Generic bounds on the approximation error for physics-informed (and) operator learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a very general framework for deriving rigorous bounds on the approximation error for physics-informed neural networks (PINNs) and operator learning architectures such as DeepONets and FNOs as well as for physics-informed operator learning. These bounds guarantee that PINNs and (physics-informed) DeepONets or FNOs will efficiently approximate the underlying solution or solution operator of generic partial differential equations (PDEs). Our framework utilizes existing neural network approximation results to obtain bounds on more involved learning architectures for PDEs. We illustrate the general framework by deriving the first rigorous bounds on the approximation error of physics-informed operator learning and by showing that PINNs (and physics-informed DeepONets and FNOs) mitigate the curse of dimensionality in approximating nonlinear parabolic PDEs.

1 Introduction

The efficient numerical approximation of partial differential equations (PDEs) is of paramount importance as PDEs mathematically describe an enormous range of interesting phenomena in the sciences and engineering. Machine learning techniques, particularly deep learning, are playing an increasingly important role in this context. For instance, given their universal approximation properties, deep neural networks serve as ansatz spaces for supervised learning of a variety of (parametric) PDEs [17, 66, 37, 50, 51] and references therein. In this setting, large amounts of training data might be required. However, this data is often acquired from expensive computer simulations or physical measurements [50], necessitating the design of learning frameworks that work with limited data. *Physics-informed neural networks (PINNs)*, proposed by [16, 39, 38] and popularized by [63, 64], are a prominent example of such a learning framework as the residual of the underlying PDE is minimized within the class of neural networks and in principle, little (or even no) training data is required. PINNs and their variants have proven to be a very powerful and computationally efficient framework for approximating solutions to PDEs, [65, 48, 52, 61, 72, 31, 32, 55, 56, 57, 2] and references therein.

Often in the context of PDEs, one needs to approximate the underlying solution operator that maps one infinite-dimensional function space into another [36]. As neural networks can only map between finite dimensional spaces, a new field of *operator learning* is emerging wherein novel learning frameworks need to be designed in order to approximate operators. These include deep operator networks (DeepONets) [8, 46] and their variants as well as *neural operators* [36], which generalize neural networks to this setting. A variety of neural operators have been proposed, see [42, 43] but arguably, the most efficient form of neural operators is provided by the so-called *Fourier neural operators* (FNOs) [41]. Both DeepONets and FNOs have been very successfully deployed in scientific computing [47, 53, 7, 45, 44, 62] and references therein. Finally, one can combine PINNs and operator learning to design physics-informed DeepONets/FNOs [70, 44, 69, 20].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

From a theoretical perspective, one needs to provide a rigorous guarantee that the learning framework can approximate the underlying PDE solution (operator) to desired accuracy. More precisely, given an error tolerance $\varepsilon > 0$, we need to rigorously prove that the approximation error of the neural network (operator) can be made smaller than ε. For *efficient approximation*, one has to further ensure that the computational complexity (measured in terms of the size) of the learning architecture grows at most *polynomially* in ε^{-1}. In particular, *exponential* growth has to be ruled out. As neural networks, DeepONets and FNOs are all *universal approximators* [12, 8, 46, 40, 35] of the underlying functions or operators, it is possible to show that the approximation error can be made as small as desired. However, these results do not guarantee efficient approximation as the underlying network size could still grow exponentially with decreasing error, see [73] for neural networks in very high spatial dimensions, [40] for DeepONets and [35] for FNOs. Hence, the real theoretical challenge in this context lies in proving efficient approximation results for the different learning architectures in scientific computing. Such efficient approximation results have mostly been obtained for neural networks in the supervised learning setting e.g. [21, 33, 29, 4, 59] and references therein. In contrast, there is a relative scarcity of such efficient approximation results for PINNs and operator learning with notable exceptions being [23, 13, 14] (for PINNs), [40] (for DeepONets) and [35] (for FNOs). Moreover, the underlying proofs in these works are often on a case-by-case basis and the overall abstract structure is not clearly identified. Finally, no similar rigorous approximation results for physics informed operator learning are available till date.

This paucity of generic efficient approximation results for PINNs and operator learning for PDEs sets the stage for the current paper where our main contribution is to propose a very general *framework* (Section 3) for proving bounds on the approximation error for space-time neural networks, PINNs, DeepONets, FNOs and physics-informed DeepONets and FNOs for very general PDEs. Consequently, we obtain the first rigorous bounds for physics-informed operator learning in literature. Our framework is based on the observation that error estimates for different types of neural network architectures can all be obtained from one another. As error estimates for neural network approximations of PDE solutions at a fixed time are the easiest to obtain, and hence constitute the largest proportion of currently available estimates, we devote particular attention to demonstrating how these available estimates can be used to obtain novel bounds on the approximation error for space-time networks, PINNs and (physics-informed) operator learning. Our results provide a *roadmap* for deriving mathematical guarantees for deep learning methods in scientific computing by simplifying the proofs, as the needed work essentially reduces to verifying a small number of assumptions. We demonstrate how the generic error bounds from Section 3 can be applied in practice in Section 4, among others by giving short alternative proofs for known results and also proving a number of novel results. In particular, we show in Section 4.1 that PINNs can overcome the curse of dimensionality for nonlinear parabolic PDEs such as the Allen-Cahn equation i.e., the network size does not grow exponentially with increasing spatial dimension. Moreover, dimension-independent convergence rates are also obtained for (physics-informed) DeepONets and FNOs, provided that the PDE solutions are sufficiently smooth. These are the first results of their kind. We note that many of the proofs and some examples are deferred to the supplementary material (**SM**).

2 Preliminaries

2.1 Setting

Given $T > 0$ and $D \subset \mathbb{R}^d$, consider the function $u : [0, T] \times D \to \mathbb{R}^m$, for $m \geq 1$, that solves the following (time-dependent) PDE,

$$\mathcal{L}_a(u)(t, x) = 0 \quad \text{and} \quad u(x, 0) = u_0 \qquad \forall (t, x) \in [0, T] \times D, \tag{2.1}$$

where $u_0 \in \mathcal{Y} \subset L^2(D)$ is the initial condition and $\mathcal{L}_a$ is a differential operator that can depend on a parameter (function) $a \in \mathcal{Z} \subset L^2(D)$. In our notation, we will often suppress the parameter dependence of $\mathcal{L} := \mathcal{L}_a$ on a for simplicity. Depending on the context, one might want to recover one of the following mathematical objects: for fixed a and u_0, one might want to approximate $u(T)$ or $u(\cdot, t)$, for all $t \in [0, T]$ with a neural network; a more challenging task would be to learn the operator $\mathcal{G} : \mathcal{X} \to L^2(\Omega)$, where $\mathcal{X} = \mathcal{Y}$ or $\mathcal{X} = \mathcal{Z}$ and $\Omega = D$ or $\Omega = [0, T] \times D$. We will use this notation consistently throughout the paper, see **SM** A.1 for an overview.

