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

Tim De Ryck ⋆
Siddhartha Mishra †

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 [19, 70, 40, 53, 54] 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 [53], necessitating the design of learning frameworks that work with limited data. Physics-informed neural networks (PINNs), proposed by [18, 42, 41] and popularized by [67, 68], 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, [69, 51, 65, 70, 54, 55, 60, 59, 60, 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 [27, 39]. 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) [9, 49] and their variants as well as neural operators [39], which generalize neural networks to this setting. A variety of neural operators have been proposed, see [45, 44] but arguably, the most efficient form of neural operators is provided by the so-called Fourier neural operators (FNOs) [44]. Both DeepONets and FNOs have been very successfully deployed in

⋆Seminar for Applied Mathematics (SAM), D-MATH, ETH Zürich, Switzerland
†Seminar for Applied Mathematics (SAM), D-MATH and ETH AI center, ETH Zürich, Switzerland

scientific computing and references therein. Finally, one can combine PINNs and operator learning to design physics-informed DeepONets/FNOs.

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 $\epsilon > 0$, we need to rigorously prove that the approximation error of the neural network (operator) can be made smaller than $\epsilon$. 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 $\epsilon^{-1}$. In particular, exponential growth has to be ruled out. As neural networks, DeepONets and FNOs are all universal approximators 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 for neural networks in very high spatial dimensions, for DeepONets and 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. and references therein. In contrast, there is a relative scarcity of such efficient approximation results for PINNs and operator learning with notable exceptions being (for PINNs), (for DeepONets) and (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 2) 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., that 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 $D \subset \mathbb{R}^d$ compact, consider the function $u : [0, T] \times D \to \mathbb{R}^m$, for $m \geq 1$, that belongs to a function space $\mathcal{H}$ and solves the following (time-dependent) PDE,

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

where $u_0 \in \mathcal{Y} \subset L^2(D)$ is the initial condition and $\mathcal{L}_a : \mathcal{H} \to L^2([0, T] \times D)$ 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 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 $u(T, \cdot)$ or $u(\cdot, \cdot)$ with a neural network; a more challenging task would be to learn the solution operator $\mathcal{G} : \mathcal{A} \to L^2(\Omega) : v \mapsto u$, where $v \in \{u_0, a\}$, $\mathcal{A} \in \{\mathcal{Y}, \mathcal{Z}\}$ and $\Omega = D$ or
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. [60, 63][10, 57]. More precisely, for $\varepsilon > 0$, we assume to have access to an operator $U^\varepsilon : \mathcal{X} \times [0, T] \to \mathcal{H}$ that for any $t \in [0, T]$ maps any initial condition/parameter function $v \in \mathcal{X}$ to a neural network $U^\varepsilon(v, t)$ that approximates the PDE solution $G(v)(\cdot, t) = u(\cdot, t) \in L^2(D)$, $q \in \{2, \infty\}$, at time $t$, as specified below. Moreover, we will assume that we know how its size depends on the accuracy $\varepsilon$. Explicit examples of the operator $U^\varepsilon$ will be given in Section 3 and SMC.

**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 $U^\varepsilon(v, t) : D \to \mathbb{R}$ and a constant $C^\varepsilon_{B, \ell} > 0$ s.t.

$$
\|U^\varepsilon(v, t) - G(v)(\cdot, t)\|_{L^2(D)} \leq \varepsilon \quad \text{and} \quad \max_{t \in [0, T]} \|U^\varepsilon(v, t)\|_{W^{\ell, q}(D)} \leq C^\varepsilon_{B, \ell}.
$$

(3.1)

**Remark 3.2.** For vanilla neural networks and PINNs one can set $\mathcal{X} := \{v\}$, $G(v) := u$ and $v := u_0$ or $v := a$ in Assumption 3.1 above and Assumption 3.4 below.

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). Finally, we also prove a general result on the generalization error (Section 3.4).

### 3.1 Estimates for (physics-informed) neural networks

We will construct a space-time neural network $u_0$ for which both $\|u_0 - u\|_{L^2([0, T] \times D)}$ and the PINN loss $\|\mathcal{L}(u_0)\|_{L^2([0, T] \times D)}$ 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 $\Omega$ one can find a well-defined piecewise FD operator if $\Omega$ satisfies the following assumption, which is satisfied by many domains (e.g. rectangular, smooth).

