mathbb{R}^{d_\ell}$ $\forall \ell \geq 1$, $W^{(1)} \in \mathbb{R}^{d_1 \times \mathbb{N}}$ *and* $W^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ *for* $\ell \geq 2$. *Assume that* $\max_{i=1,\dots,d_1} \left|W_{ij}^{(1)}\right| \leq \beta_j$ $\forall j \in \mathbb{N}$ *and that for all* $\ell = 2, \dots, L-1$ *there holds*

$$\max_{i=1,\dots,d_\ell} \sum_{j=1}^{d_{\ell-1}} \left|W_{ij}^{(\ell)}\right| \leq \frac{R}{R'}. \tag{21}$$

Then, $\phi_\theta^L \circ \xi$ *is* $(\boldsymbol{\beta}, p, \varepsilon)$*-holomorphic on polytubes.*

Proof. By Proposition 3.2 $f_{W^{(1)},b^{(1)}} \circ \xi$ is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic. Moreover, (19) and the hypothesis that $\sigma \colon \mathcal{S}_R^{d_\ell} \to \mathcal{S}_{R'}^{d_\ell}$ give that the image of $\mathcal{T}_{\boldsymbol{\rho}}$ under the map $f_{W^{(1)},b^{(1)}} \circ \xi$ is contained in $\mathcal{S}_{R'}^{d_1}$ for all $(\boldsymbol{\beta}, \varepsilon)$-admissible sequences $\boldsymbol{\rho}$. Next, for fixed $\ell \in \{2, \dots, L-1\}$, we define $\tilde{z} = W^{(\ell)} z + b^{(\ell)}$. Then, since $W^{(\ell)}, b^{(\ell)}$ are real valued, (21) gives

$$\max_{i=1,\dots,d_\ell} |\Im(\tilde{z}_i)| = \max_{i=1,\dots,d_\ell} \left| \sum_{j=1}^{d_{\ell-1}} W_{ij}^{(\ell)} \Im(z_j) \right| \leq \frac{R}{R'} \max_{j=1,\dots,d_{\ell-1}} |\Im(z_j)|,$$

which implies that the affine transformations for $\ell = 2, \dots, L-1$ map $\mathcal{S}_{R'}^{d_{\ell-1}}$ to $\mathcal{S}_R^{d_\ell}$. Hence, the network map $\phi_\theta^L \circ \xi$ is well defined and holomorphic as composition of holomorphic functions on such $\mathcal{T}_{\boldsymbol{\rho}}$.

The second condition of $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy is verified analogously to Proposition 3.2: for a polyradius $\tilde{\boldsymbol{\rho}}$ that is $(\boldsymbol{\beta}, (\varepsilon + 2R)/2)$-admissible and satisfies $\rho_j < \tilde{\rho}_j$ there holds $\overline{\mathcal{T}_{\boldsymbol{\rho}}} \subseteq \mathcal{T}_{\tilde{\boldsymbol{\rho}}}$ and $\phi_\theta^L \circ \xi$ is continuous, hence bounded on $\overline{\mathcal{T}_{\tilde{\boldsymbol{\rho}}}}$ by some constant C_ε independent of $\boldsymbol{\rho}$. □

Remark 3.9. *It is of interest to track the dependence of the uniform bound* C_ε *on the architecture. The hypotheses of Proposition 3.8 set a constraint on the width of the inner layers of the network. Very wide networks are still possible, but at the price of smaller weights allowed in the analysis. On the other hand, no formal limitation is imposed on the biases* $b^{(\ell)}, \ell = 1, \dots, L$ *and on the output weight matrix* $W^{(L)}$*, although these parameters also contribute to the bound* C_ε*. Furthermore,* C_ε *is completely independent of the number of layers* L *provided that* (21) *holds for all the intermediate layers.*

The abstract assumptions on the activation function σ in Proposition 3.8 are next verified for a selection of widely used activation functions.

Example 3.10. *Let* σ *be the standard logistic function. Then for* $z = y + i\eta \in \mathbb{C}$ *there holds*

$$\begin{aligned}
\Im(\sigma(z)) &= \Im\left(\frac{1}{1+e^{-z}}\right) = \Im\left(\frac{1+e^{-y+i\eta}}{(1+e^{-y-i\eta})(1+e^{-y+i\eta})}\right) \\
&= \Im\left(\frac{1+e^{-y+i\eta}}{1+e^{-2y}+2e^{-y}\cos(\eta)}\right) = \frac{1}{1+e^{-2y}+2e^{-y}\cos(\eta)} \Im(1+e^{-y+i\eta}) \\
&= \frac{e^{-y}\sin(\eta)}{1+e^{-2y}+2e^{-y}\cos(\eta)}.
\end{aligned}$$

Thus if $|\eta| < \frac{\pi}{2}$*, this gives* $|\Im(\sigma(z))| < \frac{1}{2}$*. Hence,* $\sigma \colon \mathcal{S}_R^{d_\ell} \to \mathcal{S}_{R'}^{d_\ell}$ *is holomorphic for* $R = \frac{\pi}{2}$*,* $R' = \frac{1}{2}$ *and Proposition 3.8 applies.*

Example 3.11. *Let* σ *be the hyperbolic tangent function. Then for* $z = y + i\eta \in \mathbb{C}$ *there holds*

$$\Im(\sigma(z)) = \Im\left(\frac{1-e^{-2z}}{1+e^{-2z}}\right) = \Im\left(\frac{(1-e^{-2(y+i\eta)})(1+e^{-2(y-i\eta)})}{(1+e^{-2(y+i\eta)})(1+e^{-2(y-i\eta)})}\right)$$

$$= \Im\left(\frac{1 - e^{-4y} + 2ie^{-2y}\sin(2\eta)}{1 + e^{-4y} + 2e^{-2y}\cos(2\eta)}\right) = \frac{2e^{-2y}\sin(2\eta)}{1 + e^{-4y} + 2e^{-2y}\cos(2\eta)}.$$

Thus if $|\eta| < \frac{\pi}{4}$, this gives $|\Im(\sigma(z))| < 1$. Hence, $\sigma\colon \mathcal{S}_R^{d_\ell} \to \mathcal{S}_{R'}^{d_\ell}$ is holomorphic for $R = \frac{\pi}{4}$, $R' = 1$ and Proposition 3.8 applies.