2.2 Approximating PDEs with neural networks

Neural networks A (feedforward) neural network $\Psi_\theta : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ is defined as a concatenation of affine maps $\mathcal{A}_l : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l} : z \mapsto W_l z + b_l$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ that is applied component-wise resulting in,

$$u_\theta(y) = \mathcal{A}_L \circ \sigma \circ \mathcal{A}_{L-1} \ldots\ldots\ldots \circ \sigma \circ \mathcal{A}_2 \circ \sigma \circ \mathcal{A}_1(y). \tag{2.2}$$

The weights and biases of the affine maps $\theta = \{W_l, b_l\}_{1 \le l \le L}$ are the trainable parameters. We will quantify the size of a neural network by its $\mathrm{depth}(u_\theta) := L$ and its $\mathrm{width}(u_\theta) = \max_l d_l$. In order to obtain a neural network that approximates the solution u of PDE (2.1) at time $t = T$, one chooses the parameters of u_θ such that a discretization (quadrature) of $\mathcal{J}(\theta) = \|u(T) - u_\theta\|_{L^2(D)}$ is minimized. The training data is acquired from either measurements or potentially expensive simulations.

PINNs Physics-informed neural networks are neural networks that are trained with a different, residual-based loss function. As the PDE solution u satisfies $\mathcal{L}(u) = 0$, the goal of physics-informed learning is to find a neural network u_θ for which the PDE residual is approximately zero, $\mathcal{L}(u_\theta) \approx 0$. To ensure uniqueness, one also needs to require that the initial condition is satisfied i.e., $u_\theta(0, x) \approx u_0(x)$, and similarly for boundary conditions. In practice one minimizes a quadrature approximation of $\mathcal{J}(\theta) = \|\mathcal{L}(u_\theta)\|^2_{L^2([0,T]\times D)} + \|u_\theta(0, \cdot) - u_0\|^2_{L^2(D)}$, where additional terms can be added to (approximately) impose boundary conditions and augmented by data. A desirable property of PINNs is that only very little or even no training data is needed to construct the loss function.

Operator learning In order to approximate operators, one needs to allow the input and output of the learning architecture to be infinite-dimensional. A possible approach is to use *deep operator networks* (DeepONets), as proposed in [46]. Given m, fixed sensor locations $\{x_j\}_{j=1}^m \subset D$ and the corresponding *sensor values* $\{v(x_j)\}_{j=1}^m$ as input, a DeepONet can be formulated in terms of two (deep) neural networks: a *branch net* $\boldsymbol{\beta} : \mathbb{R}^m \to \mathbb{R}^p$ and a *trunk net* $\boldsymbol{\tau} : D \to \mathbb{R}^{p+1}$. The branch and trunk nets are then combined to approximate the underlying nonlinear operator as the following *DeepONet* $\mathcal{G}_\theta : \mathcal{X} \to L^2(D)$, with $\mathcal{G}_\theta(v)(y) = \tau_0(y) + \sum_{k=1}^p \beta_k(v)\tau_k(y)$. A second approach is that of *neural operators*, which generalize hidden layers by including a non-local integral operator [42], of which particularly *Fourier neural operators* (FNOs) [41] are already well-established. The practical implementation (i.e. discretization) of an FNO maps from and to the space of trigonometric polynomials of degree at most $N \in \mathbb{N}$, denoted by L^2_N, and can be identified with a finite-dimensional mapping that is a composition of affine maps and nonlinear layers of the form $\mathfrak{L}_l(z)_j = \sigma(W_l v_j + b_{l,j}\mathcal{F}_N^{-1}(P_l(k) \cdot \mathcal{F}_N(z)(k)_j))$, where the $P_l(k)$ are coefficients that define a non-local convolution operator via the discrete Fourier transform $\mathcal{F}_N$, see [35].

Physics-informed operator learning Both DeepONets and FNOs are trained by choosing a suitable probability measure μ on $\mathcal{X}$ and minimizing a quadrature approximation of $\mathcal{J}(\theta) = \|\mathcal{G}_\theta(v) - \mathcal{G}(v)\|_{L^2_{\mu\times dx}(\mathcal{X}\times\Omega)}$. Generating training sets might require many calls to an expensive PDE solver, leading to an enormous computational cost. In order to reduce or even fully eliminate the need for training data, *physics-informed operator learning* has been proposed in [70] for DeepONets and in [44] for FNOs. Similar to PINNs, the training procedure aims to minimize a quadrature approximation of $\mathcal{J}(\theta) = \|\mathcal{L}(\mathcal{G}_\theta)\|_{L^2_{\mu\times dx}(\mathcal{X}\times\Omega)}$.

3 General results

We propose a framework to obtain bounds on the approximation error for the various neural network architectures introduced in Section 2.2. Figure 1 visualizes how different types of error estimates can be obtained from one another. Every box shows the name of the network architecture, the form of the relevant loss and the theorem which proves the corresponding estimate for the approximation error. Every arrow in the flowchart represents a proof technique that allows one to transfer an error estimate from one type of method to another (see caption of Figure 1 for an overview of those techniques).

We give particular attention to the case where it is known that a neural network can efficiently approximate the solution to a time-dependent PDE at a fixed time. Such neural networks are usually obtained by emulating a classical numerical method. Examples include finite difference schemes, finite volume schemes, finite element methods, iterative methods and Monte Carlo methods, e.g.

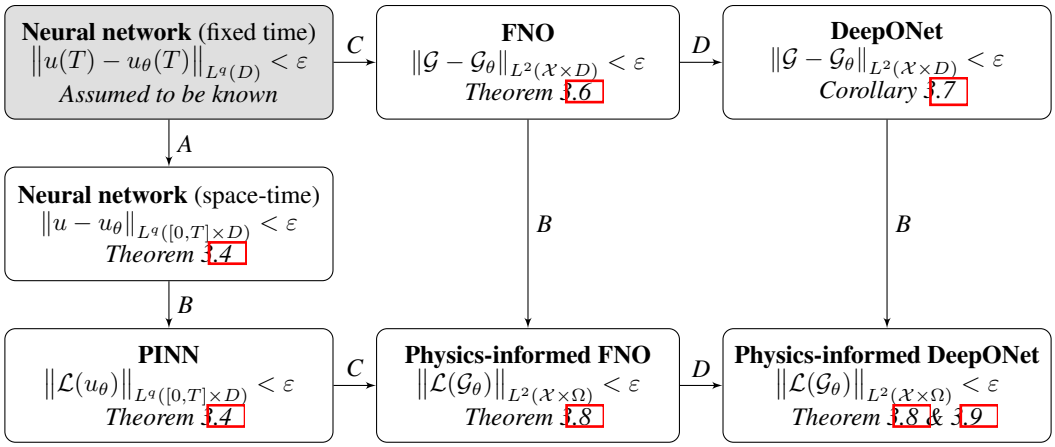

Figure 1: Flowchart of the structure of the results in this paper, with $q \in \{2, \infty\}$. The letters reflect the techniques used in the proofs: A uses Taylor approximations (Section 3.1), B is based on finite difference approximations (Section 3.1), C uses trigonometric polynomial interpolation (Section 3.2) and D uses the connection between FNOs and DeepONets (Section 3.2).