**Assumption 3.3.** There exists a finite partition $P$ of $\Omega$ such that for all $P \in P$ there exists $\varepsilon_P > 0$ and $v_P \in B_\infty = \{x \in \mathbb{R}^{\text{dim}(\mathcal{D})} : \|x\| \leq 1\}$ such that for all $x \in P$ it holds that $x + \varepsilon_P v_P + B_1 \subset \Omega$.

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 $D^{(k, \alpha)} := D^k_1 \partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}$, for $(k, \alpha) \in \mathbb{N}^{d+1}$. 

**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).
This assumption is valid for many classical solutions of PDEs. A few worked out examples can be
found in SM D.3 (gravity pendulum) and SM D.6 (Darcy flow).

**Assumption 3.4.** Let \( k, \ell \in \mathbb{N} \), \( q \in \{2, \infty\} \), \( C > 0 \) be independent from \( d \). For all \( v \in \mathcal{X} \) it holds,

\[
\| \mathcal{L}(\mathcal{G}_\theta(v)) \|_{L^q([0,T] \times D)} \leq C \cdot \text{poly}(d) \cdot \sum_{(k',\alpha) \in \mathcal{S}^{d+1}} \| D^{(k',\alpha)}(G - \mathcal{G}_\theta) \|_{L^q([0,T] \times D)}.
\]

(3.2)

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

**Theorem 3.5.** Let \( s, r \in \mathbb{N} \), let \( u \in C^{(s,r)}([0,T] \times D) \) be the solution of the PDE (2.1) and let Assumption 3.7 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 \( u_\theta : [0,T] \times D \rightarrow \mathbb{R} \) for which it holds that,

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

(3.3)

and if additionally Assumption 3.3 and Assumption 3.4 hold then,

\[
\| \mathcal{L}(u_\theta) \|_{L^2([0,T] \times D)} + \| u_\theta - u \|_{L^2(\partial([0,T] \times D))} \leq C \cdot \text{poly}(d) \cdot \ln(k(M)) \| u \|_{C^{(s,r)}(0,T \times D)} M^{2k} + M^{2k}(\varepsilon h^{-\alpha} + C \varepsilon \ell h^{-\ell})).
\]

(3.4)

Moreover, depth \( (u_\theta) \) \( \leq C \cdot \text{depth}(u^\ell) \) and width \( (u_\theta) \) \( \leq C M \cdot \text{width}(u^\ell) \).

**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) 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 \( \| u_\theta - u \|_{L^2(\partial([0,T] \times D))} \lesssim C \text{poly}(d) \| u_\theta - u \|_{H^1([0,T] \times D)} \) which follows from a Sobolev trace inequality. \( \square \)

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 3.5). In order to prove these error estimates, we need to assume that the operator \( \mathcal{U}^\ell \) from Assumption 3.1 is stable with respect to its input function, as specified in Assumption 3.6 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 (SM Lemma 41).

**Assumption 3.6.** Assumption 3.1 is satisfied and let \( p \in \{2, \infty\} \). For every \( \varepsilon > 0 \) there exists a constant \( C_{\text{stab}} > 0 \) such that for all \( v, v' \in \mathcal{X} \) it holds that,

\[
\| \mathcal{U}^\ell(v,T) - \mathcal{U}^\ell(v',T) \|_{L^2} \leq C_{\text{stab}} \| v - v' \|_{L^p}.
\]

(3.5)

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

**Theorem 3.7.** Let \( r \in \mathbb{N} \), \( T > 0 \), let \( \mathcal{G} : C^r(\mathbb{T}^d) \rightarrow C^r(\mathbb{T}^d) \) be an operator that maps a function \( u_0 \) to the solution \( u(\cdot,T) \) of the PDE (2.1) with initial condition \( u_0 \), let Assumption 3.6 be satisfied and let \( p^* \in \{2, \infty\} \setminus \{p\} \). Then there exists a constant \( C > 0 \) such that for every \( \varepsilon > 0 \), \( N \in \mathbb{N} \), there is an FNO \( \mathcal{G}_\theta : L^p(\mathbb{T}^d) \rightarrow L^p(\mathbb{T}^d) \) of depth \( O(\text{depth}(\mathcal{U}^\ell)) \) and width \( O(N^d \cdot \text{width}(\mathcal{U}^\ell)) \) with accuracy,

\[
\| \mathcal{G} - \mathcal{G}_\theta \|_{L^2} \leq C(\varepsilon + C_{\text{stab}} B N^{-r+d/p^*} + C_{\varepsilon,B} N^{-r}).
\]

(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 B.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 $\mathbb{C}^r(D)$ but the generalization to e.g. $\mathbb{C}^r\left([0, T] \times D\right)$ is immediate by considering $D' := [0, T] \times D$.