Example 3.12. *Let σ be the softmax activation function. Then, for $z = y + i\eta \in \mathbb{C}^{d_\ell}$ and for all components $k = 1, \dots, d_\ell i$, there holds*

$$\begin{aligned}\Im(\sigma(z_k)) &= \Im\left(\frac{e^{z_k}}{\sum_j e^{z_j}}\right) = \frac{1}{\left|\sum_j e^{z_j}\right|^2}\Im\left(e^{z_k}\left(\sum_j e^{y_j}\cos(\eta_j) - i\sum_j e^{y_j}\sin(\eta_j)\right)\right)\\ &= \frac{e^{y_k}\sin(\eta_k)\sum_j e^{y_j}\cos(\eta_j) - e^{y_k}\cos(\eta_k)\sum_j e^{y_j}\sin(\eta_j)}{\left(\sum_j e^{y_j}\cos(\eta_j)\right)^2 + \left(\sum_j e^{y_j}\sin(\eta_j)\right)^2}.\end{aligned}$$

Thus, if $|\eta| < \frac{\pi}{4}$, this gives

$$|\Im(\sigma(z_k))| < \frac{\sqrt{2}e^{y_k}}{\frac{1}{2}\sum_j e^{y_j}} \le 2\sqrt{2}.$$

Hence, $\sigma\colon \mathcal{S}_R^{d_\ell} \to \mathcal{S}_{R'}^{d_\ell}$ is holomorphic for $R = \frac{\pi}{4}$, $R' = 2\sqrt{2}$ and Proposition 3.8 applies.

Remark 3.13. *To simplify the notation, we restricted the present discussion to the case when all the activation functions are of the same type. However, the proof of Proposition 3.8 remains valid verbatim if we use different activations $\sigma^{(\ell)}\colon \mathcal{S}_{R_\ell}^{d_\ell} \to \mathcal{S}_{R'_\ell}^{d_\ell}$ in each layer. Specifically, we must in this case pick $\varepsilon < 2R_1$ and replace (21) by*

$$\max_{i=1,\dots,d_\ell}\sum_{j=1}^{d_{\ell-1}}\left|W_{ij}^{(\ell)}\right| \le \frac{R_\ell}{R'_{\ell-1}}.$$

3.3 Estimate on the generalization gap

In order to estimate the generalization gap, we need a further *technical* assumption on the neural networks and the underlying map i.e. we assume that, for any fixed architecture $L, d_1, \dots, d_L$ the input function $\bar{g}$ satisfies,

$$0 < c(\bar{g}, (d_1, \dots, d_L)) := \inf_{d \ge 1}\inf_{\theta}\mathcal{E}_G(\theta) \tag{22}$$

where the second infimum is taken with respect to all network parameters θ that satisfy the hypothesis of Proposition 3.8.

We observe that this constant is related to the best DNN approximation with fixed architecture, see Remark 3.15 for more details. There holds the following theorem on the generalization gap.

Theorem 3.14. *Let $\alpha \in \mathbb{N}_{\ge 2}$. Assume that the activation function $\sigma\colon \mathcal{S}_R^{d_\ell} \to \mathcal{S}_{R'}^{d_\ell}$ and the set of parameters θ satisfy the hypotheses of Proposition 3.8. Let further $\bar{g} \circ \xi : U \to \mathbb{C}^{d_L}$ be $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic for some $p < \frac{1}{\alpha}$ and with $0 < \varepsilon < 2R$ and assume that (22) holds.*

Then there exists a constant $C > 0$ that is independent of d and N such that

$$\left|\mathcal{E}_G(\theta) - \mathcal{E}_T^{(\alpha)}(\theta)\right| \le CN^{-\alpha}\,.$$

Proof. Applying the arguments in Proposition 3.7, we obtain that $(\bar{g} - \phi_\theta^L)^2 \circ \xi$ is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic. Hence $(g - \phi_\theta^L)^2$ belongs to the QMC weighted space $\mathcal{W}_{d,\alpha,\boldsymbol{\gamma},1,\infty}$ and there holds the QMC approximation error convergence from [9, Theorem 2.5] which gives

$$\left|\mathcal{E}_G^2(\theta) - (\mathcal{E}_T^{(\alpha)})^2(\theta)\right| \leq C(\theta) N^{-\alpha} \tag{23}$$

for some constant $C(\theta) > 0$ that is independent of d and N. Thus, by (22), we obtain

$$\begin{aligned}\left|\mathcal{E}_G(\theta) - \mathcal{E}_T^{(\alpha)}(\theta)\right| &= \frac{\left|\mathcal{E}_G^2(\theta) - (\mathcal{E}_T^{(\alpha)})^2(\theta)\right|}{\mathcal{E}_G(\theta) + \mathcal{E}_T^{(\alpha)}(\theta)} \\ &\leq \frac{1}{\inf_\theta \mathcal{E}_G(\theta)} \left|\mathcal{E}_G^2(\theta) - (\mathcal{E}_T^{(\alpha)})^2(\theta)\right| \leq \frac{C(\theta)}{c(\bar{g}, (d_1, \ldots, d_L))} N^{-\alpha}.\end{aligned}$$

Here, the constant $c(\bar{g}, (d_1, \ldots, d_L))$ is bounded away from zero as in (22) uniformly with respect to N, d, (but not with respect to $(d_1, \ldots, d_L)$, cf. Remark 3.15) and the proof is complete. □

Remark 3.15. *Condition* (22) *is not in contradiction with the universal approximation theorem (e.g. [35]) since we fix a priori the width of the inner layers of the networks. Note that* (22) *is true whenever g cannot be expressed as a DNN exactly a.e. in any parameter space $[0,1]^d$ and, for fixed $L, d_1, \ldots, d_L$, there holds that $\inf_\theta \mathcal{E}_G(\theta)$ is monotonically increasing with respect to d. This is a reasonable assumption since high-dimensional input d reduces the approximation power of a DNN, when the rest of the architecture is fixed. In particular we expect that in most cases, for any parameter vector θ, the constant $\frac{1}{\mathcal{E}_G(\theta)}$ decreases when increasing d. On the other hand, in the case that* (22) *does not hold, for every δ there exists θ^* such that the approximation error $\mathcal{E}_A := \left\|g - \phi_{\theta^*}^L\right\|_{L^\infty([0,1]^d)} < \delta$ and hence there holds $\mathcal{E}_G, \mathcal{E}_T^{(\alpha)} < \delta$ for the parameter sequence θ^*.*

Theorem 3.14 implies the decay of $\mathcal{E}_G$ as

$$\mathcal{E}_G(\theta) \leq \frac{C(\theta)}{c(\bar{g}, (d_1, \ldots, d_L))} N^{-\alpha} + \mathcal{E}_T^{(\alpha)}(\theta), \tag{24}$$

for all θ satisfying the decay assumptions of Proposition 3.8. One natural question is what happens when the network size grows and $c(\bar{g}, (d_1, \ldots, d_L))$ becomes smaller as a consequence. We note that in this case we have the direct, but coarse, estimate

$$\left|\mathcal{E}_G(\theta) - \mathcal{E}_T^{(\alpha)}(\theta)\right| \leq \sqrt{\left|\mathcal{E}_G^2(\theta) - (\mathcal{E}_T^{(\alpha)})^2(\theta)\right|} \leq C(\theta)^{1/2} N^{-\alpha/2}.$$

Moreover, we have that $C(\theta)$ is proportional to (the square of) the constant C_ε of Definition 3.1 (see [14, Theorem 3.1 and Proposition 4.1]) and thus is independent on the architecture of ϕ_θ by Remark 3.9. This implies a guaranteed order of $N^{-\alpha/2}$ independent of the network architecture whenever Proposition 3.8 applies. However, we observe in the numerical experiment that rate α is attained in many situations.