[33, 59, 9]. For $\varepsilon > 0$, we denote by $\mathcal{U}^\varepsilon$ an operator that for any $t \in [0, T]$ maps any initial condition/parameter function $v \in \mathcal{X}$ to a neural network $\mathcal{U}^\varepsilon(v, t)$ that approximates the PDE solution $\mathcal{G}(v)(\cdot, t) = u(\cdot, t)$ at time t, as specified below. Moreover, we will assume that we know how its size depends on the accuracy ε.

Assumption 3.1. *Let $q \in \{2, \infty\}$. For any $B, \varepsilon > 0$, $\ell \in \mathbb{N}$, $t \in [0, T]$ and any $v \in \mathcal{X}$ with $\|v\|_{C^\ell} \leq B$ there exist a neural network $\mathcal{U}^\varepsilon(v, t) : D \to \mathbb{R}$ and a constant $C^B_{\varepsilon,\ell} > 0$ s.t.*

$$\left\|\mathcal{U}^\varepsilon(v, t) - \mathcal{G}(v)(\cdot, t)\right\|_{L^q(D)} \leq \varepsilon \quad \textit{and} \quad \max_{t\in[0,T]} \left\|\mathcal{U}^\varepsilon(v, t)\right\|_{W^{\ell,q}(D)} \leq C^B_{\varepsilon,\ell}. \tag{3.1}$$

Under this assumption, we prove the existence of space-time neural networks and PINNs that efficiently approximate the PDE solution (Section 3.1), as well as FNOs and DeepONets (Section 3.2) and physics-informed FNOs and DeepONets (Section 3.3).

3.1 Estimates for (physics-informed) neural networks

We will construct a space-time neural network u_θ for which both $\|u_\theta - u\|_{L^q([0,T]\times D)}$ and the PINN loss $\|\mathcal{L}(u_\theta)\|_{L^q([0,T]\times D)}$ are small. To accurately approximate the time derivatives of u we emulate Taylor expansions, whereas for the spatial derivatives, we employ finite difference (FD) operators in our proofs. Depending on whether forward, backward or central differences are used, a FD operator might not be defined on the whole domain D, e.g. for $f \in C([0, 1])$ the (forward) operator $\Delta_h^+[f] := f(x + h) - f(x)$ is not well-defined for $x \in (1 - h, 1]$. This can be solved by resorting to piecewise-defined FD operators, e.g. a forward operator on $[0, 0.5]$ and a backward operator on $(0.5, 1]$. In a general domain Ω one can find a well-defined piecewise FD operator if Ω satisfies the following assumption, which is satisfied by many domains (e.g. rectangular, smooth).

Assumption 3.2. *There exists a finite partition $\mathcal{P}$ of Ω such that for all $P \in \mathcal{P}$ there exists $\varepsilon_P > 0$ and $v_P \in B^1_\infty = \{x \in \mathbb{R}^{\dim(\Omega)} : \|x\|_\infty \leq 1\}$ such that for all $x \in P$ it holds that $x + \varepsilon_P(v_P + B^1_\infty) \subset \Omega$.*

Additionally, we need to assume that the PINN error can be bounded in terms of the errors related to all relevant partial derivatives, denoted by $D^{(k,\boldsymbol{\alpha})} := D_t^k D_x^{\boldsymbol{\alpha}} := \partial_t^k \partial_{x_1}^{\alpha_1} \dots \partial_{x_d}^{\alpha_d}$, for $(k, \boldsymbol{\alpha}) \in \mathbb{N}_0^{d+1}$.

Assumption 3.3. *Let $k, \ell \in \mathbb{N}$, $C > 0$ be independent from d. It holds for all $v \in \mathcal{X}$ that,*

$$\left\|\mathcal{L}(\mathcal{G}_\theta(v))\right\|_{L^q([0,T]\times D)} \leq C \cdot \mathrm{poly}(d) \cdot \sum_{\substack{(k',\boldsymbol{\alpha})\in\mathbb{N}_0^{d+1} \\ k'\leq k, \|\boldsymbol{\alpha}\|_1 \leq \ell}} \left\|D^{(k',\boldsymbol{\alpha})}(\mathcal{G} - \mathcal{G}_\theta)\right\|_{L^q([0,T]\times D)}. \tag{3.2}$$

In this setting, we prove the following approximation result for space-time networks and PINNs.

Theorem 3.4. *Let $r, s \in \mathbb{N}$, let $u \in C^{(s,r)}([0,T] \times D)$ be the solution of the PDE (2.1) and let Assumption 3.1 be satisfied. There exists a constant $C(s,r) > 0$ such that for every $M \in \mathbb{N}$ and $\varepsilon, h > 0$ there exists a tanh neural network $u_\theta : [0,T] \times D \to \mathbb{R}$ for which it holds that,*

$$\|u_\theta - u\|_{L^q([0,T]\times D)} \leq C(\|u\|_{C^0} M^{-s} + \varepsilon). \tag{3.3}$$

and if additionally Assumption 3.2 and Assumption 3.3 hold then,

$$\begin{aligned} &\left\|\mathcal{L}(u_\theta)\right\|_{L^q([0,T]\times D)} + \|u_\theta - u\|_{L^2(\partial([0,T]\times D))} \\ &\quad \leq C \cdot \mathrm{poly}(d) \cdot \ln^k(M)(\|u\|_{C^{(s,\ell)}} M^{k-s} + M^{2k}(\varepsilon h^{-\ell} + C^B_{\varepsilon,\ell} h^{r-\ell})). \end{aligned} \tag{3.4}$$

Moreover, $\mathrm{depth}(u_\theta) \leq C \cdot \mathrm{depth}(\mathcal{U}^\varepsilon)$ *and* $\mathrm{width}(u_\theta) \leq CM \cdot \mathrm{width}(\mathcal{U}^\varepsilon)$.

Proof. We only provide a sketch of the full proof (**SM** B.2). The main idea is to divide $[0,T]$ into M uniform subintervals and construct a neural network that approximates a Taylor approximation in time of u in each subinterval. In the obtained formula, we approximate the monomials and multiplications by neural networks (**SM** A.7) and approximate the derivatives of u by finite differences and use (A.2) of **SM** A.2 to find an error estimate in $C^k([0,T], L^q(D))$-norm. We use again finite difference operators to prove that spatial derivatives of u are accurately approximated as well. The neural network will also approximately satisfy the initial/boundary conditions as $\|u_\theta - u\|_{L^2(\partial([0,T]\times D))} \lesssim C\mathrm{poly}(d)\|u_\theta - u\|_{H^1([0,T]\times D)}$, which follows from a Sobolev trace inequality. □

We note that the bounds (3.3) and (3.4) together imply that there exists a neural network for which the total error as well as the PINN loss can be made as small as possible, providing a solid theoretical foundation to PINNs for approximating the PDE (2.1).