**Theorem 3.9.** Consider an operator $G: \mathcal{X} \to \mathbb{C}^r(D)$, $r \in \mathbb{N}$, that satisfies Assumption 3.3 and Assumption 3.4 with $\ell \in \mathbb{N}$. 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 $G_\theta$ such that

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

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

$$\|\mathcal{L}(G_\theta(v))\|_{L^2(D)} \leq C^* p^{-\beta}. \quad (3.8)$$

**Proof.** For suitable $D^a$, use SM Lemma B.1 with $q = 2$, $f_1 = G(v)$ and $f_2 = G_\theta(v)$ together with (3.7) to find

$$\|D^a(G(v) - G_\theta(v))\|_{L^2(D)} \leq C(r, \lambda)(p^{-\lambda}h^{\ell} + p^{\sigma(r)}h^{\ell-\ell}). \quad (3.9)$$

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 3.4.

Finally, we use Theorem 5.5 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 3.9. The proof of the following theorem can be found in SM B.4.

**Theorem 3.10.** Let $s, r \in \mathbb{N}, T > 0$, let $G: \mathbb{C}^r(\mathbb{T}^d) \to \mathbb{C}^{(s, r)}([0, T] \times \mathbb{T}^d)$ be an operator that maps a function $u_0$ to the solution $u$ of the PDE (2.1) with initial condition $u_0$, let Assumption 3.7 and Assumption 3.8 be satisfied and let $p^* \in [2, \infty) \setminus \{p\}$ There exists a constant $C > 0$ such that for every $Z, N, M \in \mathbb{N}, \varepsilon, \rho > 0$ there is an DeepONet $G_\theta : \mathbb{C}^r(\mathbb{T}^d) \to L^2([0, T] \times \mathbb{T}^d)$ with $Z^d$ sensors with accuracy

$$\|G(v) - G_\theta(v)\|_{L^2([0, T] \times \mathbb{T}^d)} \leq CM^p(\|u\|_{C^{(s, r)}([0, T] \times \mathbb{T}^d)} M^{-s} + M^{s-1}(\varepsilon + C_{\text{stab}} Z^{r+d/p^*} + C_{\varepsilon, T}^B N^{-r})) \quad (3.10)$$

and if additionally Assumption 3.3 and Assumption 3.4 hold then,

$$\|\mathcal{L}(G_\theta(v))\|_{L^2([0, T] \times \mathbb{T}^d)} \leq CM^{k+p^*}(\|u\|_{C^{(s, r)}([0, T] \times \mathbb{T}^d)} M^{-s} + M^{s-1}N^{\ell}(\varepsilon + C_{\text{stab}} Z^{r+d/p^*} + C_{\varepsilon, T}^B N^{-r})). \quad (3.11)$$

for all $v$. Moreover, it holds that, $\text{depth}(\beta) = \text{depth}(U^F)$, $\text{width}(\beta) = \mathcal{O}(M(Z^d+N^d\text{width}(U^F)))$, $\text{depth}(\tau) = 3$ and $\text{width}(\tau) = \mathcal{O}(MN^d(N + \ln(N)))$. 

Proof. We give a sketch of the proof, details can be found in SM B.4. 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 B.1 to construct a neural network. The resulting approximation is then projected onto the space $L^2_{\lambda}$, of trigonometric polynomials of degree at most $N \in \mathbb{N}$, again through trigonometric polynomial interpolation.

A recent result, SM 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 $\varepsilon \sim \text{poly}(1/N)$ to Theorem 3.7 to obtain the following generic error bound for DeepONets.

**Corollary 3.8.** Assume the setting of Theorem 3.7. Then for every $\varepsilon > 0$, $N \in \mathbb{N}$ and every corresponding FNO $G_\theta$ from Theorem 3.7 there exists a DeepONet $G^*_\theta : \mathcal{X} \to L^2(D)$ with $\text{width}(\beta) = \mathcal{O}(N^d)$, $\text{depth}(\beta) = \mathcal{O}(\text{depth}(G_\theta))$, $\text{width}(\tau) = \mathcal{O}(N^{d+1})$ and $\text{depth}(\tau) \leq 3$ that satisfies 3.6.
3.4 A posteriori bound on the generalization error