3.4 Impact of Discretization on DNN surrogate fidelity

The key result, Theorem 3.14, provided an apriori bound on the generalization gap between the DNN and the DtO map of interest, g. In practice, and in the numerical experiments reported in Section 4 ahead, however, the DNN training algorithms will, in general, not be able to numerically access the exact map g. Instead, a numerical approximation g_δ is available, which stems from

some discretization scheme applied to a governing PDE, with discretization parameter $\delta \in (0,1]$. The DNN training process will numerically access g_δ. Completion of DNN training, therefore, will result in a DNN $\phi_{\theta^*}^L = \hat{g}_\delta$ instead of the DNN $\hat{g}$. With $|Y| \leq 1$,

$$\left(\int_Y |g(y) - \hat{g}_\delta(y)|^2 \,\mathrm{d}y\right)^{1/2} \leq \sup_{y\in Y} |g(y) - g_\delta(y)| + \left(\int_Y |g_\delta(y) - \hat{g}_\delta(y)|^2 \,\mathrm{d}y\right)^{1/2} \tag{25}$$

Assuming $\{g_\delta\}_{0<\delta\leq 1}$ to be convergent, i.e. (2) holds, it remains to control the *generalization error subject to discretized DtO maps*, $|g_\delta(y) - \hat{g}_\delta(y)|$. The proposed DNN training algorithm requires ensuring applicability of Theorem 3.14 to the discretized DtO map g_δ rather than to the truth g. To this end, we require *uniform (w.r. to $\delta \in (0,1]$) parametric $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy of $y \mapsto g_\delta(\xi(y))$ for some $p < \frac{1}{\alpha}$*, i.e. where the conditions in (16) are met with $\boldsymbol{\beta}, \varepsilon, C_\varepsilon$ independent of δ. Generally, for DtO maps $\bar{g} \circ \xi$ which are $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic, and for g_δ obtained by *discretizations which are uniformly* (w.r. to parametric inputs in a complex neighborhood of input data Y) *stable*, uniform $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy of the family $\{g_\delta\}_{0<\delta\leq 1}$ follows from that of the DtO map g. As a consequence, this stability allows to bound the generalization error in the second term of (25), as we did in (24) for the true map g. This uniform discretization stability required to apply Theorem 3.14 must be verified on a case-by-case basis.

For the benefit of the reader, we shall verify uniform $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy of FEM discretizations of a model PDE example in Section 4.3.1.

4 Numerical Experiments

4.1 Implementation

The implementation and testing of the DL-HoQMC algorithm is performed using the machine learning framework PyTorch [34]. Both the training and test sets (to approximate the generalization error (14) with QMC quadrature) are based on points generated with previously described extrapolated (and interlaced) lattice rules, where the size of the test set is chosen such that it significantly outnumbers the size of the training set, i.e. at least twice the amount of testing points than for the biggest training set. The training is performed using the ADAM optimizer [24] in a full-batch mode using a maximum of 20k epochs (learning steps). Hyperparameters, listed in Table 1 are selected via an ensemble training process, as described in [27]. The weights of the networks are initialized based on the so-called Xavier normal initializer [17], which is standard for training networks with sigmoid (or tanh) activation functions. We train the networks for EPL

Table 1: Hyperparameters for the ensemble training

Hyperparameter	values used for the ensemble
learning rate	10^{-4}
regularization parameter λ	$10^{-5}, 10^{-6}, 10^{-7}$
depth L	$2^2, 2^3, 2^4$
width d_j (constant)	$3 \times 2^1, 3 \times 2^2, 3 \times 2^3$
# initializations	2

points based on the upper bound (12) and plot the training error (11). For IPL points, however, we use the standard mean-square error (9) for training and plot the training error based on the L^2 norm (8). Furthermore, we emphasize that all results are based on quantities (i.e. $\mathcal{E}_G$, $\mathcal{E}_T$ and so on) which are averaged over the trained ensemble of networks. This implies a certain stability

towards the choice of the DNN architecture in order to obtain the desired rate of convergence. If not stated differently, we base the subsequent experiments on EPL training points. Rates of convergence for all subsequent experiments are estimated using the exponential of the least squares fit of the logarithmized data, where the first order coefficient of the least squares fit defines the rate of decay. The scripts to perform the ensemble training together with all data sets used in the experiments can be downloaded from *https://github.com/tk-rusch/DL-HoQMC*.

4.2 Function approximation

For our first numerical experiment, we approximate the following function:

$$g(y) = \frac{1}{1 + 0.5 \sum_{j=1}^{d} y_j j^{-2.5}}, \tag{26}$$

with the DL-HoQMC algorithm. Here, there is no additional discretization error as mentioned in Section 3.4. In Figure 1, we present the generalization error of the trained neural networks for different number of EPL training points together with the training error for $d = 50$, i.e. a 50-dimensional parameter space. We can see that while the training error is very low and does not seem to decay with increasing number of training points, the generalization error decays with a rate of around 2.1 in terms of $\#(\mathcal{S})$ and thus the generalization gap should decay at the same rate. This is indeed verified in Figure 2, where we observe a decay rate of approximately 2.3 for the generalization gap, which agrees with our theoretical predictions. Furthermore, we also train DNNs based on the interlaced lattice rule (IPL). Figure 2 also shows the generalization gap using IPL points. We can see that the generalization gap based on the IPL rule has approximately the same convergence rate as using the EPL rule, i.e. a convergence rate of around 2.3. Note that in this experiment a bigger learning rate is used, i.e. a learning rate of 10^{-3}.

