APPLIED PHYSICS

Learning the solution operator of parametric partial differential equations with physics-informed DeepONets

Sifan Wang[1], Hanwen Wang[1], Paris Perdikaris[2]*

Partial differential equations (PDEs) play a central role in the mathematical analysis and modeling of complex dynamic processes across all corners of science and engineering. Their solution often requires laborious analytical or computational tools, associated with a cost that is markedly amplified when different scenarios need to be investigated, for example, corresponding to different initial or boundary conditions, different inputs, etc. In this work, we introduce physics-informed DeepONets, a deep learning framework for learning the solution operator of arbitrary PDEs, even in the absence of any paired input-output training data. We illustrate the effectiveness of the proposed framework in rapidly predicting the solution of various types of parametric PDEs up to three orders of magnitude faster compared to conventional PDE solvers, setting a previously unexplored paradigm for modeling and simulation of nonlinear and nonequilibrium processes in science and engineering.

INTRODUCTION

As machine learning (ML) methodologies take center stage across diverse disciplines in science and engineering, there is an increased interest in adopting data-driven methods to analyze, emulate, and optimize complex physical systems. The dynamic behavior of such systems is often described by conservation and constitutive laws expressed as systems of partial differential equations (PDEs) (*1*). A classical task then involves the use of analytical or computational tools to solve such equations across a range of scenarios, e.g., different domain geometries, input parameters, and initial and boundary conditions (IBCs). Mathematically speaking, solving these so-called parametric PDE problems involves learning the solution operator that maps variable input entities to the corresponding latent solutions of the underlying PDE system. Tackling this task using traditional tools [e.g., finite element methods (*2*)] bears a formidable cost, as independent simulations need to be performed for every different domain geometry, input parameter, or IBCs. This challenge has motivated a growing literature on reduced-order methods (*3*–*9*) that leverage existing datasets to build fast emulators, often at the price of reduced accuracy, stability, and generalization performance (*10*, *11*). More recently, ML tools are actively developed to infer solutions of PDEs (*12*–*18*); however, most existing tools can only accommodate a fixed given set of input parameters or IBCs. Nevertheless, these approaches have found wide applicability across diverse applications including fluid mechanics (*19*, *20*), heat transfer (*21*, *22*), bioengineering (*23*, *24*), materials (*25*–*28*), and finance (*29*, *30*), showcasing the remarkable effectiveness of ML techniques in learning black box functions, even in high-dimensional contexts (*31*). A natural question then arises: Can ML methods be effective in building fast emulators for solving parametric PDEs?

Solving parametric PDEs requires learning operators (i.e., maps between infinite dimensional function spaces) instead of functions (i.e., maps between finite dimensional vector spaces), thus defining a new and relatively under explored realm for ML-based approaches. Neural operator methods (*32*–*34*) represent the solution map of parametric PDEs as an integral Hilbert-Schmidt operator, whose kernel is parametrized and learned from paired observations, either using local message passing on a graph-based discretization of the physical domain (*32*, *33*) or using global Fourier approximations in the frequency domain (*34*). By construction, neural operators methods are resolution independent (i.e., the model can be queried at any arbitrary input location), but they require large training datasets, while their involved implementation often leads to slow and computationally expensive training loops. More recently, Lu *et al.* (*35*) has presented a novel operator learning architecture coined as DeepONet that is motivated by the universal approximation theorem for operators (*36*, *37*). DeepONets still require large annotated datasets consisting of paired input-output observations, but they provide a simple and intuitive model architecture that is fast to train, while allowing for a continuous representation of the target output functions that is independent of resolution. Beyond deep learning approaches, operator-valued kernel methods (*38*, *39*) have also been demonstrated as a powerful tool for learning nonlinear operators, and they can naturally be generalized to neural networks acting on function spaces (*40*), but their applicability is generally limited due to their computational cost. Here, we should again stress that the aforementioned techniques enable inference in abstract infinite-dimensional Banach spaces (*41*), a paradigm shift from current ML practice that mainly focuses on learning functions instead of operators. Recent theoretical findings also suggest that the sample complexity of deep neural networks (*31*, *42*, *43*), and DeepONets in particular (*44*), can circumvent the curse of dimensionality in certain scenarios.

While the aforementioned methodologies have demonstrated early promise across a range of applications (*45*–*49*), their application to solving parametric PDEs faces two fundamental challenges. First, they require a large corpus of paired input-output observations. In many realistic scenarios, the acquisition of such data involves the repeated evaluation of expensive experiments or costly high-fidelity simulators, thus generating sufficient large training datasets that may be prohibitively expensive. Ideally, one would wish to be able to train such models without any observed data at all (i.e., given only knowledge of the PDE form and its corresponding IBCs). The second challenge relates to the fact that, by construction, the methods outlined above can only return a crude approximation to the target

[1]Graduate Group in Applied Mathematics and Computational Science, University of Pennsylvania, Philadelphia, PA 19104, USA. [2]Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104, USA. *Corresponding author. Email: pgp@seas.upenn.edu

solution operator in the sense that the predicted output functions are not guaranteed to satisfy the underlying PDE. Recent efforts (*16*, *50*–*53*) attempt to address some of these challenges by designing appropriate architectures and loss functions for learning discretized operators (i.e., maps between high-dimensional Euclidean spaces). Although these approaches can relax the requirement for paired input-output training data, they are limited by the resolution of their underlying mesh discretization and, consequently, need modifications to their architecture for different resolutions/discretizations to achieve consistent convergence [if at all possible, as demonstrated in (*32*)].

In this work, we aim to address the aforementioned challenges by exploring a simple yet remarkably effective extension of the DeepONet framework (*35*). Drawing motivation from physics-informed neural networks (*14*), we recognize that the outputs of a DeepONet model are differentiable with respect to their input coordinates, therefore allowing us to use automatic differentiation (*54*, *55*) to formulate an appropriate regularization mechanism for biasing the target output functions to satisfy the underlying PDE constraints. This yields a simple procedure for training physics-informed DeepONet models even in the absence of any training data for the latent output functions, except for the appropriate IBCs of a given PDE system. By constraining the outputs of a DeepONet to approximately satisfy an underlying governing law, we observe substantial improvements in predictive accuracy (up to one to two orders of magnitude reduction in predictive errors), enhanced generalization performance even for out-of-distribution prediction and extrapolation tasks, as well as enhanced data efficiency (up to 100% reduction in the number of examples required to train a DeepONet model). Hence, we demonstrate how physics-informed DeepONet models can be used to solve parametric PDEs without any paired input-output observations, a setting for which existing approaches for operator learning in Banach spaces fall short. Moreover, a trained physics-informed DeepONet model can generate PDE solutions up to three orders of magnitude faster compared to traditional PDE solvers. Together, the computational infrastructure developed in this work can have broad technical impact in reducing computational costs and accelerating scientific modeling of complex nonlinear, nonequilibrium processes across diverse applications including engineering design and control, Earth System science, and computational biology.