Although the main focus of this paper is on the approximation error for different neural network architectures, we now demonstrate that it is possible to provide similar bounds for other sources of error, such as the generalization error. We therefore prove a general a posteriori upper bound on the generalization error of the all the considered neural network architectures. Consider \( f : \mathcal{D} \to \mathbb{R} \) (an operator or function) and the neural network architecture \( f_\theta : \mathcal{D} \to \mathbb{R}, \theta \in \Theta \), which includes all architectures of Section 3.2. neural networks \((\mathcal{D} = \Omega, f = u \text{ and } f_\theta = u_0)\), PINNs \((\mathcal{D} = \Omega, f = 0 \text{ and } f_\theta = \mathcal{L}(u_0))\), operator learning \((\mathcal{D} = \Omega \times \mathcal{X}, f = \mathcal{G} \text{ and } f_\theta = \mathcal{G}_\theta)\) and physics-informed operator learning \((\mathcal{D} = \Omega \times \mathcal{X}, f = 0 \text{ and } f_\theta = \mathcal{L}(\mathcal{G}_\theta))\). Given a training set \( \mathcal{S} = \{X_1, \ldots, X_n\} \), where \( \{X_i\}_{i=1}^n \) are iid random variables on \( \mathcal{D} \) (according to a measure \( \mu \)), the training error \( \mathcal{E}_T \) and generalization error \( \mathcal{E}_G \) are,

\[
\mathcal{E}_T(\theta, \mathcal{S})^2 = \frac{1}{n} \sum_{i=1}^n |f(z_i) - f_\theta(z_i)|^2, \quad \mathcal{E}_G(\theta)^2 = \int_\mathcal{D} |f_\theta(z) - f(z)|^2 d\mu(z),
\]

where \( \mu \) is a probability measure on \( \mathcal{D} \). The following theorem provides a computable a posteriori error bound on the expectation of the generalization error for a general class of approximators. We refer to e.g. [6, 16] for bounds on \( d_\theta, c \) and \( \mathcal{L} \).

**Theorem 3.11.** For \( R > 0 \) and \( d_\theta \in \mathbb{N} \), let \( \Theta = [-R, R]^{d_\theta} \) be the set of trainable parameters, and for every training set \( \mathcal{S} \), let \( \theta^*(\mathcal{S}) \in \Theta \) be an (approximate) minimizer of \( \theta \mapsto \mathcal{E}_T(\theta, \mathcal{S})^2 \); assume that \( \theta \mapsto \mathcal{E}_G(\theta)^2 \) and \( \theta \mapsto \mathcal{E}_T(\theta)^2 \) are bounded by \( c > 0 \) and Lipschitz continuous with Lipschitz constant \( \mathcal{L} > 0 \). If \( n \geq 2c^2e^8/(2R\mathcal{L})^{d_\theta/2} \) then it holds that

\[
\mathbb{E} \left[ \mathcal{E}_G(\theta^*(\mathcal{S}))^2 \right] \leq \mathbb{E} \left[ \mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S})^2 \right] + \sqrt{\frac{2c^2(d_\theta + 1)}{n} \ln \left( n \mathcal{L} \right)}.
\]

**Proof.** The proof (SM 3.5) combines standard techniques, based on covering numbers and Hoeffding’s inequality, with an error composition from [16].

For any type of neural network architecture of depth \( L \), width \( W \) and weights bounded by \( R \), one finds that \( d_\theta \sim LW(W + d) \). For tanh neural networks and operator learning architectures, one has that \( \ln(\mathcal{L}) \sim L \ln(dRW) \), whereas for physics-informed neural networks and DeepONets one finds that \( \ln(\mathcal{L}) \sim (k + \ell)L \ln(dRW) \) with \( k \) and \( \ell \) as in Assumption 3.4 [14, 16]. Taking this into account, one also finds that the imposed lower bound on \( n \) is not very restrictive. Moreover, the RHS of (3.13) depends at most polynomially on \( L, W, R, d, k, \ell \) and \( c \). For physics-informed architectures, however, upper bounds on \( c \) often depend exponentially on \( L \) [16, 14].

**Remark 3.12.** As Theorem 3.11 is an a posteriori error estimate, one can use the network sizes of the trained networks for \( L, W \) and \( R \). The sizes stemming from the approximation error estimates of the previous sections can be disregarded for this result. Moreover, instead of considering the expected values of \( \mathcal{E}_G \) and \( \mathcal{E}_T \) in (3.13), one can also prove that such an inequality holds with a certain probability (see SM 3.5).