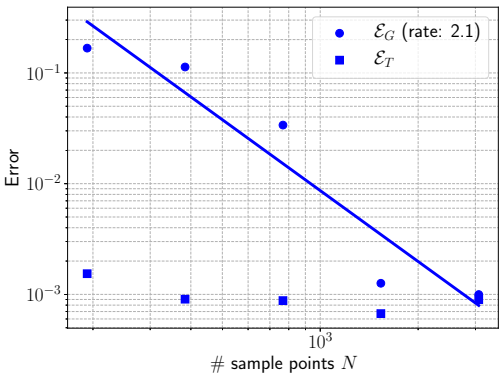

Figure 1: Generalization error $\mathcal{E}_G$ for approximating the function g (26) in $d = 50$ dimensions together with the training error $\mathcal{E}_T$.

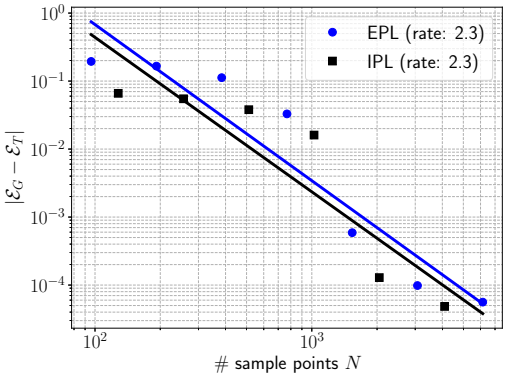

Figure 2: Generalization gap for approximating the function g (26) with parameter dimension $d = 50$ using both methods, the extrapolated polynomial lattice rule (EPL) as well as the interlaced polynomial lattice rule (IPL).

We also observe that the IPL and EPL based designs $\mathcal{S}$ are considerably more economic than deterministic, tensor product constructions: two points per coordinate imply then $\#(\mathcal{S}) = 2^d$, i.e. $2^{50} \simeq 10^{16}$ points. This is to be compared to $10^3 - 10^4$ points in the lattices used in our numerical

examples. Using $\mathcal{S}$ based on i.i.d random draws, a reduction of the (mean square) generalization gap by a factor of 10^4 as displayed in Figure 2 would mandate a 10^8 fold increase of $\#(\mathcal{S})$, i.e. about $\#(\mathcal{S}) \sim 10^{10}$ which is likewise prohibitive.

4.3 Elliptic parametric PDEs

For the second numerical experiment, we consider a test case that arises as a prototype for uncertainty quantification (UQ) in elliptic PDEs with uncertain coefficient [6, 7, 9]. On the bounded physical domain $D = (0,1)^2$ and with parameters $y \in \left[-\frac{1}{2}, \frac{1}{2}\right]^d$, we consider the following elliptic equation with homogeneous Dirichlet boundary conditions

$$\begin{cases} -\mathrm{div}\left(a(x,y)\nabla u(x,y)\right) = f(x) & x \in D \\ u(x,y) = 0 & x \in \partial D \end{cases}. \tag{27}$$

We choose a deterministic source $f(x) = 10x_1$, and the following observable

$$g(y) := \frac{1}{|\tilde{D}|}\int_{\tilde{D}} u(\cdot, y),$$

for $\tilde{D} = (0, \frac{1}{2})^2$. We denote the approximated observable, i.e. the observable based on a FEM approximation u_h of u, as g_h.

We assume that the diffusion coefficient $a(x,y)$ is affine-parametric, that is

$$a(x,y) = \bar{a}(x) + \sum_{j=1}^{d} y_j \psi_j(x). \tag{28}$$

Here we choose $\bar{a} \equiv 1$, $\psi_j(x) := \psi_{(k_1,k_2)}(x) = \frac{1}{(k_1^2+k_2^2)^\eta}\sin(k_1\pi x_1)\sin(k_2\pi x_2)$ and the ordering is defined by $(k_1,k_2) < (\bar{k}_1,\bar{k}_2)$ when $k_1^2 + k_2^2 < \bar{k}_1^2 + \bar{k}_2^2$ and is arbitrary when equality holds. In particular this ensures well-posedness of the problem and the asymptotic decay

$$\beta_j := \frac{\|\psi_j\|_{L^\infty(D)}}{2\,\mathrm{essinf}_{x\in D}\,\bar{a}(x)} \sim j^{-\eta} \tag{29}$$

which in turn implies that u, (hence g) is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic for any $p > \frac{1}{\eta}$, with the arguments from [6].

4.3.1 Uniform $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy

We start by sketching the basic arguments to show uniform $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy in the sense of Section 3.4, for the case of linear, second order elliptic PDEs in the divergence form (27) (28). First we rewrite the weak formulation of (27) for $z \in \mathbb{C}^{\mathbb{N}}$: denoting $V := H_0^1(D;\mathbb{C})$ the complexified Banach space of H_0^1, we seek

$$u \in V \;\text{ such that }\; \mathfrak{a}_z(u,v) = \langle f, v\rangle \quad \forall v \in V\,. \tag{30}$$

Here the complex extension of the parametric bilinear form is the sesquilinear form $\mathfrak{a}_z : V \times V \to \mathbb{C}$ defined by

$$\mathfrak{a}_z(w,v) := \int_D a(\cdot,z)\nabla w \cdot \overline{\nabla v}$$

and the brackets denote the pairing of V with its (topological) dual V^* over the field $\mathbb{C}$. Now, let $\boldsymbol{\rho}$ be $(\boldsymbol{\beta}, \varepsilon)$-admissible for the same (real) $\boldsymbol{\beta}$ as in (29) and $0 < \varepsilon < (1 - \|\boldsymbol{\beta}\|_{\ell^1(\mathbb{N})})/2$. Then there holds uniform coercivity of $\mathfrak{a}_z$ on V as $\forall z \in \mathcal{T}_{\boldsymbol{\rho}}$,

$$|a(x,z)| \geq \Re(a(x,z)) \geq \bar{a}(x) \left(1 - \sum_{j \geq 1} \Re(z_j)\beta_j\right) \geq (1 - \varepsilon - \|\boldsymbol{\beta}\|_{\ell^1(\mathbb{N})}) \operatorname{essinf} \bar{a} > 0. \tag{31}$$

Uniform (on $\mathcal{T}_{\boldsymbol{\rho}}$) continuity is readily verified, so that the complexified variational form of the PDE (27) is well-defined by the (complex version of the) Lax-Milgram lemma. Furthermore, (31) directly implies also uniform coercivity for any conforming FE discretization space $V_h \subset V$, including in particular the first order, continuous Lagrangian FEM with mesh width $0 < h \leq 1$, used in our numerical experiments. Therefore, the discrete problem obtained replacing V with V_h in (30) admits a unique solution $u_h \in V_h$ and there holds the uniform (with respect to $z \in \mathcal{T}_{\tilde{\boldsymbol{\rho}}}$ and h) bound

$$\sup_{0<h\leq 1} \sup_{z \in \mathcal{T}_{\tilde{\boldsymbol{\rho}}}} |g_h(z)| \lesssim \sup_{0<h\leq 1} \sup_{z \in \mathcal{T}_{\tilde{\boldsymbol{\rho}}}} \|u_h(\cdot, z)\|_V \leq \frac{\|f\|_{V^*}}{(1 - 2\varepsilon - \|\boldsymbol{\beta}\|_{\ell^1(\mathbb{N})}) \operatorname{essinf} \bar{a}} =: C_\varepsilon$$

for some $\tilde{\boldsymbol{\rho}} > \boldsymbol{\rho}$.

