Physics-guided Data Augmentation for Learning the Solution Operator of Linear Differential Equations

Ye Li, Yiwen Pang, and Bin Shan

Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China
yeli20@nuaa.edu.cn

Abstract: Neural networks, especially the recent proposed neural operator models, are increasingly being used to find the solution operator of differential equations. Compared to traditional numerical solvers, they are much faster and more efficient in practical applications. However, one critical issue is that training neural operator models require large amount of ground truth data, which usually comes from the slow numerical solvers. In this paper, we propose a physics-guided data augmentation (PGDA) method to improve the accuracy and generalization of neural operator models. Training data is augmented naturally through the physical properties of differential equations such as linearity and translation. We demonstrate the advantage of PGDA on a variety of linear differential equations, showing that PGDA can improve the sample complexity and is robust to distributional shift.

Keywords: Data augmentation; Neural operator; Differential equation

1 Introduction

While there is a rapid growth in machine learning in the last 20 years, most researchers concentrated on image recognition and natural language processing tasks [1, 2], and the development of simulating the physical world around us by deep learning is relatively recent [3]. Many physical systems are described by partial differential equations (PDEs). Solving the underlying PDEs is usually analytically intractable. Although there are numerous numerical solvers developed by many mathematicians and physicists in the last half century, classical numerical solvers are expensive in computation and custom designed per PDE class [4].

Recently, there has been much work applying deep learning to learn PDE solvers [5, 6, 7], in which neural operators [8, 9] have drawn the attention of many researchers for their strong ability to learn the mapping between infinite functional spaces. However, current approaches need abundant high-quality data to generalize well, otherwise real-world out-of-distribution test data is difficult to align with training data, see Figure 1 for an illustration. In reality, we have to train a well-generalized neural operator model with limited 'expensive' training data. One potential route is to incorporate symmetries into the forecasting model to improve generalization by Wang et al. [10, 11]. This leads to the so-called equivariant network. However, the design of equivariant layers is a difficult task.

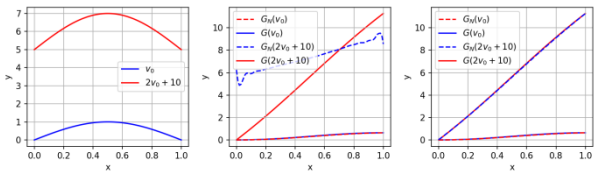

Figure 1 An illustrative neural operator learning of the antiderivative operator $\mathcal{G}: v(x) \mapsto u(x) = \int_0^x v(\tau)d\tau$ in Section 5.1. (**Left**) Input function $v(x) = v_0(x) = \sin(\pi x)$. (**Middle**) $\mathcal{G}_\mathcal{N}$ without PGDA, then $\mathcal{G}_\mathcal{N}(v_0)$ is well predicted but $\mathcal{G}_\mathcal{N}(2v_0 + 10)$ failed. (**Right**) $\mathcal{G}_\mathcal{N}$ with PGDA, both $\mathcal{G}_\mathcal{N}(v_0)$ and $\mathcal{G}_\mathcal{N}(2v_0 + 10)$ are well predicted.

We propose a simple yet effective data augmentation technique to improve the accuracy and generalization of neural operator models. Training data are augmented naturally through the physical properties of differential equations such as linearity and translation. We name it by physics-guided data augmentation (PGDA), with details in Section 4.2. Our contributions include:

- We study the problem of improving the generalization capability of deep learning models for learning the solution operator of different linear differential equations (of evolutionary/stationary type).
- We design a physics-guided data augmentation (PGDA) method for the training of neural operators. PGDA is based on the physical properties of differential equations such as linearity and translation.
- We demonstrate experimentally that PGDA can improve prediction accuracy under distribution shift up to $1 \sim 4$ orders of magnitude on a SOTA neural operator model.

2 Background

Here we present the problem setup of solution operator learning of PDEs. Without loss of generality, we formulate any PDEs as:

$$\begin{cases} \mathcal{L}u(x) = f(x), & x \in \Omega \subset \mathbb{R}^n, \\ \mathcal{B}u(x) = v(x), & x \in \partial\Omega, \end{cases} \tag{1}$$

where $\mathcal{L}$ is the partial differential operator and $\mathcal{B}$ is the boundary condition. Throughout this work we will consider the case where $\mathcal{L}$ and $\mathcal{B}$ are linear operators.