3.2 Estimates for operator learning

In this section, we use Assumption 3.1 to prove estimates for DeepONets and FNOs. First, we prove a generic error estimate for FNOs. Using the known connection between FNOs and DeepONets (**SM** Lemma B.6) this result can then easily be applied to DeepONets (Corollary 3.7). In order to prove these error estimates, we need to assume that the operator $\mathcal{U}^\varepsilon$ from Assumption 3.1 is stable with respect to its input function, as specified in Assumption 3.5 below. Moreover, we will take the d-dimensional torus as domain $D = \mathbb{T}^d = [0, 2\pi)^d$ and assume periodic boundary conditions for simplicity in what follows. This is not a restriction, as for every Lipschitz subset of $\mathbb{T}^d$ there exists a (linear and continuous) $\mathbb{T}^d$-periodic extension operator of which also the derivatives are $\mathbb{T}^d$-periodic [35, Lemma 41].

Assumption 3.5. *Assumption 3.1 is satisfied, let $p \in \{2, \infty\}$ and let $p^* \in \{2, \infty\} \setminus \{p\}$. For every $\varepsilon > 0$ there exists a constant $C^\varepsilon_{\mathrm{stab}} > 0$ such that for all $v, v' \in \mathcal{X}$ it holds that,*

$$\left\|\mathcal{U}^\varepsilon(v,T) - \mathcal{U}^\varepsilon(v',T)\right\|_{L^2} \leq C^\varepsilon_{\mathrm{stab}} \left\|v - v'\right\|_{L^p}. \tag{3.5}$$

In this setting, we prove a generic approximation result for FNOs.

Theorem 3.6. *Let $r \in \mathbb{N}$, $T > 0$, let $\mathcal{G} : C^r(\mathbb{T}^d) \to C^r(\mathbb{T}^d)$ be an operator that maps a function u_0 to the solution $u(\cdot, T)$ of the PDE (2.1) with initial condition u_0, and let Assumption 3.5 be satisfied. Then there exists a constant $C > 0$ such that for every $\varepsilon > 0$, $N \in \mathbb{N}$ there is an FNO $\mathcal{G}_\theta : L^2_N(\mathbb{T}^d) \to L^2_N(\mathbb{T}^d)$ of depth $\mathcal{O}(\mathrm{depth}(\mathcal{U}^\varepsilon))$ and width $\mathcal{O}(N^d \mathrm{width}(\mathcal{U}^\varepsilon))$ with accuracy,*

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{L^2} \leq C(\varepsilon + C^\varepsilon_{\mathrm{stab}} B N^{-r+d/p^*} + C^{CB}_{\varepsilon,r} N^{-r}). \tag{3.6}$$

Proof. We give a sketch of the proof, details can be found in **SM** B.3. Given function values of v on a uniform grid with grid size $1/N$, we use trigonometric polynomial interpolation (**SM** A.6) to reconstruct v and use this together with Assumption 3.1 to construct a neural network. The resulting approximation is then projected onto L^2_N through trigonometric polynomial interpolation. □

A recent result, [35, Theorem 36] (**SM** Lemma B.6), shows that any error bound for FNOs also implies an error bound for DeepONets, by choosing the trunk nets as neural network approximations of the Fourier basis. We apply this result with $\varepsilon \sim \mathrm{poly}(1/N)$ to Theorem 3.6 to obtain the following generic error bound for DeepONets.

Corollary 3.7. *Assume the setting of Theorem 3.6. Then for every $\varepsilon > 0$, $N \in \mathbb{N}$ and every corresponding FNO $\mathcal{G}_\theta$ from Theorem 3.6 there exists a DeepONet $\mathcal{G}_\theta^* : \mathcal{X} \to L^2(D)$ with* $\mathrm{width}(\boldsymbol{\beta}) = \mathcal{O}(N^d)$, $\mathrm{depth}(\boldsymbol{\beta}) = \mathcal{O}(\mathrm{depth}(\mathcal{G}_\theta))$, $\mathrm{width}(\boldsymbol{\tau}) = \mathcal{O}(N^{d+1})$ *and* $\mathrm{depth}(\boldsymbol{\tau}) \leq 3$ *that satisfies* (3.6).

3.3 Estimates for physics-informed operator learning

Using the techniques from previous sections, we now present the very first theoretical result for physics-informed operator learning. We demonstrate that if an error estimate for a DeepONet/FNO and the growth of its derivatives are known (see **SM** D.1 on how to obtain these), then one can prove an error estimate for the corresponding physics-informed DeepONet/FNO. For simplicity, the following result focuses only on operators mapping to $C^r(D)$ but the generalization to e.g. $C^r([0,T] \times D)$ is immediate by setting $D' := [0,T] \times D$.

Theorem 3.8. *Consider an operator $\mathcal{G} : \mathcal{X} \to C^r(D)$, $r \in \mathbb{N}$, that satisfies Assumption 3.2 and Assumption 3.5 with $\ell \in N$. Let $\lambda^* \in (0, \infty]$, let $\lambda, C(\lambda) > 0$ with $\lambda \leq \lambda^*$ and let $\sigma : \mathbb{N} \to \mathbb{R}$ be a function such that for all $p \in \mathbb{N}$ there is a DeepONet/FNO $\mathcal{G}_\theta$ such that*

$$\left\|\mathcal{G}(v) - \mathcal{G}_\theta(v)\right\|_{L^2(D)} \leq Cp^{-\lambda} \qquad \text{and} \qquad \left|\mathcal{G}_\theta(v)\right|_{C^r(D)} \leq Cp^{\sigma(r)} \quad \forall r \in \mathbb{N}, v \in \mathcal{X}. \tag{3.7}$$

Then for all $\beta \in \mathbb{R}$ with $0 < \beta \leq \frac{(r-\ell)\lambda^* - \ell\sigma(r)}{r}$ there exists a constant $C^* > 0$ such that for all $v \in \mathcal{X}$ and $p \in \mathbb{N}$ it holds that

$$\left\|\mathcal{L}(\mathcal{G}_\theta(v))\right\|_{L^2(D)} \leq C^* p^{-\beta}. \tag{3.8}$$

Proof. For suitable $D^{\boldsymbol{\alpha}}$, use **SM** Lemma B.1 with $q = 2$, $f_1 = \mathcal{G}(v)$ and $f_2 = \mathcal{G}_\theta(v)$ together with (3.7) to find

$$\left\|D^{\boldsymbol{\alpha}}(\mathcal{G}(v) - \mathcal{G}_\theta(v))\right\|_{L^2(D)} \leq C(r,\lambda)(p^{-\lambda}h^{-\ell} + p^{\sigma(r)}h^{r-\ell}). \tag{3.9}$$

Let $\beta \in \mathbb{R}$ with $0 < \beta \leq \frac{(r-\ell)\lambda^* - \ell\sigma(r)}{r}$. We carefully balance terms by setting $h = p^{-\frac{\sigma(r)+\beta}{r-\ell}}$ and $\lambda = \frac{\ell}{r-\ell}\sigma(r) + \frac{r}{r-\ell}\beta$ to find (3.8). Conclude using Assumption 3.5. □

Finally, we use Theorem 3.4 to present an alternative error estimate for a physics-informed DeepONet in the case that Assumption 3.1 is satisfied. As this assumption is different from assuming access to an error bound for the corresponding DeepONet, it is interesting to use the techniques from the previous sections rather than directly apply Theorem 3.8. The proof of the following theorem can be found in **SM** B.4.