4.3.2 Results with EPL training points

Given the preceding section, it is clear that we can use a FEM simulation in order to provide training data for the deep neural networks approximating the underlying observable. For the first example, we set $\alpha = 2, \eta = 2.5$, and we run first order Lagrangian finite element (FEM) simulations with 131072 triangular elements for each QMC sample point, in order to generate the training data.

The resulting generalization gap (averaged over the ensemble) for approximating the observable g_h on parameter domains of dimension 16 and 32, with EPL training points, can be seen in Figure 3. We observe that for both choices of the parameter dimension d, the generalization gap not only has the same convergence rate of approximately 2.1, but also has almost the same absolute error. This is again in accordance with our theoretical findings, where we claimed the generalization gap to be independent of the dimension of the underlying problem. Note that we slightly increased the depth of the networks for the 32 dimensional case in order to have similar training error, i.e. instead of using networks with depths of $4, 8$ and 16, we use depths of $6, 10$ and 18 for the ensemble training procedure.

4.3.3 Holomorphy assumptions on the neural networks

We examine the assumptions in Proposition 3.8 which guarantee that a DNN is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic on polytubes in the following numerical experiment. As Proposition 3.8 does not rely on trained neural networks, we will focus on the case where the neural networks are untrained, i.e. the weights and biases are selected a priori and the generalization error (14) is evaluated on an increasing set of *test points*. To this end, we generate a neural network with fixed width and depth and randomly generated weights. Note that the weights are chosen such that the neural network does not satisfy the assumptions of Proposition 3.8. We also construct a second DNN by clamping the weights of the first network into certain ranges such that the resulting network satisfies all assumptions of Proposition 3.8 and thus is $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic on polytubes. Figure 4 shows the generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ using the non-clamped (non-$(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic) neural network as well as the clamped network ($(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic). We see that in the case of the clamped neural network, a second order convergence is obtained, while the non-clamped network has a convergence rate of a bit less than 1. This supports the sufficiency of the assumptions of Proposition 3.8 also numerically.

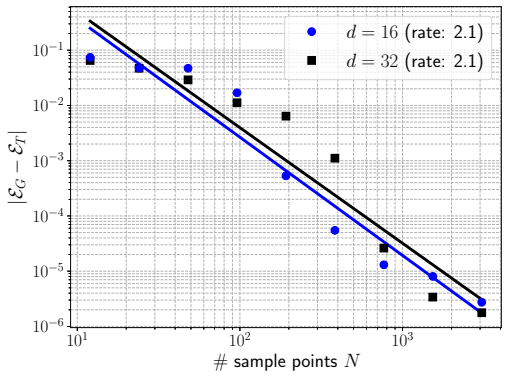

Figure 3: Generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ for approximating the observable g_h corresponding to the elliptic PDE (27) in 16 dimensions as well as in 32 dimensions.

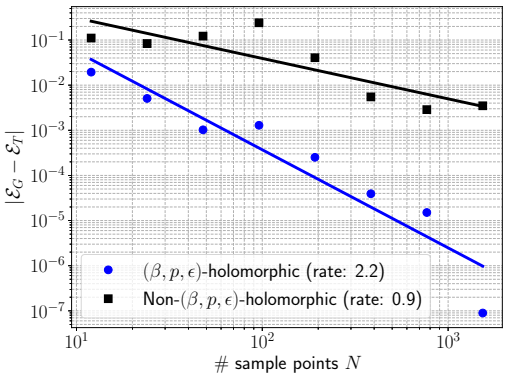

Figure 4: Generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ using untrained DNNs based on the observable g_h corresponding to the elliptic PDE (27) in 16 dimensions for both $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic as well as non-$(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic networks.

Despite this observation, we will not constrain the weights during training. Apart from the technical difficulty of doing so, we discover that most of the trained networks verified the assumption of Proposition 3.8 *a posteriori*, possibly on account of the role played by the regularization term in the loss function (13).

4.3.4 On the choice of activation functions

As the holomorphy of the DNN follows from holomorphy of the underlying activation function in (7), we expect that the ReLU activation, which is not holomorphic at the origin, will lead to DNNs with worse convergence rates of the generalization gap than the DNNs with a holomorphic activation function such as the hyperbolic tangent. To investigate this, in Figure 5, we present the numerical generalization error and the numerically estimated training error for approximating the observable g_h corresponding to the elliptic PDE (27) in 16 dimensions using tanh as well as ReLU as activations of the neural networks. We observe that using ReLU activations results in a slightly lower absolute generalization error than using tanh for small number of training samples. However, the rate of convergence using tanh activation appears to be close to 2.2, while the convergence rate for using ReLU activation is close to 0.9 and thus roughly one order lower than using tanh. We conclude that for this example, the absolute generalization error is lower for tanh than for ReLU when the number of training samples is increased. Figure 6 shows the generalization gap for the same experiment. We observe the same behavior as in Figure 5. This is to some extent expected, as the training error in Figure 5 is almost constant for different numbers of training points for both tanh activation as well as for ReLU activation.

4.3.5 On the discretization of the training data

As already mentioned in a previous section, in practice the training data is not provided by exact values of the DtO map but rather based on some approximations, for instance based on FEM approximations of the PDE. In the following, we conduct an experiment to analyze the effect of the approximation of the training data on the generalization of the DNNs numerically. To this

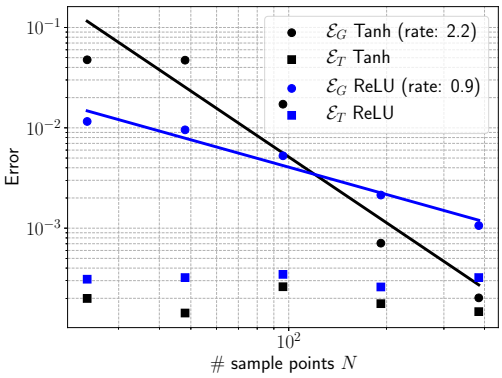

Figure 5: Generalization error $\mathcal{E}_G$ together with the training error $\mathcal{E}_T$ for approximating the observable g_h corresponding to the elliptic PDE (27) in 16 dimensions using tanh as well as ReLU as activations of the neural networks.