RESULTS

The proposed physics-informed DeepONet architecture is summarized in Fig. 1. Motivated by the universal approximation theorem for operators (*35*, *36*), the architecture features two neural networks coined as the "branch" and "trunk" networks, respectively; the automatic differentiation of which enables us to learn the solution operator of arbitrary PDEs. The associated loss functions, performance metrics, computational cost, hyperparameters, and training details are discussed in the Supplementary Materials. In the following, we demonstrate the effectiveness of physics-informed DeepONets across a series of comprehensive numerical studies for solving various types of parametric PDEs. A summary of the different benchmarks considered is presented in Table 1. It is worth emphasizing that, in all cases, the proposed deep learning models are trained without any paired input-output data, assuming only knowledge of the governing equation and its corresponding initial or boundary conditions.

Solving a parametric ordinary differential equation

We begin with a pedagogical example involving the antiderivative operator. The underlying governing law corresponds to an initial value problem described by the following ordinary differential equation (ODE)

$$\frac{ds(x)}{dx} = u(x), x \in [0,1] \quad (1)$$

$$s(0) = 0 \quad (2)$$

Here, we aim to learn the solution operator mapping any forcing term $u(x)$ to the ODE solution $s(x)$ using a physics-informed DeepONet. The model is trained on random realizations of $u(x)$ generated by sampling a Gaussian random field (GRF) as detailed in the Supplementary Materials, while prediction accuracy is measured in new

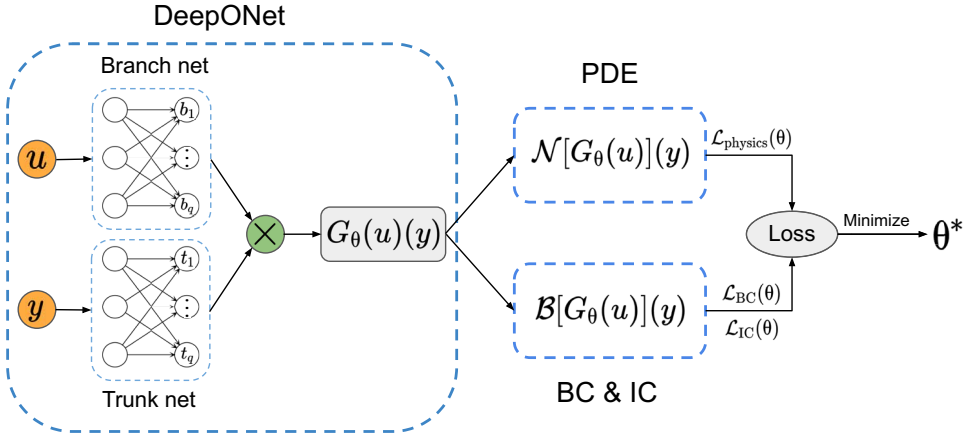

Fig. 1. Making DeepONets physics informed. The DeepONet architecture (*35*) consists of two subnetworks, the branch net for extracting latent representations of input functions and the trunk net for extracting latent representations of input coordinates at which the output functions are evaluated. A continuous and differentiable representation of the output functions is then obtained by merging the latent representations extracted by each subnetwork via a dot product. Automatic differentiation can then be used to formulate appropriate regularization mechanisms for biasing the DeepONet outputs to satisfy a given system of PDEs. BC, boundary conditions; IC, initial conditions.

Table 1. Summary of benchmarks for assessing the performance of physics-informed DeepONets across various types of parametric differential equations. The reported test error corresponds to the relative L^2 prediction error of the trained model, averaged over all examples in the test dataset (see eq. S20).

Governing law	Equation form	Random input	Test error
Linear ODE	$\frac{ds(x)}{dx} = u(x)$	Forcing terms	0.33 ± 0.32%
Diffusion reaction	$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x)$	Source terms	0.45 ± 0.16%
Burgers'	$\frac{\partial s}{\partial t} + s\frac{\partial s}{\partial x} - \nu\frac{\partial^2 s}{x^2} = 0$	Initial conditions	1.38 ± 1.64%
Advection	$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} = 0$	Variable coefficients	2.24 ± 0.68%
Eikonal	$\| \nabla s \|_2 = 0$	Domain geometries	0.42 ± 0.11%

unseen realizations that are not used during model training. Results for one representative input sample $u(x)$ from the test dataset are presented in Fig. 2. It is evident that an excellent agreement can be achieved between the physics-informed DeepONet predictions and the ground truth. More impressively, below, we show that physics-informed DeepONets can also accommodate irregular input functions by using an appropriate neural network architecture, such as a Fourier features network (*56*) for their trunk. As shown in Fig. 2, the predicted solutions $s(x)$ and their corresponding ODE residuals $u(x)$ obtained by a physics-informed DeepONet with a Fourier feature trunk network are in excellent agreement with the exact solutions for this benchmark. Additional systematic studies and visualizations are provided in the Supplementary Materials (see figs. S1 to S11 and tables S6 to S10). On the basis of these observations, we may also conclude that physics-informed DeepONets can be regarded as a class of deep learning models that greatly enhance and generalize the capabilities of physics-informed neural networks (*57*), which are limited to solving ODEs and PDEs for a given set of input parameters that remain fixed during both the training and prediction phases (see tables S7 and S8 for a more detailed comparison).

It is also worth pointing out that the trained physics-informed DeepONet is even capable of yielding accurate predictions for out-of-distribution test data. To illustrate this, we create a test dataset by sampling input functions from a GRF with a larger length scale of $l = 0.2$ (recall that the training data for this case is generated using $l = 0.01$). The corresponding relative L^2 prediction error averaged over 1000 test examples is measured as 0.7%. Additional visualizations of the model predictions for this out-of-distribution prediction task can be found in the Supplementary Materials (fig. S9).