Theorem 3.9. *Let $r, s \in \mathbb{N}$, $T > 0$, let $\mathcal{G} : C^r(\mathbb{T}^d) \to C^{(s,r)}([0,T] \times \mathbb{T}^d)$ be an operator that maps a function u_0 to the solution u of the PDE (2.1) with initial condition u_0, and let Assumption 3.5 be satisfied. There exists a constant $C > 0$ such that for every $Z, N, M \in \mathbb{N}$, $\varepsilon, \rho > 0$ there is an DeepONet $\mathcal{G}_\theta : C^r(\mathbb{T}^d) \to L^2([0,T] \times \mathbb{T}^d)$ with Z^d sensors with accuracy,*

$$\left\|\mathcal{L}(\mathcal{G}_\theta(v))\right\|_{L^2([0,T]\times\mathbb{T}^d)} \leq CM^{k+\rho}(\|u\|_{C^{(s,\ell)}}M^{-s} + M^{s-1}N^\ell(\varepsilon + C^\varepsilon_{\mathrm{stab}}Z^{-r+d/p^*} + C^{CB}_{\varepsilon,r}N^{-r})). \tag{3.10}$$

Moreover, it holds that, $\mathrm{depth}(\boldsymbol{\beta}) = \mathrm{depth}(\mathcal{U}^\varepsilon)$, $\mathrm{width}(\boldsymbol{\beta}) = \mathcal{O}(M(Z^d + N^d\mathrm{width}(\mathcal{U}^\varepsilon)))$, $\mathrm{depth}(\boldsymbol{\tau}) = 3$ *and* $\mathrm{width}(\boldsymbol{\tau}) = \mathcal{O}(MN^d(N + \ln(NM)))$.

4 Applications

We demonstrate the power and generality of the framework proposed in Section 3 by applying the presented theory to the following case studies. First, we demonstrate how these generic bounds can be used to overcome the curse of dimensionality (CoD) for linear Kolmogorov PDEs and nonlinear parabolic PDEs (Section 4.1). These are the first available results that overcome the CoD for nonlinear parabolic PDEs for PINNs and (physics-informed) operator learning. Next, we apply the results of Section 3.3 to both linear and nonlinear operators and provide bounds on the approximation error for physics-informed operator learning.

4.1 Overcoming the curse of dimensionality

For high-dimensional PDEs, it is not possible to obtain efficient approximation results using standard neural network approximation theory [73, 13] as they will lead to convergence rates that suffer from the CoD, meaning that the neural network size scales exponentially in the input dimension. In literature, one has shown for some PDEs that their solution at a fixed time can be approximated to accuracy $\varepsilon > 0$ with a network that has size $\mathcal{O}(\text{poly}(d)\varepsilon^{-\beta})$ and therefore *overcomes the CoD*.

Linear Kolmogorov PDEs We consider linear time-dependent PDEs of the following form.

Setting 4.1. *Let $s, r \in \mathbb{N}$, $u_0 \in C_0^2(\mathbb{R}^d)$ and let $u \in C^{(s,r)}([0,T] \times \mathbb{R}^d)$ be the solution of*

$$\partial_t u(x,t) = \frac{1}{2}\textit{Trace}(\sigma(x)\sigma(x)^T \Delta_x[u](x,t)) + \mu(x)^T \cdot \nabla_x[u](x,t), \qquad u(0,x) = u_0(x) \quad (4.1)$$

for all $(x,t) \in D \times [0,T]$, where $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ and $\mu : \mathbb{R}^d \to \mathbb{R}^d$ are affine functions and for which $\|u\|_{C^{(s,2)}}$ grows at most polynomially in d. For every $\varepsilon > 0$, there is a neural network $\widehat{u}_0$ of depth $\mathcal{O}(\text{poly}(d)\log(1/\varepsilon))$ and width $\mathcal{O}(\text{poly}(d)\varepsilon^{-\beta})$ such that $\|u_0 - \widehat{u}_0\|_{L^\infty(\mathbb{R}^d)} < \varepsilon$.

Prototypical examples of such *linear Kolmogorov PDEs* include the heat equation and the Black-Scholes equation. In [21, 6, 33] the authors construct a neural network that approximates $u(T)$ and overcomes the CoD by emulating Monte-Carlo methods based on the Feynman-Kac formula. In [14] one has proven that PINNs overcome the CoD as well, in the sense that the network size grows as $\mathcal{O}(\text{poly}(d\rho_d)\varepsilon^{-\beta})$ with ρ_d as defined in **SM** (C.10)). For a subclass of linear Kolmogorov PDEs it is known that $\rho_d = \text{poly}(d)$, such that the CoD can be fully overcome. We provide a shorter, alternative proof (in **SM** C.2) for this result based on Theorem 3.4. **SM** Lemma C.6 verifies that Assumption 3.1 is satisfied.

Theorem 4.2. *Assume that Setting 4.1 holds. For every $\varepsilon > 0$ and $d \in \mathbb{N}$, there is a neural network u_θ of depth $\mathcal{O}(\text{depth}(\widehat{u}_0))$ and width $\mathcal{O}(\text{poly}(d\rho_d)\varepsilon^{-(2+\beta)\frac{r+\sigma}{r-2}\frac{s+1}{s-1}-\frac{1+\sigma}{s-1}})$ such that,*

$$\left\|\mathcal{L}(u_\theta)\right\|_{L^2([0,T]\times[0,1]^d)} + \|u_\theta - u\|_{L^2(\partial([0,T]\times[0,1]^d))} \leq \varepsilon. \quad (4.2)$$

Nonlinear parabolic PDEs Next, we consider nonlinear parabolic PDEs as in Section 4.3, which typically arise in the context of nonlinear diffusion-reaction equations that describe the change in space and time of some quantities, such as in the well-known *Allen-Cahn equation* [1].