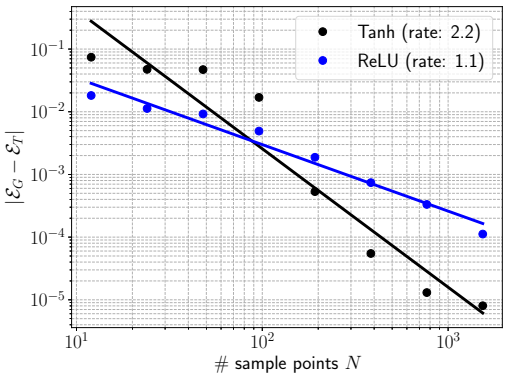

Figure 6: Generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ for approximating the observable g_h corresponding to the elliptic PDE (27) on a parameter domain of 16 dimensions using tanh as well as ReLU as activations of the neural networks.

end, we consider the observable g_h corresponding to the elliptic PDE (27) in 16 dimensions. We compute the training points of the observable based on three different FEM meshes, i.e. g_{h_1} with 32768 elements, g_{h_2} with 131072 elements and g_{h_3} with 524288 elements. Note that we use first order Lagrange elements in all cases and that the other experiments are all based on g_{h_2}. Figure 7 shows the generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ based on all three data sets corresponding to different meshes. The generalization error $\mathcal{E}_G$, for each set of training data, based on a different number of elements, is calculated using test points corresponding to different ground truths, i.e. g_{h_i}, for $i = 1, 2, 3$. We can see that in all cases the convergence rate is almost the same and in particular of second order. We can conclude that as long as the FEM discretization is stable, the impact of the approximation error in the training data on the DL-HoQMC algorithm is negligible.

4.3.6 On the holomorphy of the cost function

So far, we exclusively considered the case where $\mathcal{E}_G$ and $\mathcal{E}_T$ are based on the L^2 norm (or any $L^{2\mathbb{N}}$ norm) in order to ensure holomorphy of the integrand in (14). Hence, one question that naturally arises is if this condition is not only sufficient but also necessary. To test this numerically, we change the definition of $\mathcal{E}_G$ and $\mathcal{E}_T$ by using L^q norms, where q is an odd natural number. Figure 8 shows the computed generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ based on two different L^q norms, i.e. L^1 and L^3. We can see that a convergence of roughly second order is still obtained in both cases. Thus, this experiment suggests that the holomorphy condition on the cost function is sufficient but, apparently, not necessary in order to obtain second order convergence using the DL-HoQMC algorithm.

4.4 Linear Parabolic PDEs

For the next experiment, we consider a time-dependent problem defined by the following parametric parabolic PDE, which arises in the context of UQ for parabolic PDEs with uncertain initial data.

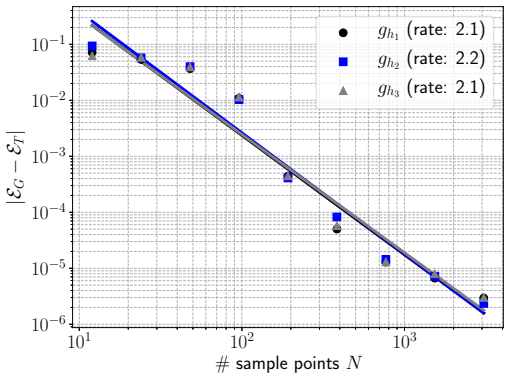

Figure 7: Generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ for approximating observables g_{h_i} corresponding to the elliptic PDE (27) in 16 dimensions and based on three different FEM meshes.

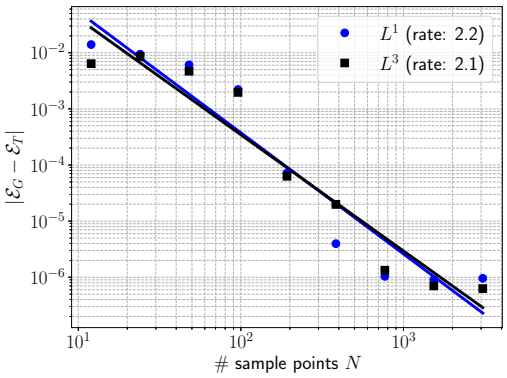

Figure 8: Generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ for approximating observable g_h corresponding to the elliptic PDE (27) in 16 dimensions using different L^q norms in the cost function, i.e. L^1 and L^3.

For all $y \in \left[-\frac{1}{2}, \frac{1}{2}\right]^d$,

$$\begin{cases} \partial_t u(x,t,y) - \operatorname{div}\left(\nabla u(x,t,y)\right) = f(x,t) & x \in D, t \in [0,T] \\ u(x,t,y) = 0 & x \in \partial D, t \in [0,T] \\ u(x,0,y) = u_0(x,y) & x \in D \end{cases}. \tag{32}$$

Here the divergence div and the gradient ∇ are understood only with respect to the variables x. We use $D = (0,1)^2$ and the uncertain initial data u_0 is parametrized by

$$u_0(x,y) = \exp\left(100 \sum_{j=1}^{d} y_j \psi_j(x)\right) - 1,$$

where we use the same sin expansion for ψ_j as in the elliptic case in the previous section. As for the source term f, we select a moving localized source as

$$f(x,t) = 100 \exp(-20(x_1 - t)^2 - 20(x_2 - t)^2)$$

and for an observable we choose $g\colon y \mapsto \frac{1}{|\tilde{D}|}\int_{\tilde{D}} u(\cdot, T, y)$. Again, we select $\alpha = 2, \eta = 2.5, d = 16$; then we run FEM simulations with 131072 elements for each QMC sample point, here corresponding to EPL points, and final time $T = 0.5$. The time integration is by a backward Euler method, with constant time-step $\Delta t = 10^{-3}$. We denote this approximation again as g_h. Figure 9 shows the generalization error $\mathcal{E}_G$ together with the training error $\mathcal{E}_T$ of the averaged trained ensemble for approximating the observable g_h corresponding to the time-evolution equation (32) for $d = 16$ dimensions. We can see that the decay of the generalization error is approximately of second order. Additionally, Figure 10 shows the generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$, which also decays with a rate of almost 2. This is again in accordance with our theoretical findings and thus demonstrates that the proposed algorithm can successfully be applied to UQ for time-dependent PDEs.