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 [7, 15] 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 \( \mathcal{O}(\text{poly}(d)\varepsilon^{-\beta}) \), with \( \beta > 0 \) independent of \( d \), and therefore overcomes the CoD.

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

**Setting 4.1.** Let \( s, r \in \mathbb{N} \), \( u_0 \in C^2_0(\mathbb{R}^d) \) and let \( u \in C^{(s,r)}([0,T] \times \mathbb{R}^d) \) be the solution of

\[
\mathcal{L}(u)(x,t) = \partial_t u(x,t) - \frac{1}{2} \text{Tr}(\sigma(x)\sigma(x)^T \Delta_x[u](x,t)) - \mu(x)^T \nabla_x[u](x,t) = 0, \quad u(0,x) = u_0(x)
\]

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

Prototypical examples of such linear Kolmogorov PDEs include the heat equation and the Black-Scholes equation. In [23, 24, 50] 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 [16] 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 \( \rho_d \) as defined in SM (C.10). For a subclass of Kolmogorov PDEs it is known that \( \rho_d = \text{poly}(d) \), such that the CoD is fully overcome.

We demonstrate that the generic bounds of Section 3 (Theorem 3.5) can be used to provide a much shorter proof for this result. SM Lemma C.6 verifies that Assumption 3.1 is indeed satisfied. The full proof can be found in SM C.2.

**Theorem 4.2.** Assume that Setting 4.1 holds. For every \( \sigma, \| \sigma \|_{C^{(s,r)}} > 0 \) and \( d \in \mathbb{N} \), there is a tanh neural network \( u_0 \) of depth \( \mathcal{O}(\text{depth}(\tilde{u}_0)) \) and width \( \mathcal{O}(\text{poly}(d\rho_d)\varepsilon^{-((2+\beta)/2 + 1/2 - 1/2)\rho_d}) \) such that

\[
\|\mathcal{L}(u_0)\|_{L^2([0,T]\times[0,1]^d)} + \|u_0 - u\|_{L^2([0,T]\times[0,1]^d)} \leq \varepsilon.
\]

**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 [11].

**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

\[
\mathcal{L}(u)(x,t) = \partial_t u(x,t) - \Delta_x u(x,t) - F(u(t,x)) = 0, \quad u(0,x) = u_0(x),
\]

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

In [32] 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.14 for tanh neural networks to show that Assumption 5.7 is satisfied. Using Theorem 4.2, 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 4.4. For every \( \varepsilon > 0 \) and \( d \in \mathbb{N} \) there is a tanh neural network \( u_0 \) of depth \( \mathcal{O}(\text{depth}(\tilde{u}_0) + \text{poly}(d)\ln(1/\varepsilon)) \) and width \( \mathcal{O}(\text{poly}(d)\varepsilon^{-((2+\beta)/2 + 1/2 - 1/2)\rho_d}) \) such that

\[
\|\mathcal{L}(u_0)\|_{L^2([0,T]\times\mathbb{T}^d,\mu)} + \|u - u_0\|_{L^2([0,T]\times\mathbb{T}^d,\mu^*)} \leq \varepsilon.
\]

Similarly, one can use the results from Section 5.2 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 [43], which incurred the CoD. For simplicity, we present results for \( C^{(s,r)} \) 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.3.

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

\[
\|G - G_{\theta}\|_{L^2(\mathbb{T}^d,\mathcal{X})} \leq \varepsilon, \quad \|\mathcal{L}(G_{\theta}^*)_\|_{L^2([0,T]\times\mathbb{T}^d,\mathcal{X})} \leq \varepsilon.
\]
Moreover, for $G_0$ we have $O(\varepsilon^{-\frac{d+\sigma}{2}})$ sensors and,
\[
\text{width}(\beta) = O(\varepsilon^{-\frac{(d+\sigma)(2+\theta)}{2}}), \quad \text{depth}(\beta) = O(\ln(1/\varepsilon)),
\]
\[
\text{width}(\tau) = O(\varepsilon^{-\frac{d+\sigma}{2}}), \quad \text{depth}(\tau) = 3,
\]  
(4.6)
whereas for $G_0^*$ we have $O(\varepsilon^{-\frac{3(d+\sigma)2}{2}})$ sensors and,
\[
\text{width}(\beta) = O(\varepsilon^{-1-\frac{(3(d+\sigma)+d+2+\beta)}{2}}), \quad \text{depth}(\beta) = O(\ln(1/\varepsilon)),
\]
\[
\text{width}(\tau) = O(\varepsilon^{-1-\frac{3(d+\sigma)(d+1)}{2}}), \quad \text{depth}(\tau) = 3.
\]  
(4.7)

## 4.2 Error bounds for physics-informed operator learning

We demonstrate how Theorem 3.9 can be used to generalize available error estimates for DeepONets and FNOs, e.g. [43, 38] and SM D.1 to estimates for their physics-informed counterparts.