Setting 4.3. *Let $s, r \in \mathbb{N}$ and for $u_0 \in \mathcal{X} \subset C^r(\mathbb{T}^d)$ let $u \in C^{(s,r)}([0,T] \times \mathbb{T}^d)$ be the solution of*

$$(\partial_t u)(t,x) = (\Delta_x u)(t,x) + F(u(t,x)), \qquad u(0,x) = u_0(x), \qquad (t,x) \in [0,T] \times D, \quad (4.3)$$

with period boundary conditions, where $F : \mathbb{R} \to \mathbb{R}$ is a polynomial and for which $\|u\|_{C^{(s,2)}}$ grows at most polynomially in d. For every $\varepsilon > 0$, there is a neural network $\widehat{u}_0$ of depth $\mathcal{O}(\text{poly}(d)\ln(1/\varepsilon))$ and width $\mathcal{O}(\text{poly}(d)\varepsilon^{-\beta})$ such that $\|u_0 - \widehat{u}_0\|_{L^\infty(\mathbb{T}^d)} < \varepsilon$. Let μ, resp. μ^*, be the normalized Lebesgue measure on $[0,T] \times \mathbb{T}^d$, resp. $\partial([0,T] \times \mathbb{T}^d)$.

In [29] the authors have proven that ReLU neural networks overcome the CoD in the approximation of $u(T)$. We have reproven this result in **SM** Lemma C.15 for tanh neural networks to show that Assumption 3.1 is satisfied. Using Theorem 3.4 we can now prove that PINNs overcome the CoD for nonlinear parabolic PDEs. The proof is analogous to that of Theorem 4.2.

Theorem 4.4. *Assume Setting 4.3. For every $\sigma, \varepsilon > 0$ and $d \in \mathbb{N}$ there is a neural network u_θ of depth $\mathcal{O}(\ln(1/\varepsilon))$ and width $\mathcal{O}(\text{poly}(d)\varepsilon^{-(2+\beta)\frac{r+\sigma}{r-2}\frac{s+1}{s-1}-\frac{1+\sigma}{s-1}})$ such that,*

$$\left\|\partial_t u_\theta - \Delta_x u_\theta - F(u_\theta)\right\|_{L^2([0,T]\times\mathbb{T}^d,\mu)} + \|u - u_\theta\|_{L^2(\partial([0,T]\times\mathbb{T}^d,\mu^*))} \leq \varepsilon. \quad (4.4)$$

Similarly, one can use the results from Section 3.2 to obtain estimates for (physics-informed) DeepONets for nonlinear parabolic PDEs (4.3) such as the Allen-Cahn equation. In particular, a dimension-independent convergence rate can be obtained if the solution is smooth enough, which improves upon the result of [40], which incurred the CoD. For simplicity, we present results for $C^{(2,r)}$ functions, rather than $C^{(s,r)}$ functions, as we found that assuming more regularity did not necessarily further improve the convergence rate. The proof is given in **SM** B.4.

Theorem 4.5. *Assume Setting 4.3 and let $\mathcal{G} : \mathcal{X} \to C^r(\mathbb{T}^d) : u_0 \mapsto u(T)$ and $\mathcal{G}^* : \mathcal{X} \to C^{(2,r)}([0,T] \times \mathbb{T}^d) : u_0 \mapsto u$. For every $\sigma, \varepsilon > 0$, there exists a DeepONets $\mathcal{G}_\theta$ and $\mathcal{G}^*_\theta$ such that*

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{L^2(\mathbb{T}^d \times \mathcal{X})} \leq \varepsilon, \qquad \left\|\mathcal{L}(\mathcal{G}^*_\theta)\right\|_{L^2([0,T]\times\mathbb{T}^d\times\mathcal{X})} \leq \varepsilon. \tag{4.5}$$

Moreover, for $\mathcal{G}_\theta$ we have $\mathcal{O}(\varepsilon^{-\frac{d+\sigma}{r}})$ sensors and,

$$\begin{aligned} \operatorname{width}(\boldsymbol{\beta}) &= \mathcal{O}(\varepsilon^{-\frac{(d+\sigma)(2+\beta)}{r}}), \quad & \operatorname{depth}(\boldsymbol{\beta}) &= \mathcal{O}(\ln(1/\varepsilon)), \\ \operatorname{width}(\boldsymbol{\tau}) &= \mathcal{O}(\varepsilon^{-\frac{d+1+\sigma}{r}}), & \operatorname{depth}(\boldsymbol{\tau}) &= 3, \end{aligned} \tag{4.6}$$

whereas for $\mathcal{G}^*_\theta$ we have $\mathcal{O}(\varepsilon^{-\frac{(3+\sigma)d}{r-2}})$ sensors and,

$$\begin{aligned} \operatorname{width}(\boldsymbol{\beta}) &= \mathcal{O}(\varepsilon^{-1-\frac{(3+\sigma)(d+r(2+\beta))}{r-2}}), \quad & \operatorname{depth}(\boldsymbol{\beta}) &= \mathcal{O}(\ln(1/\varepsilon)), \\ \operatorname{width}(\boldsymbol{\tau}) &= \mathcal{O}(\varepsilon^{-1-\frac{(3+\sigma)(d+1)}{r-2}}), & \operatorname{depth}(\boldsymbol{\tau}) &= 3. \end{aligned} \tag{4.7}$$

4.2 Error bounds for physics-informed operator learning

We demonstrate how Theorem 3.8 can be used to generalize available error estimates for DeepONets and FNOs, e.g. [40, 35] and **SM** D.1, to estimates for their physics-informed counterparts.

Linear operators In the simplest case, the operator $\mathcal{G}$ of interest is linear. In [40, Theorem D.2], a general error bound for ReLU DeepONets for linear operators has been established, which still holds for tanh DeepONets. Using Theorem 3.8 it is then straightforward to prove convergence rates for physics-informed DeepONets for solution operators of linear PDEs (2.1).

Consider an operator $\mathcal{G} : \mathcal{X} \to L^2(\mathbb{T}^d) : v \mapsto u$ as in Section 2.1, where v is the parameter/initial condition and u the solution of the PDE (2.1). Following [40], we fix the measure μ on $L^2(\mathbb{T}^d)$ as a Gaussian random field, such that v allows the Karhunen-Loève expansion $v = \sum_{k\in\mathbb{Z}^d} \alpha_k X_k \mathbf{e}_k$, where $|\alpha_k| \leq \exp(-\ell|k|)$ with $\ell > 0$, the $X_k \sim \mathcal{N}(0,1)$ are iid Gaussian random variables and $\{\mathbf{e}_k\}_{k\in\mathbb{Z}^d}$ is the standard Fourier basis (**SM** A.5). In this setting, we can prove the following approximation result, the proof of which can be found in **SM** D.3.

Theorem 4.6. *Assume the setting above and that of Assumption 3.3, and assume that $\mathcal{G}(v) \in C^{\ell+1}(\mathbb{T}^d)$ for all $v \in \mathcal{X}$. For all $\beta > 0$ there exists a constant $C > 0$ such that for any $p \in \mathbb{N}$ there exists a DeepONet $\mathcal{G}_\theta$ with p sensors and branch and trunk nets such that*

$$\left\|\mathcal{L}(\mathcal{G}_\theta))\right\|_{L^2(L^2(\mathbb{T}^d),\mu)} \leq Cp^{-\beta}. \tag{4.8}$$

Moreover, $\operatorname{size}(\boldsymbol{\tau}) \leq Cp^{\frac{d+1}{d}}$, $\operatorname{depth}(\boldsymbol{\tau}) = 3$, $\operatorname{size}(\boldsymbol{\beta}) \leq p$ *and* $\operatorname{depth}(\boldsymbol{\beta}) = 1$.