Note that compared to the previous experiments, the numerical range of the observable is rather small in most of the parameter domain $\left[-\frac{1}{2}, \frac{1}{2}\right]^d$, with the exception of values of y towards the

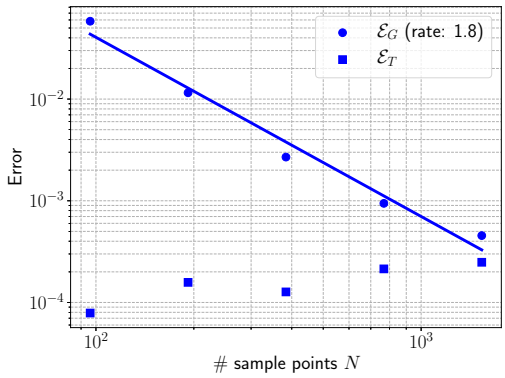

Figure 9: Generalization error $\mathcal{E}_G$ for approximating the observable g_h corresponding to the linear parabolic PDE (32) in 16 dimensions together with the training error $\mathcal{E}_T$.

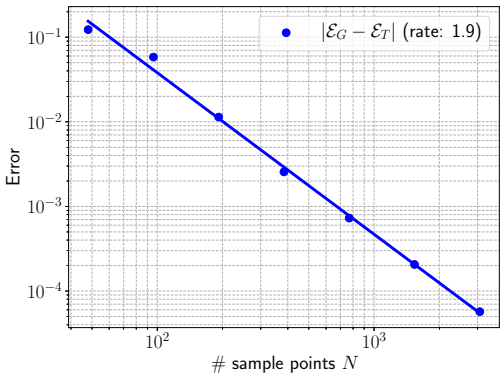

Figure 10: Generalization gap $|\mathcal{E}_G - \mathcal{E}_T|$ for approximating the observable g_h corresponding to the linear parabolic PDE (32) in 16 dimensions.

corner $(\frac{1}{2}, \ldots, \frac{1}{2})$, where the range is much larger. The presence of such localized strong variations in the samples makes the training particularly difficult. Hence, in order to allow the DNNs to learn such outliers, we reduce the regularization parameter λ in Table 1 to $10^{-7}, 10^{-8}, 10^{-9}$. Additionally, we increase the number of epochs (equivalent to training iterations in our full-batch mode) to 100k.

5 Discussion

A diverse set of problems in scientific computing involving PDEs, such as uncertainty quantification (UQ), (Bayesian) inverse problems, optimal control and design are of the *many query* type. I.e., their numerical solution requires a large number of calls to some underlying numerical PDE solver. As PDE solvers, particularly in multiple space dimensions, could be expensive, the numerical solution of such many query problems can be prohibitively expensive. Hence, the design of efficient and accurate surrogate models is of great importance as they can make such many query problems computationally tractable in engineering applications.

In this article, we propose a surrogate algorithm based on deep neural networks, to approximate observables of interest, on solution families of PDE models. The key novelty of our algorithm *lies in the use of deterministic families of training points, defined by polynomial lattices, that arise in the context of high-order Quasi-Monte Carlo (HoQMC) integration methods.* In particular, we employ training points that correspond to quadrature points defined by extrapolated polynomial lattice (EPL) points ([11, 9]) or interlaced polynomial lattice (IPL) points ([12] and the references therein). The resulting DL-HoQMC algorithm possesses the following attribute; as long as the underlying map (observable) is holomorphic in a precise sense, defined in Section 3, and the underlying deep neural networks are such that the activation function is similarly holomorphic as well as the weights of the neural network satisfy the conditions of Proposition 3.8, we prove that the resulting generalization gap (i.e., the difference between generalization and training errors),

- decays quadratically (and, given sufficiently small summability exponent $p \in (0, 1)$ and sufficient large QMC integration order, at arbitrary high-order) with respect to the number of training points,

- the rate of convergence is *independent of the dimension d of the parameter domain.*

Thus, at least for a (large) class of maps, the proposed algorithm has a significantly higher rate of decay of the generalization gap (in terms of the size of training set) than standard deep learning algorithms that use random training points, while at the same time overcoming the *curse of dimensionality*. This is afforded by suitable sparsity in the DtO map, as ensured here via quantified parametric holomorphy. This removes a major bottleneck in the use of deep learning algorithms for regression problems in scientific computing, where the use of i.i.d random training points requires large computational resources on account of the slow rate of convergence. See, e.g., [30] and references therein.

We present numerical experiments, involving model problems for both elliptic and parabolic PDEs, that validate the proposed theory and demonstrate the ability of the DL-HoQMC algorithm to approximates observables of PDEs in very high dimensions, efficiently.

Hence, the proposed algorithm promises to provide efficient DNN surrogates for parametric PDEs with high dimensional state- and / or parameter spaces. Nevertheless, it is important to point out the following caveats:

- The estimates we prove in Theorem 3.14 are on the generalization gap and we do not attempt to estimate the training error in any way. However, this is standard practice in machine learning [29] as estimating the training error that arises from using stochastic gradient descent for a (highly) non-convex very high dimensional optimization problem is quite challenging.

- The estimate on the generalization gap requires holomorphy of the DtO map and of the DNN emulating this DtO map. In Proposition 3.8, we provide sufficient conditions on the weights to the network in order to ensure this holomorphy. However, in practice and as pointed out in Section 4, we do not explicitly ensure that the trained weights satisfy these bounds. These bounds are verified *a posteriori* imply a subtle role for regularization of the loss function (13) that needs to further elucidated.

- We measured the generalization gap in integral norms with even $q \in 2\mathbb{N}$, in order to allow for analytic continuation of the parametric integrand, which was a necessary ingredient to allow for QMC integration. Numerical experiments indicated, however, a similar generalization gap decay also for $q = 1, 3$, indicating that the condition $q \in 2\mathbb{N}$ is not necessary.

- In the numerical examples reported in Section 4, we considered linear, elliptic and parabolic partial differential equations, subject to either affine-parametric uncertain input data, or subject to holomorphic maps (specifically, exp()) of such inputs. It was proved in e.g. [6] for nonlinear, parametric holomorphic operator equations that such inputs, with summability conditions of the sequence $\boldsymbol{\beta}$ following from (assumed) decay conditions for the elements of the sequence $\{\psi_j\}_{j\geq 1}$, imply $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphy of the parametric solution manifolds. Further examples of $(\boldsymbol{\beta}, p, \varepsilon)$-holomorphic DtO maps include time-harmonic electromagnetic scattering (e.g. [2]) in parametric scatterers, viscous, incompressible fluids in uncertain geometries (e.g. [8]), boundary integral equations on parametric boundaries (e.g. [21]), and parametric, dynamical systems described by large systems of initial-value ODEs (e.g. [39] and, for a proof of parametric holomorphy of solution manifolds, [20]), and [22] for DtO maps for Bayesian Inverse Problems for PDEs.

  The presently developed results being based only on quantified, parametric holomorphy on the DtO maps will be directly applicable also to these settings.