Diffusion-reaction dynamics

Our next example involves an implicit operator described by a nonlinear diffusion-reaction PDE with a source term $u(x)$

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), (x,t) \in (0,1] \times (0,1] \quad (3)$$

assuming zero IBCs, while $D = 0.01$ is the diffusion coefficient and $k = 0.01$ is the reaction rate. Here, we aim to learn the solution operator for mapping source terms $u(x)$ to the corresponding PDE solutions $s(x)$. The model is trained on random realizations of $u(x)$ generated by sampling a GRF as detailed in the Supplementary Materials, while prediction accuracy is measured in new unseen realizations that are not used during model training.

The top panels of Fig. 3 show the comparison between the predicted and the exact solution for a random test input sample. More visualizations for different input samples can be found in the Supplementary Materials (fig. S12). We observe that the physics-informed DeepONet predictions achieve an excellent agreement with the corresponding reference solutions. Furthermore, we provide a comparison against the conventional DeepONet formulation recently put forth by Lu *et al.* (*35*). This case necessitates observations of paired input-output pairs $[u(x), s(x, t)]$ to be provided as training data, as no physical constraints are leveraged during model training. The mean and SD of relative L^2 errors of the conventional DeepONet and physics-informed DeepONet over the test dataset are visualized in the bottom panel of Fig. 3. The average relative L^2 error of DeepONet and physics-informed DeepONet are ~1.92 and ~0.45%, respectively. In contrast to the conventional DeepONet that is trained on paired input-output measurements, the proposed physics-informed DeepONet can yield much more accurate predictions even without any paired training data (except for the specified IBCs). In our experience, predictive accuracy can be generally improved by using a larger batch size during training. A study of the effect of batch size for training physics-informed DeepONets can be found in the Supplementary Materials (figs. S13 and S16). A series of convergence studies aiming to illustrate how predictive accuracy is affected by the number of input sensor locations m and different neural network architectures is also presented in the Supplementary Materials (fig. S14).

Burgers' transport dynamics

To highlight the ability of the proposed framework to handle nonlinearity in the governing PDEs, we consider the one-dimensional (1D) Burgers' benchmark investigated in Li *et al.* (*34*)

$$\frac{ds}{dt} + s\frac{ds}{dx} - \nu\frac{d^2 s}{dx^2} = 0, (x,t) \in (0,1) \times (0,1] \quad (4)$$

$$s(x,0) = u(x), x \in (0,1) \quad (5)$$

with periodic boundary conditions

$$s(0,t) = s(1,t) \quad (6)$$

$$\frac{ds}{dx}(0,t) = \frac{ds}{dx}(1,t) \quad (7)$$

where $t \in (0,1)$, the viscosity is set to $\nu = 0.01$, and the initial condition $u(x)$ is generated from a GRF $\sim\mathcal{N}(0, 25^2(-\Delta + 5^2 I)^{-4})$, satisfying the periodic boundary conditions.

Our goal here is to use the proposed physics-informed DeepONet model to learn the solution operator mapping initial conditions $u(x)$ to the full spatiotemporal solution $s(x, t)$ of the 1D Burgers' equation. To this end, the model is trained on random realizations of $u(x)$ generated by sampling a GRF as detailed in the Supplementary Materials, while prediction accuracy is measured in new unseen realizations that are not used during model training.

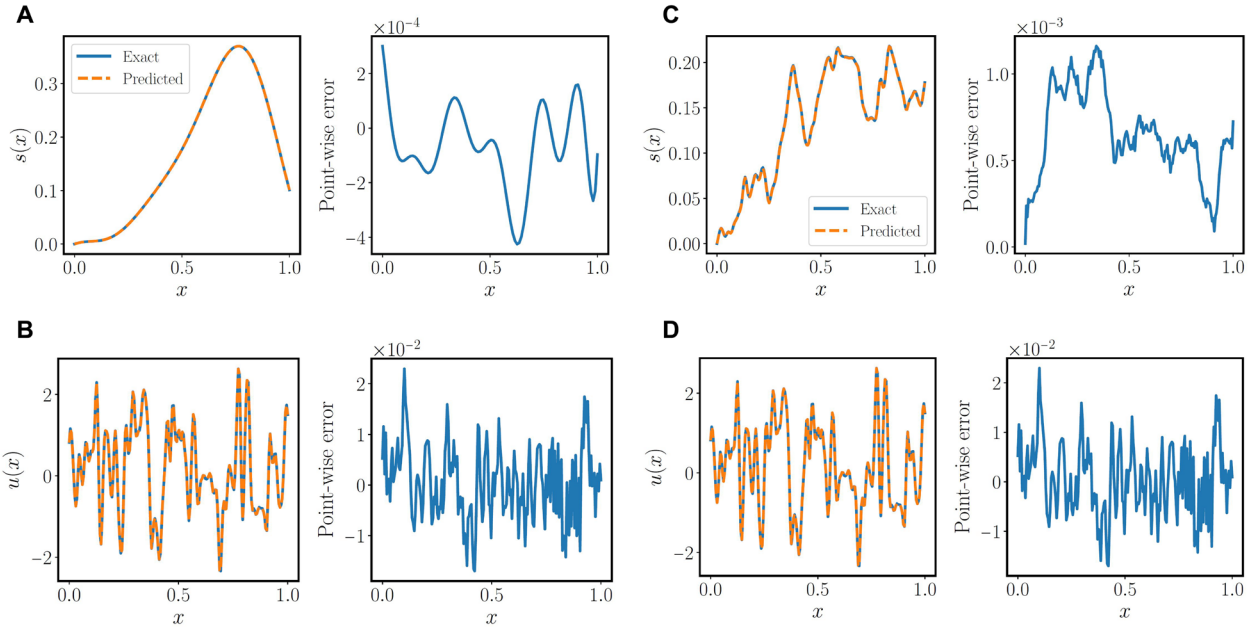

Fig. 2. Solving a one-dimensional parametric ODE. (**A** and **B**) Exact solution and residual versus the predictions of a trained physics-informed DeepONet for a representative input function sampled from a GRF with length scale $l = 0.2$. (**C** and **D**) Exact solutions and corresponding ODE residuals versus the predictions of a trained physics-informed DeepONet with Fourier feature embeddings (*56*) for a representative input function sampled from a GRF with length scale $l = 0.01$. The predicted residual $u(x)$ is computed via automatic differentiation (*55*).