### Linear operators

In the simplest case, the operator $G$ of interest is linear. In [43, Theorem D.2], a general error bound for ReLU DeepONets for linear operators has been established, which still holds for tanh DeepONets. Using Theorem 3.9 it is then straightforward to prove convergence rates for physics-informed DeepONets for solution operators of linear PDEs (2.1).

Consider an operator $G : X \rightarrow L^2(\mathbb{T}^d) : v \mapsto u$ as in Section 2.1, where $v$ is the parameter/initial condition and $u$ the solution of the PDE (2.1). Following [43], we fix the measure $\mu$ on $L^2(\mathbb{T}^d)$ as a Gaussian random field, such that $v$ allows the Karhunen-Loève expansion $v = \sum_{k \in \mathbb{Z}^d} \alpha_k X_k \phi_k$, where $|\alpha_k| \leq \exp(-\ell |k|)$ with $\ell > 0$, the $X_k \sim \mathcal{N}(0,1)$ are iid Gaussian random variables and $\{\phi_k\}_{k \in \mathbb{Z}^d}$ is the standard Fourier basis (SM A.3). In this setting, we can prove the following approximation result, the proof of which can be found in SM D.3. The result can be generalized to other data distributions $\mu$ for which a convergence result for DeepONets can be proven, as in [43].

**Theorem 4.6.** Assume the setting above and that of Assumption 3.4 and assume that $G(v) \in C^{\ell+1}(\mathbb{T}^d)$ for all $v \in X$. For all $\beta > 0$ there exists a constant $C > 0$ such that for any $p \in \mathbb{N}$ there exists a DeepONet $G_\beta$ with $p$ sensors and branch and trunk nets such that
\[
\|\mathcal{L}(G_\beta)\|_{L^2(L^2(\mathbb{T}^d),\mu)} \leq C \rho^{-\beta}.
\]

Moreover, size($\tau$) $\leq C p^{\frac{d+\sigma}{2}}$, depth($\tau$) = 3, size($\beta$) $\leq p$ and depth($\beta$) = 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 3.9 for every PDE of interest on a case-by-case basis. 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 [31] and [24] 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 3.5). To bound the approximation error of PINNs one can use the generic error bounds in Sobolev norms of e.g. [25, 26] for very general activation functions or the more concrete bounds [33] 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. [21, 22, 30]. For DeepONets, convergence rates for advection-diffusion equations are presented in [17] and a clear workflow for obtaining generic error estimates as well as worked out examples can be found in [43]. Similar results are obtained for FNOs in [35]. A comprehensive comparison of DeepONets and FNOs is the topic of [SU]. To the best of
the authors’ knowledge, no theoretical results for physics-informed operator learning are currently available. Unrelated to the approximation error, we also report generic bounds on the expected value of the generalization error of all the aforementioned deep learning architectures, in the form of an a posteriori error estimate on the generalization error.

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. \[ 23, 50 \], nonlinear parabolic PDEs \[ 32 \] and elliptic PDEs \[ 4, 10, 57 \]. By assuming that the initial data lies in a Barron class, the authors of \[ 52 \] proved for elliptic PDEs that the Deep Ritz Method \[ 20 \] can overcome the CoD. Since the Barron class is a Banach algebra \[ 10 \] 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 \[ 16 \]. 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 4.3). DeepONets and FNOs can overcome the CoD in many cases \[ 43, 38 \] 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. Similar results are expected to hold for e.g. elliptic PDEs by using the results from \[ 3, 10, 57 \].

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 3.4 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 3.5 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. \[ 26, 15 \].

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 \[ 10 \] 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.

Here, we have considered the approximation and generalization errors in the present analysis. It is clear that the bounds on the generalization error may not be sharp, as in traditional deep learning. Obtaining sharper bounds will be an interesting topic for further investigation. Finally, there is no explicit bound on the training (optimization) errors. Obtaining such bounds will be considered in the future.

Acknowledgments and Disclosure of Funding

The research of SM was performed under a project that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (Grant Agreement No. 770880).
References


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]