Let $\mathcal{V}$ and $\mathcal{U}$ be the spaces of function $v(x)$ and function $u(x)$, respectively. Then, the mapping $\mathcal{G}: \mathcal{V} \to \mathcal{U}$ with $\mathcal{G}(v)(x) = u(x)$ is referred to the solution operator of the PDE (1). Since the solution operator $\mathcal{G}$ is always nonanalytic, we try to learn an approximate solution operator $\mathcal{G}_{\mathcal{N}}$ by neural networks and train the network from a training dataset $\mathcal{T} = \{(v^{(1)}, u^{(1)}), (v^{(2)}, u^{(2)}), \cdots, (v^{(N)}, u^{(N)})\}$. Such neural networks $\mathcal{G}_{\mathcal{N}}$ are named by neural operators, or operator networks.

3 Related Work

Data Augmentation Although data augmentation techniques are widely used in image applications [12], little work is concerned about the continuous data arising in real-world physical systems such as PDEs. The closest work to ours is Brandstetter et al. [13], who designed a Lie point symmetry data augmentation (LPSDA) technique for neural PDE solvers. Lie point symmetry group of PDE includes time shift, space shift, Galilean boost, scaling, etc, which is very similar to the linearity and transformation in our PGDA method. Our PGDA is also applicable for elliptic PDEs defined on bounded domain with Dirichlet or Neumann boundary conditions, in which symmetries break because open neighborhoods cannot be defined for LPSDA. Luo et al. [14] designed a physics-directed data augmentation technique to reduce target-domain training data sampling complexity in transfer learning. However, a parametric first principle governing the domain shift need to be identified by the user first.

Equivariant Networks Developing neural networks that preserve symmetries has been a fundamental task in image recognition [15, 16, 17, 18], but little work on PDE solvers. The pioneering work of Wang et al. [10] incorporated Lie point symmetries into neural network solvers of Navier-Stokes equation and Heat equation. Their models are both theoretically and experimentally robust to distributional shift by symmetry group transformations and enjoy favorable sample complexity. However, the manual design of equivariant networks is not easy. An automatic symmetry discovery and the approximation for imperfectly symmetric dynamics are presented in their subsequent work [11, 19]. A convolutional layer that is equivariant to transformations from any specified Lie group with a surjective exponential map is constructed by Finzi et al. in [17]. Mattheakis et al. [20] embedded even/odd symmetry of a function and energy conservation into neural networks to solve differential equations.

Physics-informed Deep Learning Physics-informed deep learning integrating seamlessly data and mathematical physics models (for example, PDEs), has been one of the hotspots in current researches [3, 21]. Physics-informed neural networks (PINNs) proposed by Raissi et al. [22] are one of the outstanding neural PDE solvers in the last few years. Similar work [23, 24, 25, 26] solving PDEs with neural networks has been proposed in the same period. Other techniques include the neural operators [8, 27] presented in Section 4.1, and their variants [28, 29, 9, 30, 31]. There are researches about the so-called neural augmentation where a neural component is added to finite elements [32], multigrid solvers [7], and eikonal solvers [33].

4 Methodology

We first briefly introduce two neural operators with promising results in the literatures, namely DeepONet and FNO, then show how the physical properties of PDEs can be used as a data augmentation technique for neural operators.

4.1 Neural Operator

DeepONet The first neural operator, Deep Operator Network (DeepONet), was published in 2019 by Lu et al. [8] with subsequent theoretical and computational extensions in 2021 [34], and its original architecture was based on the universal approximation theorem of Chen & Chen [35]. Unlike function regression with finite-dimensional inputs and outputs, DeepONet aims to map infinite-dimensional functions (inputs) to infinite-dimensional functions (outputs). More importantly, DeepONet allows for solving PDEs with different boundary and initial conditions without the need for retraining the neural network.

A DeepONet has two sub-networks: a "trunk" network and a "branch" network. The trunk net takes the coordinates x as the input, and a p-dimensional vector $[t_1(x), t_2(x), \cdots, t_p(x)]$ as the output. The branch net takes the discretized function $\mathbf{v} = [v(x_1), v(x_2), \cdots, v(x_m)]$ as the input, and the same p-dimensional vector $[b_1(\mathbf{v}), b_2(\mathbf{v}), \cdots, b_p(\mathbf{v})]$ as the output. The general output of the DeepONet takes the form:

$$\mathcal{G}_{\mathcal{N}}(\mathbf{v})(x) = \sum_{k=1}^{p} b_k(\mathbf{v}) t_k(x) + b_0 \qquad (2)$$