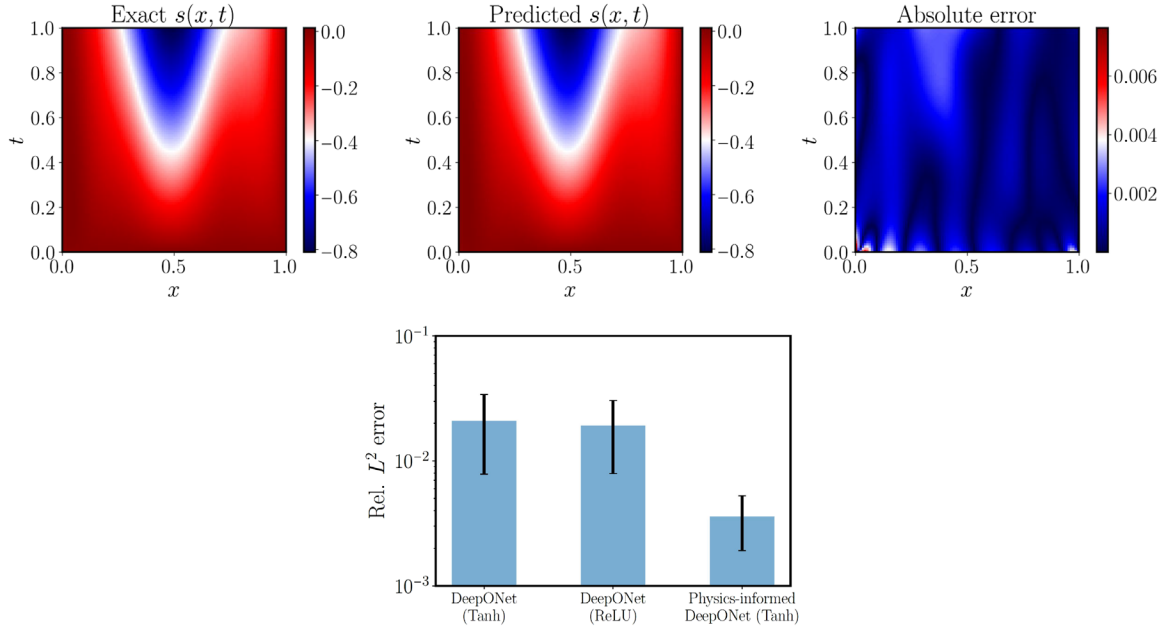

Fig. 3. Solving a parametric diffusion-reaction system. (**Top**) Exact solution versus the prediction of a trained physics-informed DeepONet for a representative example in the test dataset. (**Bottom**) Mean and SD of the relative L^2 prediction error of a trained DeepONet (with paired input-output training data) and a physics-informed DeepONet (without paired input-output training data) over 1000 examples in the test dataset. The mean and SD of the relative L^2 prediction are ~1.92 ± 1.12% (DeepONet) and ~0.45 ± 0.16% (physics-informed DeepONet), respectively. The physics-informed DeepONet yields ~80% improvement in prediction accuracy with 100% reduction in the dataset size required for training. Tanh, hyperbolic tangent; ReLU, rectified linear unit.

The average relative L^2 error of the best trained model is ~1.38% (see figs. S17 to S19). The physics-informed DeepONet achieves the comparable accuracy compared to Fourier operator methods (*34*), albeit the latter has been only tested for a simpler case corresponding to $\nu = 0.1$ and requires training on a large corpus of paired input-output data. Furthermore, visualizations corresponding to the worst example in the test dataset are shown in the top panels of Fig. 4. One can see that the predicted solution achieves a good agreement

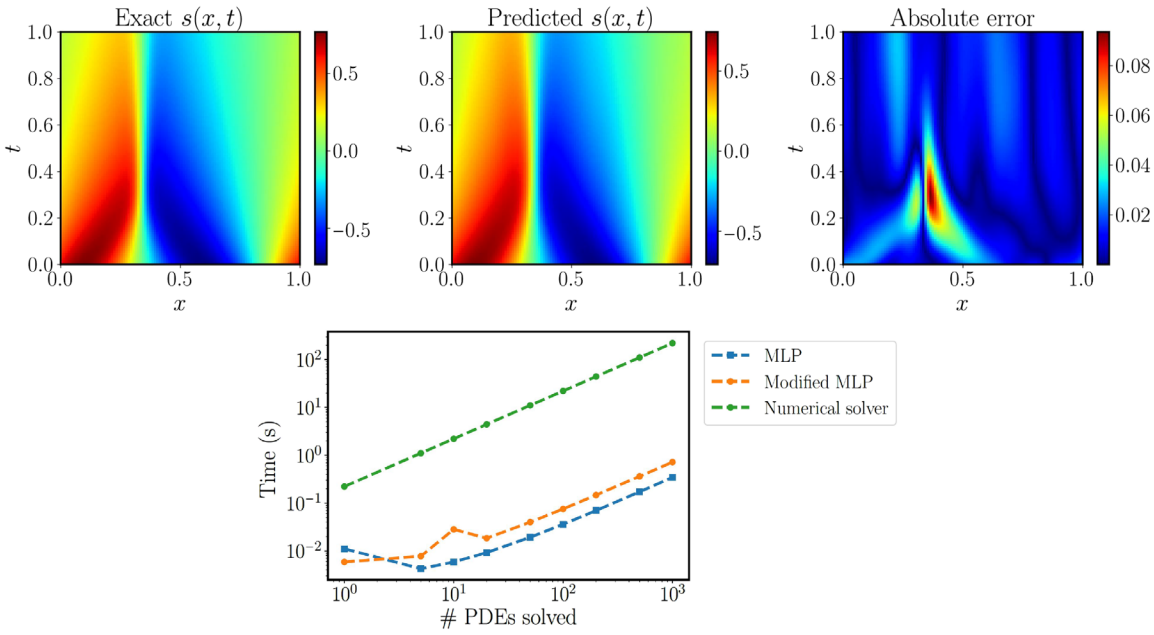

Fig. 4. Solving a parametric Burgers' equation. (**Top**) Exact solution versus the prediction of the best-trained physics-informed DeepONet. The resulting relative L^2 error of the predicted solution is 3%. (**Bottom**) Computational cost (s) for performing inference with a trained physics-informed DeepONet model [conventional or modified multilayer perceptron (MLP) architecture], as well as corresponding timing for solving a PDE with a conventional spectral solver (*58*). Notably, a trained physics-informed DeepONet model can predict the solution of $\mathcal{O}(10^3)$ time-dependent PDEs in a fraction of a second, up to three orders of magnitude faster compared to a conventional PDE solver. Reported timings are obtained on a single NVIDIA V100 graphics processing unit (GPU).

against the reference solution, with a the relative L^2 error of 3.30%. Here, we must also emphasize that a trained physics-informed DeepONet model can rapidly predict the entire spatiotemporal solution of the Burgers equation in ~10 ms. Inference with physics-informed DeepONets is trivially parallelizable, allowing for the solution of $\mathcal{O}(10^3)$ PDEs in a fraction of a second, yielding up to three orders of magnitude in speed up compared to a conventional spectral solver (*58*), see the bottom panel of Fig. 4.