Nonlinear operators For nonlinear PDEs a general result like Theorem 4.6 can not be obtained from the currently available tools. Instead one needs to use Theorem 3.8 for every PDE of interest. In the **SM**, we demonstrate this for a nonlinear ODE (gravity pendulum with external force, **SM** D.5) and an elliptic PDE (Darcy flow, **SM** D.6).

5 Related work and discussion

This is the first paper to rigorously expose the connections between the different deep learning frameworks from Section 2.2 for generic PDEs. Until now, most available results focus on providing generic results for one specific method. In [28] and [22] one uses neural networks that approximate solutions to a generic ODE/PDE at a fixed time to construct space-time neural networks. A generalization to PINNs is not immediate as the proof involves the emulation of the forward Euler method. We have overcome this difficulty by constructing space-time neural networks using Taylor expansions instead (Theorem 3.4). To bound the approximation error of PINNs one can use the generic error bounds in Sobolev norms of e.g. [23, 24] for very general activation functions or the more concrete bounds [13] for tanh neural networks. In both approaches, the only assumption is that the solution of the PDE has sufficient Sobolev regularity. As a consequence, these results incur the curse of dimensionality and are not applicable to high-dimensional PDEs. The authors of [13] analyze PINNs based on three theoretical questions related to approximation, stability and generalization.

Other theoretical analyses of PINNs include e.g. [67, 68, 27]. For DeepONets, convergence rates for advection-diffusion equations are presented in [15] and a clear workflow for obtaining generic error estimates as well as worked out examples can be found in [40]. Similar results are obtained for FNOs in [35]. A comprehensive comparison of DeepONets and FNOs is the topic of [47]. To the best of the authors' knowledge, no theoretical results for physics-informed operator learning are currently available.

A second goal of the paper is to prove that deep learning-based frameworks can overcome the curse of dimensionality (CoD). PDEs for which the curse of dimensionality has been overcome include linear Kolmogorov PDEs e.g. [21, 33], nonlinear parabolic PDEs [29] and elliptic PDEs [4]. By assuming that the initial data lies in a Barron class, the authors of [49] proved for elliptic PDEs that the Deep Ritz Method [18] can overcome the CoD. Since the Barron class is a Banach algebra [9] it is possible that our results, which mostly only involve multiplications and additions of neural networks, can be extended to Barron functions. For PINNs, it is proven that they can overcome the CoD for linear Kolmogorov PDEs [14]. We give an alternative proof of this result, improve the convergence rate (Theorem 4.2) and additionally prove that PINNs can also overcome the CoD for nonlinear parabolic PDEs (Theorem 4.4). DeepONets and FNOs can overcome the CoD in many cases [40, 35] but we note that this does not yet include nonlinear parabolic PDEs such as the Allen-Cahn equation. In Theorem 4.5 we prove that dimension-independent convergence rates can be obtained if the solution is sufficiently regular.

It is evident that the generic bounds presented here can only be obtained under suitable assumptions. These should always be checked to prevent misleading claims about mathematical guarantees for the considered deep learning methods. We briefly discuss how restrictive these are and whether they can be relaxed. Assuming the existence of a neural network that approximates the solution of PDE at a fixed time (Assumption 3.1) is of course essential, but such a result can usually be obtained by emulating an existing numerical method. Proving a bound on the Sobolev norm of that network is always possible as we only consider smooth networks. Assumption 3.2 holds for many domains, including rectangular and smooth ones. Assumption 3.3 and Assumption 3.5 also hold for a very broad class of PDEs, much like the assumption on the size of the neural network approximation in Setting 4.1 and 4.3 holds for most functions of interest. Therefore, the assumption that the PDE solution is $C^{(s,r)}$-regular seems to be the most restrictive. However, results like Theorem 3.4 could be extended to e.g. Sobolev regular functions by using the Bramble-Hilbert lemma instead of Taylor expansions. Another restriction is that we exclusively focused on neural networks with the tanh activation function. This was only for simplicity of exposition. All results still hold for other sigmoidal activation functions, as well as more general smooth activation functions, which might give rise to slightly different convergence rates. A last restriction is that the obtained rates are not optimal, but this is not the goal of our framework. In particular, for PINNs for low-dimensional PDEs it is beneficial to use e.g. [24, 13].

Optimizing the obtained convergence rates and comparing with optimal ones is one direction for future research. Previously mentioned possibilities include extending to more general activation functions and less regular functions. Another direction is to make the connection between our results and that of [9] where they prove that Barron spaces are Banach algebras and use this to obtain dimension-independent convergence rates for PDEs with initial data in a Barron class by emulating numerical methods. Finally, we only considered the approximation error in the present analysis. It would be interesting to prove generic results for the total error of the trained learning architectures by also bounding the underlying generalization error.

References

[1] S. M. Allen and J. W. Cahn. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta metallurgica*, 27(6):1085–1095, 1979.

[2] G. Bai, U. Koley, S. Mishra, and R. Molinaro. Physics informed neural networks (PINNs) for approximating nonlinear dispersive PDEs. *arXiv preprint arXiv:2104.05584*, 2021.

[3] A. Barth, A. Jentzen, A. Lang, and C. Schwab. *Numerical Analysis of Stochastic Ordinary Differential Equations*. ETH Zürich, 2018.

[4] C. Beck, L. Gonon, and A. Jentzen. Overcoming the curse of dimensionality in the numerical approximation of high-dimensional semilinear elliptic partial differential equations. *arXiv preprint arXiv:2003.00596*, 2020.

[5] C. Beck, F. Hornung, M. Hutzenthaler, A. Jentzen, and T. Kruse. Overcoming the curse of dimensionality in the numerical approximation of Allen-Cahn partial differential equations via truncated full-history recursive multilevel Picard approximations. *Journal of Numerical Mathematics*, 28(4):197–222, 2020.

[6] J. Berner, P. Grohs, and A. Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

[7] S. Cai, Z. Wang, L. Lu, T. A. Zaki, and G. E. Karniadakis. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.

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

[9] Z. Chen, J. Lu, and Y. Lu. On the representation of solutions to elliptic PDEs in Barron spaces. *arXiv preprint arXiv:2106.07539*, 2021.

[10] A. Cohen, R. Devore, and C. Schwab. Analytic regularity and polynomial approximation of parametric and stochastic elliptic PDEs. *Analysis and Applications*, 9(01):11–47, 2011.

[11] G. Constantine and T. Savits. A multivariate Faa di Bruno formula with applications. *Transactions of the American Mathematical Society*, 348(2):503–520, 1996.

[12] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[13] T. De Ryck, S. Lanthaler, and S. Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 2021.