Finally, we point out that the DL-HoQMC algorithm can be used to accelerate the computations for UQ [27] and PDE constrained optimization [28], among other many-query problems for DtO maps of systems governed by PDEs.

Acknowledgements.

The research of SM and TKR was partially supported by European Research Council Consolidator grant ERCCoG 770880: COMANFLO.

References

[1] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263, 2018.

[2] Ruben Aylwin, Carlos Jerez-Hanckes, Christoph Schwab, and Jakob Zech. Domain uncertainty quantification in computational electromagnetics. *SIAM/ASA J. Uncertainty Quantification*, 8(1):301–341, 2020.

[3] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen. Solving stochastic differential equations and kolmogorov equations by means of deep learning. Preprint, available as arXiv:1806.00421v1.

[4] Helmut Bölcskei, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM J. Math. Data Sci.*, 1(1):8–45, 2019.

[5] Russel E Caflisch. Monte-Carlo and Quasi-Monte Carlo Methods. *Acta Numerica*, 7:1–49, 1998.

[6] Abdellah Chkifa, Albert Cohen, and Christoph Schwab. Breaking the curse of dimensionality in sparse polynomial approximation of parametric PDEs. *J. Math. Pures Appl. (9)*, 103(2):400–428, 2015.

[7] Albert Cohen, Ronald DeVore, and Christoph Schwab. Convergence rates of best N-term Galerkin approximations for a class of elliptic sPDEs. *Found. Comput. Math.*, 10(6):615–646, 2010.

[8] Albert Cohen, Christoph Schwab, and Jakob Zech. Shape Holomorphy of the stationary Navier-Stokes Equations. *SIAM J. Math. Analysis*, 50(2):1720–1752, 2018.

[9] J. Dick, M. Longo, and Ch. Schwab. Extrapolated lattice rule integration in computational uncertainty quantification. Technical Report 2020-29, Seminar for Applied Mathematics, ETH Zürich, Switzerland, 2020.

[10] Josef Dick, Robert N. Gantner, Quoc T. Le Gia, and Christoph Schwab. Higher order quasi-Monte Carlo integration for Bayesian PDE inversion. *Comput. Math. Appl.*, 77(1):144–172, 2019.

[11] Josef Dick, Takashi Goda, and Takehito Yoshiki. Richardson extrapolation of polynomial lattice rules. *SIAM J. Numer. Anal.*, 57(1):44–69, 2019.

[12] Josef Dick, Frances Y. Kuo, Quoc T. Le Gia, Dirk Nuyens, and Christoph Schwab. Higher order QMC Petrov-Galerkin discretization for affine parametric operator equations with random field inputs. *SIAM J. Numer. Anal.*, 52(6):2676–2702, 2014.

[13] Josef Dick, Frances Y. Kuo, and Ian H. Sloan. High-dimensional integration: the quasi-Monte Carlo way. *Acta Numer.*, 22:133–288, 2013.

[14] Josef Dick, Quoc T. Le Gia, and Christoph Schwab. Higher order quasi-Monte Carlo integration for holomorphic, parametric operator equations. *SIAM/ASA J. Uncertain. Quantif.*, 4(1):48–79, 2016.

[15] Alexander I. J. Forrester, Andras Sobester, and Andy J. Keane. *Engineering design via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

[16] Robert N. Gantner and Christoph Schwab. Computational higher order quasi-Monte Carlo integration. In *Monte Carlo and quasi-Monte Carlo methods*, volume 163 of *Springer Proc. Math. Stat.*, pages 271–288. Springer, [Cham], 2016.

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[20] Markus Hansen and Christoph Schwab. Sparse adaptive approximation of high dimensional parametric initial value problems. *Vietnam Journal of Mathematics*, 41(2):181–215, 2013.

[21] F. Henriquez and Ch. Schwab. Shape Holomorphy of the Calderón Projector for the Laplacean in $\mathbb{R}^2$. Technical Report 2019-43, Seminar for Applied Mathematics, ETH Zürich, Switzerland, 2019.

[22] L. Herrmann, Ch. Schwab, and J. Zech. Deep ReLU Neural Network Expression Rates for Data-to-QoI Maps in Bayesian PDE Inversion. Technical Report 2020-02, Seminar for Applied Mathematics, ETH Zürich, Switzerland, 2020.

[23] Jan S. Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified reduced basis methods for parametrized partial differential equations*. SpringerBriefs in Mathematics. Springer, Cham; BCAM Basque Center for Applied Mathematics, Bilbao, 2016. BCAM SpringerBriefs.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[26] Kjetil O Lye, Siddhartha Mishra, and Roberto Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms. *European Journal of Applied Mathematics*, 2020.

[27] Kjetil O Lye, Siddhartha Mishra, and Deep Ray. Deep learning observables in computational fluid dynamics. *Journal of Computational Physics*, page 109339, 2020.

[28] K.O. Lye, S. Mishra, P. Chandrasekhar, and D. Ray. Iterative surrogate model optimization (ismo): An active learning algorithm for pde constrained optimization with deep neural networks. Preprint, available from ArXiv 2008.05730v1, 2020.

[29] A. Rostamizadeh M. Mohri and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[30] S. Mishra and T. Konstantin Rusch. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. Preprint, available as arXiv:2005.12564, 2020.

[31] Harald Niederreiter. Low-discrepancy point sets obtained by digital constructions over finite fields. *Czechoslovak Math. J.*, 42(117)(1):143–166, 1992.

[32] Joost A. A. Opschoor, Philipp C. Petersen, and Christoph Schwab. Deep ReLU Networks and High-Order Finite Element Methods. Technical Report 2019-07 (revised), Seminar for Applied Mathematics, ETH Zürich, 2019. (to appear in Analysis and Applications (Sing.) 2020).

[33] Joost A. A. Opschoor, Christoph Schwab, and Jakob Zech. Exponential ReLU DNN expression of holomorphic maps in high dimension. Technical Report 2019-35, Seminar for Applied Mathematics, ETH Zürich, 2019. (to appear in Constr. Approximation (2021)).

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Workshop Proceedings of Neural Information Processing Systems*, 2017.

[35] Allan Pinkus. Approximation theory of the MLP model in neural networks. In *Acta numerica, 1999*, volume 8 of *Acta Numer.*, pages 143–195. Cambridge Univ. Press, Cambridge, 1999.

[36] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations*, volume 92 of *Unitext*. Springer, Cham, 2016. An introduction, La Matematica per il 3+2.

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

[38] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[39] F. Regazzoni, L. Dedè, and A. Quarteroni. Machine learning for fast and reliable solution of time-dependent differential equations. *J. Comput. Phys.*, 397:108852, 26, 2019.