Despite the promising results presented here, we must note the need for further methodological advances toward enhancing the accuracy and robustness of physics-informed DeepONets in tackling PDE systems with stiff, turbulent, or chaotic dynamics. For example, we have observed that the predictive accuracy of physics-informed DeepONets degrades in regions where the PDE solution exhibits steep gradients; a behavior that is pronounced as the viscosity parameter in the Burgers equation is further decreased (see fig. S20 and table S11 for more details and quantitative results). We conjecture that these issues can be tackled in the future by designing of more specialized architectures that are tailored to the dynamic behavior of a given PDE, as well as more effective optimization algorithms for training.

Advection equation

This example aims to investigate the performance of physics-informed DeepONets for tackling advection-dominated PDEs; a setting for which conventional approaches to reduced-order modeling faces significant challenges (*7*, *10*, *11*). To this end, we consider a linear advection equation with variable coefficients

$$\frac{\partial s}{\partial t} + u(x)\frac{\partial s}{\partial x} = 0, (x,t) \in (0,1) \times (0,1) \tag{8}$$

with the IBC

$$s(x,0) = f(x) \tag{9}$$

$$s(0,t) = g(t) \tag{10}$$

where $f(x) = \sin(\pi x)$ and $g(t) = \sin\left(\frac{\pi}{2}t\right)$. To make the input function $u(x)$ strictly positive, we let $u(x) = v(x) - \min_x v(x) + 1$, where $v(x)$ is sampled from a GRF with a length scale $l = 0.2$. The goal is to learn the solution operator G mapping variable coefficients $u(x)$ to associated solutions $s(x, t)$ (see the Supplementary Materials for more details).

As shown in Fig. 5, the trained physics-informed DeepONet is able to achieve an overall good agreement with the reference PDE solution, although some inaccuracies can be observed in regions where the solution exhibits steep gradients (similarly to the Burgers' example discussed above; see additional visualizations presented in fig. S21). The resulting relative L^2 prediction averaged over all examples in the test dataset is 2.24%, leading to the conclusion that physics-informed DeepONets can be effective surrogates for advection-dominated PDEs.

Eikonal equation

Our last example aims to highlight the capability of the proposed physics-informed DeepONet to handle different types of input functions. To this end, let us consider a 2D eikonal equation of the form

$$\begin{aligned} \|\nabla s(\mathbf{x})\|_2 &= 1 \\ s(\mathbf{x}) &= 0, \mathbf{x} \in \partial\Omega \end{aligned} \tag{11}$$

where $\mathbf{x} = (x, y) \in \mathbb{R}^2$ denotes 2D spatial coordinates, and Ω is an open domain with a piece-wise smooth boundary $\partial\Omega$. A solution to the above equation is a signed distance function measuring the distance of a point in Ω to the closest point on the boundary $\partial\Omega$, i.e.

$$s(\mathbf{x}) = \begin{cases} d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \in \Omega \\ -d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \in \Omega^c \end{cases}$$

where $d(\cdot, \cdot)$ is a distance function defined as

$$d(\mathbf{x}, \partial\Omega) \coloneqq \inf_{\mathbf{y} \in \partial\Omega} d(\mathbf{x}, \mathbf{y}) \tag{12}$$

Signed distance functions (SDFs) have recently sparked increased interest in the computer vision and graphics communities as a tool for shape representation learning (*59*). This is because SDFs can continuously represent abstract shapes or surfaces implicitly as their zero-level set, yielding high-quality shape representations, interpolation, and completion from partial and noisy input data (*59*). In this example, we seek to learn the solution map from a well-behaved closed curve Γ to its associated signed distance function, i.e., the solution of the eikonal equation defined in Eq. 11. As a benchmark we consider different airfoil geometries from the University of Illinois--Urbana-Champaign (UIUC) database (*60*), a subset of which is used to train the model (see the Supplementary Materials for more details).

The trained DeepONet model is then capable of predicting the solution of the eikonal equation for any given input airfoil geometry. To evaluate its performance, we visualize the zero-level set of the learned signed distance function and compare it with the exact airfoil geometry. As shown in Fig. 6, the zero-level sets achieve a good agreement with the exact airfoil geometries. One may conclude that the proposed framework is capable of achieving an accurate approximation of the exact signed distance function. Additional systematic studies and quantitative comparisons are provided in the Supplementary Materials (see figs. S23 to S25).

DISCUSSION

This paper presents physics-informed DeepONets, a novel deep learning framework for approximating nonlinear operators in infinite-dimensional Banach spaces. Leveraging automatic differentiation, we present a simple yet remarkably effective mechanism for biasing the outputs of DeepONets toward physically consistent predictions, allowing us to realize significant improvements in predictive accuracy, generalization performance, and data efficiency compared to existing operator learning techniques. An even more intriguing finding is that physics-informed DeepONets can learn the solution operator of parametric ODEs and PDEs, even in the absence of any paired input-output training data. This capability is introducing a new radical way of simulating nonlinear and non-equilibrium phenomena across different applications in science and engineering up to three orders of magnitude faster compared to conventional solvers.

Given the prominent role that PDEs play in the mathematical analysis, modeling, and simulation of complex physical systems, the

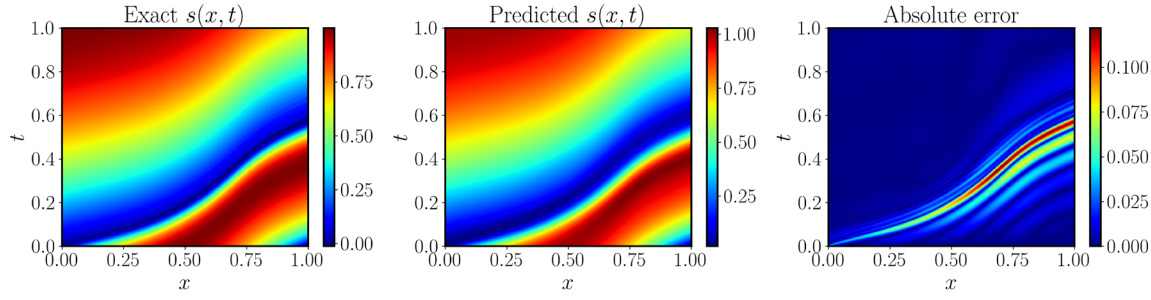

Fig. 5. Solving a parametric advection equation. Exact solution versus the prediction of a trained physics-informed DeepONet for a representative example in the test dataset.