[14] T. De Ryck and S. Mishra. Error analysis for physics informed neural networks (PINNs) approximating Kolmogorov PDEs. *arXiv preprint arXiv:2106.14473*, 2021.

[15] B. Deng, Y. Shin, L. Lu, Z. Zhang, and G. E. Karniadakis. Convergence rate of DeepONets for learning operators arising from advection-diffusion equations. *arXiv preprint arXiv:2102.10621*, 2021.

[16] M. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 1994.

[17] W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[18] W. E and B. Yu. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[19] R. A. Fisher. The wave of advance of advantageous genes. *Annals of eugenics*, 7(4):355–369, 1937.

[20] S. Goswami, M. Yin, Y. Yu, and G. E. Karniadakis. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391:114587, 2022.

[21] P. Grohs, F. Hornung, A. Jentzen, and P. Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.

[22] P. Grohs, F. Hornung, A. Jentzen, and P. Zimmermann. Space-time error estimates for deep neural network approximations for differential equations. *arXiv preprint arXiv:1908.03833*, 2019.

[23] I. Gühring, G. Kutyniok, and P. Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *Analysis and Applications*, 18(05):803–859, 2020.

[24] I. Gühring and M. Raslan. Approximation rates for neural networks with encodable weights in smoothness spaces. *Neural Networks*, 134:107–130, 2021.

[25] P. Henry-Labordere. Counterparty risk valuation: A marked branching diffusion approach. *Available at SSRN 1995503*, 2012.

[26] P. Henry-Labordere, X. Tan, and N. Touzi. A numerical algorithm for a class of BSDEs via the branching process. *Stochastic Processes and their Applications*, 124(2):1112–1140, 2014.

[27] B. Hillebrecht and B. Unger. Certified machine learning: A posteriori error estimation for physics-informed neural networks. *arXiv preprint arXiv:2203.17055*, 2022.

[28] F. Hornung, A. Jentzen, and D. Salimova. Space-time deep neural network approximations for high-dimensional partial differential equations. *arXiv preprint arXiv:2006.02199*, 2020.

[29] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen. A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. *SN partial differential equations and applications*, 1(2):1–34, 2020.

[30] M. Hutzenthaler, A. Jentzen, B. Kuckuck, and J. L. Padgett. Strong L^p-error analysis of nonlinear Monte Carlo approximations for high-dimensional semilinear partial differential equations. *arXiv preprint arXiv:2110.08297*, 2021.

[31] A. D. Jagtap and G. E. Karniadakis. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

[32] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

[33] A. Jentzen, D. Salimova, and T. Welti. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*, 2018.

[34] A. N. Kolmogorov. Étude de l'équation de la diffusion avec croissance de la quantité de matière et son application à un problème biologique. *Bull. Univ. Moskow, Ser. Internat., Sec. A*, 1:1–25, 1937.

[35] N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for Fourier Neural Operators. *arXiv preprint arXiv:2107.07562*, 2021.

[36] N. Kovachki, Z. Li, B. Liu, K. Azizzadensheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481v3*, 2021.

[37] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric PDEs. *Constructive Approximation*, pages 1–53, 2021.

[38] I. E. Lagaris, A. Likas, and P. G. D. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11:1041–1049, 2000.

[39] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 2000.

[40] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for DeepONets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.

[41] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.

[42] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Neural operator: Graph kernel network for partial differential equations. *CoRR*, abs/2003.03485, 2020.

[43] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, A. M. Stuart, K. Bhattacharya, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6755–6766. Curran Associates, Inc., 2020.

[44] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[45] C. Lin, Z. Li, L. Lu, S. Cai, M. Maxey, and G. E. Karniadakis. Operator learning for predicting multiscale bubble growth dynamics. *The Journal of Chemical Physics*, 154(10):104118, 2021.

[46] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[47] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[48] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[49] Y. Lu, J. Lu, and M. Wang. A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic partial differential equations. In *Conference on Learning Theory*, pages 3196–3241. PMLR, 2021.

[50] K. O. Lye, S. Mishra, and D. Ray. Deep learning observables in computational fluid dynamics. *Journal of Computational Physics*, page 109339, 2020.

[51] K. O. Lye, S. Mishra, D. Ray, and P. Chandrashekar. Iterative surrogate model optimization (ISMO): An active learning algorithm for pde constrained optimization with deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 374:113575, 2021.

[52] Z. Mao, A. D. Jagtap, and G. E. Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[53] Z. Mao, L. Lu, O. Marxen, T. A. Zaki, and G. E. Karniadakis. DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. *Journal of Computational Physics*, 447:110698, 2021.

[54] H. P. McKean. Application of brownian motion to the equation of Kolmogorov-Petrovskii-Piskunov. *Communications on pure and applied mathematics*, 28(3):323–331, 1975.

[55] S. Mishra and R. Molinaro. Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs. *arXiv preprint arXiv:2006.16144*, 2020.

[56] S. Mishra and R. Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. *IMA Journal of Numerical Analysis*, 2021.

[57] S. Mishra and R. Molinaro. Physics informed neural networks for simulating radiative transfer. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 270:107705, 2021.

[58] B. Øksendal. *Stochastic differential equations*. Springer, 2003.

[59] J. A. Opschoor, P. C. Petersen, and C. Schwab. Deep ReLU networks and high-order finite element methods. *Analysis and Applications*, 18(05):715–770, 2020.

[60] J. A. Opschoor, C. Schwab, and J. Zech. Exponential ReLU DNN expression of holomorphic maps in high dimension. *Constructive Approximation*, pages 1–46, 2021.

[61] G. Pang, L. Lu, and G. E. Karniadakis. fPINNs: Fractional physics-informed neural networks. *SIAM journal of Scientific computing*, 41:A2603–A2626, 2019.

[62] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, p. Hassanzadeh, K. Kashinath, and A. Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[63] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[64] 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.

[65] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint arXiv:1808.04327*, 2018.

[66] C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Analysis and Applications*, 17(01):19–55, 2019.

[67] Y. Shin, J. Darbon, and G. E. Karniadakis. On the convergence and generalization of physics informed neural networks. *arXiv preprint arXiv:2004.01806*, 2020.

[68] Y. Shin, Z. Zhang, and G. E. Karniadakis. Error estimates of residual minimization using neural networks for linear equations. *arXiv preprint arXiv:2010.08019*, 2020.

[69] S. Wang and P. Perdikaris. Long-time integration of parametric evolution equations with physics-informed DeepONets. *arXiv preprint arXiv:2106.05384*, 2021.

[70] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. *arXiv preprint arXiv:2103.10974*, 2021.

[71] J. Yang, Q. Du, and W. Zhang. Uniform L^p-bound of the Allen-Cahn equation and its numerical discretization. *International Journal of Numerical Analysis & Modeling*, 15, 2018.

[72] L. Yang, X. Meng, and G. E. Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3 and Section 4.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All assumptions are either stated in the theorem statement or described in the text right above the theorem statement.
   (b) Did you include complete proofs of all theoretical results? [Yes] For each result we mention where the proof can be found.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]