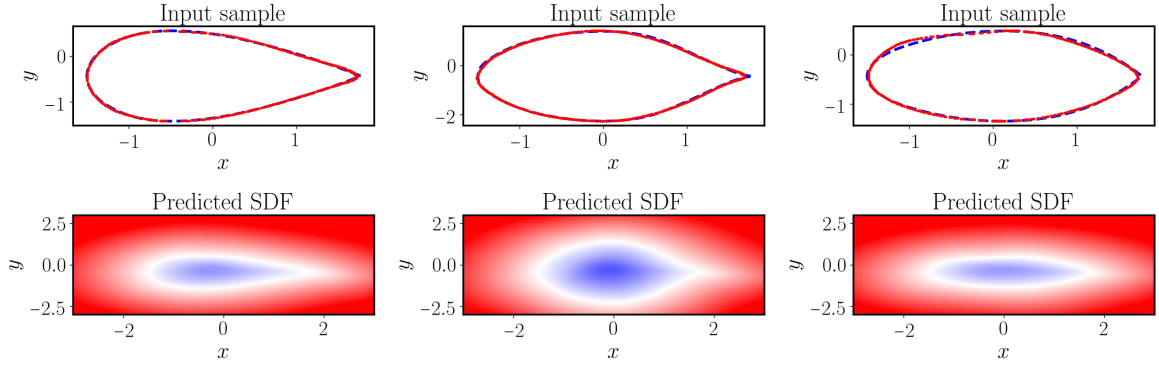

Fig. 6. Solving a parametric eikonal equation (airfoils). (**Top**) Exact airfoil geometry versus the zero-level set obtained from the predicted signed distance function for three different input examples in the test dataset. (**Bottom**) Predicted signed distance function of a trained physics-informed DeepONet for three different airfoil geometries in the test dataset.

physics-informed DeepONet architecture can be broadly applied in science and engineering since PDEs are prevalent across diverse problem domains including fluid mechanics, electromagnetics, quantum mechanics, and elasticity. However, despite the early promise demonstrated here, numerous technical questions remain open and require further investigation. Motivated by the successful application of Fourier feature networks (*56*), it is natural to ask the following: For a given parametric governing law, what is the optimal features embedding or network architecture of a physics-informed DeepONet? Recently, Wang *et al.* (*61*) proposed a multiscale Fourier feature network to tackle PDEs with multiscale behavior. Such an architecture may be potentially used as the backbone of physics-informed DeepONets to learn multiscale operators and solve multiscale parametric PDEs. Another question arises from the possibility of achieving improved performance by assigning weights in the physics-informed DeepONet loss function. It has been shown that these weights play an important role in enhancing the trainability of constrained neural networks (*62*–*64*). Therefore, it is natural to ask the following: What are the appropriate weights to use for training physics-informed DeepONets? How to design effective algorithms for accelerating training and ensuring accuracy and robustness in the predicted outputs? We believe that addressing these questions will not only enhance the performance of physics-informed DeepONets but also introduce a paradigm shift in how we model and simulate complex, nonlinear, and multiscale physical systems across diverse applications in science and engineering.

METHODS

DeepONets (*35*) present a specialized deep learning architecture that encapsulates the universal approximation theorem for operators (*36*). Here, we illustrate how DeepONets can be effectively applied to learning the solution operator of parametric PDEs. Here, the terminology "parametric PDEs" refers to the fact that some parameters of a given PDE system are allowed to vary over a certain range. These input parameters may include, but are not limited to, the shape of the physical domain, the initial or boundary conditions, constant or variable coefficients (e.g., diffusion or reaction rates), source terms, etc. To describe such problems in their full generality, let $(\mathcal{U}, \mathcal{V}, \mathcal{S})$ be a triplet of Banach spaces and $\mathcal{N}: \mathcal{U} \times \mathcal{S} \to \mathcal{V}$ be a linear or nonlinear differential operator. We consider general parametric PDEs taking the form

$$\mathcal{N}(\mathbf{u}, \mathbf{s}) = 0 \tag{13}$$

where $\mathbf{u} \in \mathcal{U}$ denotes the parameters (i.e., input functions) and $\mathbf{s} \in \mathcal{S}$ denotes the corresponding unknown solutions of the PDE system. Specifically, we assume that, for any $\mathbf{u} \in \mathcal{U}$, there exists an unique solution $\mathbf{s} = \mathbf{s}(\mathbf{u}) \in \mathcal{S}$ to 13 (subject to appropriate IBCs). Then, we can define the solution operator $G: \mathcal{U} \to \mathcal{S}$ as

$$G(\mathbf{u}) = \mathbf{s}(\mathbf{u}) \tag{14}$$

Following the original formulation of Lu *et al.* (*35*), we represent the solution map G by an unstacked DeepONet G_θ, where θ denotes all trainable parameters of the DeepONet network. As illustrated in Fig. 1, the unstacked DeepONet is composed of two separate neural networks referred to as the branch and trunk networks, respectively. The branch network takes $\mathbf{u}$ as input and returns a features embedding $[b_1, b_2, \ldots, b_q]^T \in \mathbb{R}^q$ as output, where $\mathbf{u} = [\mathbf{u}(\mathbf{x}_1), \mathbf{u}(\mathbf{x}_2), \ldots, \mathbf{u}(\mathbf{x}_m)]$ represents a function $\mathbf{u} \in \mathcal{U}$ evaluated at a collection of fixed locations $\{\mathbf{x}_i\}_{i=1}^m$. The trunk network takes the continuous coordinates $\mathbf{y}$ as inputs and outputs a features embedding $[t_1, t_2, \ldots, t_q]^T \in \mathbb{R}^q$. To obtain the final output of the DeepONet, the outputs of the branch and trunk networks are merged together via a dot product. More specifically, a DeepONet G_θ prediction of a function $\mathbf{u}$ evaluated at $\mathbf{y}$ can be expressed by

$$G_\theta(\mathbf{u})(\mathbf{y}) = \sum_{k=1}^{q} \underbrace{b_k(\mathbf{u}(\mathbf{x}_1), \mathbf{u}(\mathbf{x}_2), \ldots, \mathbf{u}(\mathbf{x}_m))}_{\text{branch}} \underbrace{t_k(\mathbf{y})}_{\text{trunk}} \tag{15}$$

where θ denotes the collection of all trainable weight and bias parameters in the branch and trunk networks.

Notice that the outputs of a DeepONet model are continuously differentiable with respect to their input coordinates. Therefore, one may use automatic differentiation (*54*, *55*) to formulate an appropriate regularization mechanism for biasing the target output functions to satisfy any given differential constraints.

Consequently, we may then construct a "physics-informed" DeepONet by formulating the following loss function

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{operator}}(\theta) + \mathcal{L}_{\text{physics}}(\theta) \tag{16}$$

where

$$\mathcal{L}_{\text{operator}}(\theta) = \frac{1}{NP} \sum_{i=1}^{N} \sum_{j=1}^{P} | G_\theta(\mathbf{u}^{(i)})(\mathbf{y}_{u,j}^{(i)}) - G(\mathbf{u}^{(i)})(\mathbf{y}_{u,j}^{(i)}) |^2 \tag{17}$$

$$\mathcal{L}_{\text{physics}}(\theta) = \frac{1}{NQm} \sum_{i=1}^{N} \sum_{j=1}^{Q} \sum_{k=1}^{m} | \mathcal{N}(u^{(i)}(\mathbf{x}_k), G_\theta(\mathbf{u}^{(i)})(\mathbf{y}_{r,j}^{(i)})) |^2 \tag{18}$$

Here, $\{\mathbf{u}^{(i)}\}_{i=1}^N$ denotes N separate input functions sampled from $\mathcal{U}$. For each $\mathbf{u}^{(i)}$, $\{\mathbf{y}_{u,j}^{(i)}\}_{j=1}^P$ are P locations that are determined by the data observations, initial or boundary conditions, etc. Besides, $\{\mathbf{y}_{r,j}^{(i)}\}_{j=1}^Q$ is a set of collocation points that can be randomly sampled in the domain of $G(\mathbf{u}^{(i)})$. As a consequence, $\mathcal{L}_{\text{operator}}(\theta)$ fits the available solution measurements while $\mathcal{L}_{\text{physics}}(\theta)$ enforces the underlying PDE constraints. Contrary to the fixed sensor locations of $\{x_i\}_{i=1}^m$, we remark that the locations of $\{\mathbf{y}_{u,j}^{(i)}\}_{j=1}^P$ and $\{\mathbf{y}_{r,j}^{(i)}\}_{j=1}^Q$ may vary for different i, thus allowing us to construct a continuous representation of the output functions $\mathbf{s} \in \mathcal{S}$. More details on how this general framework can be adapted the different PDE systems presented in Results—including the choice of neural network architectures, formulation of loss functions, and training details—are provided in the Supplementary Materials.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at https://science.org/doi/10.1126/sciadv.abi8605

REFERENCES AND NOTES

1. R. Courant, D. Hilbert, *Methods of Mathematical Physics: Partial Differential Equations* (John Wiley & Sons, 2008).
2. T. J. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis* (Courier Corporation, 2012).

3. D. J. Lucia, P. S. Beran, W. A. Silva, Reduced-order modeling: New approaches for computational physics. *Prog. Aerosp. Sci.* **40**, 51–117 (2004).
4. J. N. Kutz, S. L. Brunton, B. W. Brunton, J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (SIAM, 2016).
5. P. Benner, M. Ohlberger, A. Patera, G. Rozza, K. Urban, *Model Reduction of Parametrized Systems* (Springer, 2017).
6. W. H. Schilders, H. A. Van der Vorst, J. Rommes, in *Model Order Reduction: Theory, Research Aspects and Applications* (Springer, 2008), vol. 13.
7. A. Quarteroni, G. Rozza, in *Reduced Order Methods for Modeling and Computational Reduction* (Springer, 2014), vol. 9.
8. I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dyn.* **41**, 309–325 (2005).
9. B. Peherstorfer, K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction. *Comput. Methods Appl. Mech. Eng.* **306**, 196–215 (2016).
10. A. J. Majda, D. Qi, Strategies for reduced-order models for predicting the statistical responses and uncertainty quantification in complex turbulent dynamical systems. *SIAM Rev.* **60**, 491–549 (2018).
11. T. Lassila, A. Manzoni, A. Quarteroni, G. Rozza, Model order reduction in fluid dynamics: Challenges and perspectives, in *Reduced Order Methods for Modeling and Computational Reduction* (Springer, 2014), pp. 235–273.
12. D. C. Psichogios, L. H. Ungar, A hybrid neural network-first principles approach to process modeling. *AIChE J.* **38**, 1499–1511 (1992).
13. I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **9**, 987–1000 (1998).
14. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
15. L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Eng.* **361**, 112732 (2020).
16. Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *J. Comput. Phys.* **394**, 56–81 (2019).
17. S. Karumuri, R. Tripathy, I. Bilionis, J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks. *J. Comput. Phys.* **404**, 109120 (2020).
18. J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **375**, 1339–1364 (2018).
19. M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **367**, 1026–1030 (2020).
20. A. Tartakovsky, C. O. Marrero, P. Perdikaris, G. Tartakovsky, D. Barajas-Solano, Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resour. Res.* **56**, e2019WR026731 (2020).
21. O. Hennigh, S. Narasimhan, M. A. Nabian, A. Subramaniam, K. Tangsali, M. Rietmann, J. del Aguila Ferrandis, W. Byeon, Z. Fang, S. Choudhry, NVIDIA SimNet: An ai-accelerated multi-physics simulation framework. arXiv:2012.07938 (2020).
22. S. Cai, Z. Wang, S. Wang, P. Perdikaris, G. Karniadakis, Physics-informed neural networks (pinns) for heat transfer problems. *J. Heat Transfer* **143**, 060801 (2021).
23. G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **358**, 112623 (2020).
24. F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, E. Kuhl, Physics-informed neural networks for cardiac activation mapping. *Front. Phys.* **8**, 42 (2020).
25. L. Lu, M. Dao, P. Kumar, U. Ramamurty, G. E. Karniadakis, S. Suresh, Extraction of mechanical properties of materials through deep learning from instrumented indentation. *Proc. Natl. Acad. Sci. U.S.A.* **117**, 7052–7062 (2020).
26. Y. Chen, L. Lu, G. E. Karniadakis, L. Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express* **28**, 11618–11633 (2020).
27. S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theor. Appl. Fract. Mech.* **106**, 102447 (2020).
28. D. Z. Huang, K. Xu, C. Farhat, E. Darve, Learning constitutive relations from indirect observations using deep neural networks. *J. Comput. Phys.* **416**, 109491 (2020).
29. D. Elbrächter, P. Grohs, A. Jentzen, C. Schwab, Dnn expression rate analysis of high-dimensional PDEs: Application to option pricing. arXiv:1809.07669 (2018).
30. J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci. U.S.A.* **115**, 8505–8510 (2018).
31. T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, Q. Liao, Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017).
32. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Graph kernel network for partial differential equations. arXiv:2003.03485 (2020).
33. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Multipole graph neural operator for parametric partial differential equations. arXiv:2006.09535 (2020).
34. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations. arXiv:2010.08895 (2020).
35. L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
36. T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Netw.* **6**, 911–917 (1995).
37. A. D. Back, T. Chen, Universal approximation of multiple nonlinear operators by neural networks. *Neural Comput.* **14**, 2561–2566 (2002).
38. H. Kadri, E. Duflos, P. Preux, S. Canu, A. Rakotomamonjy, J. Audiffren, Operator-valued kernels for learning from functional response data. *J. Mach. Learn. Res.* **17**, 1–54 (2016).
39. M. Griebel, C. Rieger, Reproducing kernel hilbert spaces for parametric partial differential equations. *SIAM-ASA J. Uncertain. Quantif.* **5**, 111–137 (2017).
40. H. Owhadi, Do ideas have shape? plato's theory of forms as the continuous limit of artificial neural networks. arXiv:2008.03920 (2020).
41. N. H. Nelsen, A. M. Stuart, The random feature model for input-output maps between banach spaces. arXiv:2005.10224 (2020).
42. C. Schwab, J. Zech, Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Anal. Appl.* **17**, 19–55 (2019).
43. S. Wojtowytsch, W. E, Can shallow neural networks beat the curse of dimensionality? A mean field training perspective. *IEEE Trans. Artif. Intell.* **1**, 121–129 (2021).
44. S. Lanthaler, S. Mishra, G. E. Karniadakis, Error estimates for deeponets: A deep learning framework in infinite dimensions. arXiv:2102.09618 (2021).
45. S. Cai, Z. Wang, L. Lu, T. A. Zaki, G. E. Karniadakis, DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. arXiv:2009.12935 (2020).
46. C. Lin, Z. Li, L. Lu, S. Cai, M. Maxey, G. E. Karniadakis, Operator learning for predicting multiscale bubble growth dynamics. arXiv:2012.12816 (2020).
47. B. Liu, N. Kovachki, Z. Li, K. Azizzadenesheli, A. Anandkumar, A. Stuart, K. Bhattacharya, A learning-based multiscale method and its application to inelastic impact problems. arXiv:2102.07256 (2021).
48. P. C. Di Leoni, L. Lu, C. Meneveau, G. Karniadakis, T. A. Zaki, Deeponet prediction of linear instability waves in high-speed boundary layers. arXiv:2105.08697 (2021).
49. Z. Mao, L. Lu, O. Marxen, T. A. Zaki, 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. arXiv:2011.03349 (2020).
50. Y. Khoo, J. Lu, L. Ying, Solving parametric PDE problems with artificial neural networks. arXiv:1707.03351 (2017).
51. N. Geneva, N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *J. Comput. Phys.* **403**, 109056 (2020).
52. Y. Chen, B. Dong, J. Xu, Meta-mgnet: Meta multigrid networks for solving parameterized partial differential equations. arXiv:2010.14088 (2020).
53. D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, S. Hoyer, Machine learning accelerated computational fluid dynamics. arXiv:2102.01010 (2021).
54. A. Griewank, On automatic differentiation, in *Mathematical Programming: Recent Developments and Applications* (Kluwer Academic Publishers, 1989), pp. 83–108.
55. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* **18**, 1–43 (2018).
56. M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, R. Ng, Fourier features let networks learn high frequency functions in low dimensional domains. arXiv:2006.10739 (2020).
57. M. Raissi, H. Babaee, P. Givi, Deep learning of turbulent scalar mixing. *Phys. Rev. Fluids* **4**, 124501 (2019).
58. T. A. Driscoll, N. Hale, L. N. Trefethen, Chebfun guide (2014).
59. J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 165–174.
60. M. S. Selig, Uiuc airfoil data site (1996).
61. S. Wang, H. Wang, P. Perdikaris, On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. arXiv:2012.10047 (2020).
62. S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv:2001.04536 (2020).

63. S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective. arXiv:2007.14527 (2020).
64. L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism. arXiv:2009.04544 (2020).
65. J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: Composable transformations of Python+NumPy programs (2018).
66. J. D. Hunter, Matplotlib: A 2D graphics environment. *IEEE Ann. Hist. Comput.* **9**, 90–95 (2007).
67. C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with numpy. *Nature* **585**, 357–362 (2020).
68. C. Rasmussen, C. Williams, *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning (MIT Press, 2006).
69. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. arXiv:1412.6980 (2014).
70. C. Finn, P. Abbeel, S. Levine, *International Conference on Machine Learning* (PMLR, 2017), pp. 1126–1135.
71. A. Iserles, in *A First Course in the Numerical Analysis of Differential Equations* (Cambridge Univ. Press, 2009), no. 44.
72. E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput. Methods Appl. Mech. Eng.* **379**, 113741 (2021).
73. S. Wang, P. Perdikaris, Long-time integration of parametric evolution equations with physics-informed deeponets. arXiv:2106.05384 (2021).
74. S. M. Cox, P. C. Matthews, Exponential time differencing for stiff systems. *J. Comput. Phys.* **176**, 430–455 (2002).

Acknowledgments: We thank the developers of the software that enabled our research, including JAX (*65*), Matplotlib (*66*), and NumPy (*67*). **Funding:** This work received support from DOE grant DE-SC0019116, AFOSR grant FA9550-20-1-0060, and DOE-ARPA grant DE-AR0001201. **Author contributions:** S.W. and P.P. conceptualized the research and designed the numerical studies. S.W. and H.W. implemented the methods and conducted the numerical experiments. P.P. provided funding and supervised all aspects of this work. All authors contributed in writing the manuscript. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. All code and data accompanying this manuscript are publicly available at https://doi.org/10.5281/zenodo.5206676 and https://github.com/PredictiveIntelligenceLab/Physics-informed-DeepONets.

Submitted 5 April 2021
Accepted 3 August 2021
Published 29 September 2021
10.1126/sciadv.abi8605

Citation: S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Sci. Adv.* **7**, eabi8605 (2021).