where $b_0 \in \mathbb{R}$ is a bias. By sampling collocation points $\{y_1, y_2, \cdots, y_M\}$ on the solution function $u(x)$, we get the discrete loss function:

$$L = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} \left| u^{(i)}(y_j) - \mathcal{G}_{\mathcal{N}}(\mathbf{v}^{(i)})(y_j) \right|^2 \qquad (3)$$

where $(\mathbf{v}^{(i)}, y_j)$ are the $N \times M$ different pairs of branch and trunk inputs, $u^{(i)}(y_j)$ is the corresponding labelled outputs, and w_{ij} is the associated weight.

FNO The second neural operator, named Fourier Neural Operator (FNO), was published in 2020 by Li et al. [27], and it is based on parameterizing the integral kernel in the Fourier space. The feature of FNO is that the main network parameters are defined and learned in the Fourier space rather than the physical space. One difference between DeepONet and FNO is that FNO discretizes both the input function $v(x)$ and the output function $u(x)$ in

the same equispaced mesh.

First, the input function value $v(x)$ is lifted to a higher d_v dimensional representation $z_0(x)$ by

$$z_0(x) = P\big(v(x)\big), \tag{4}$$

where P is usually parameterized by a shallow fully connected neural network. Then following an iterative neural operator architecture $z_0 \mapsto z_1 \mapsto \cdots \mapsto z_L$ with

$$z_{t+1}(x) = \sigma\left(\int_\Omega \kappa_\phi(x-y)\cdot z_t(y)dy + W\cdot z_t(x)\right),$$
$$t = 0,1,\dots,L-1. \tag{5}$$

The convolution kernel function $\kappa_\phi(x-y)$ comes from the perspective of fundamental solutions of PDEs, and will be learned from data. If we parameterize κ_ϕ directly in Fourier space and using the Fast Fourier Transform (FFT) to efficiently compute the integral, we get the so called Fourier layer (see [27] Fig.2), and hence the name of FNO. Lastly, the hidden layer output z_L is projected back to the target dimension of $u(x)$ by another neural network Q:

$$\mathcal{G}_\mathcal{N}\big(v(x)\big) = Q\big(z_L(x)\big). \tag{6}$$

4.2 Physics-guided Data Augmentation

Here we show how to augment the training data of neural operators when applied to learn the solution operator of linear differential equations. Different from image data augmenting techniques like flip, rotation, crop, etc, we augment the training data with prior physical information of PDEs like linearity and translation.

For a given PDE (1) defined on a bounded domain Ω, we may assume $f = 0$ by simple solution transformation techniques. The solution operator $\mathcal{G}$ with $u(x) = \mathcal{G}(v)(x)$ is linear since the operators $\mathcal{L}$ and $\mathcal{B}$ are both linear. Assume we have obtained a training set $\mathcal{T} = \{\big(v^{(1)},u^{(1)}\big),\big(v^{(2)},u^{(2)}\big),\cdots,\big(v^{(N)},u^{(N)}\big)\}$, we can then augment the training set by the linear combination of sampling solutions. The new training data can be expressed as

$$\big(c_1v^{(i)} + c_2v^{(j)}, c_1u^{(i)} + c_2u^{(j)}\big), \quad \forall c_1,c_2 \in \mathbb{R}$$
$$\text{and } i,j = 1,\dots,N. \tag{7}$$

Another technique is to use the translation invariance of the PDE. We first obtain a sampled training data with constant inputs $v^{(0)} = 1$ on the domain Ω, and the corresponding solution $u^{(0)} = \mathcal{G}\big(v^{(0)}\big)$. Then we have the new training data

$$\big(v^{(i)} + v^{(0)}, u^{(i)} + u^{(0)}\big), \quad \forall i = 1,\dots,N. \tag{8}$$

The augmentation technique (7) and (8) can be combined to produce more training data without additional cost to generate data

$$\big(c_0v^{(0)} + c_1v^{(i)} + c_2v^{(j)}, c_0u^{(0)} + c_1u^{(i)} + c_2u^{(j)}\big),$$
$$\forall c_0,c_1,c_2 \in \mathbb{R}; i,j = 1,\dots,N \tag{9}$$

For the case when $\mathcal{L}$, $\mathcal{B}$ or the operator $\mathcal{G}$ are nonlinear, then the universal linearity and transformation augmentation techniques are no longer valid. The augmentations can be tailored to the physical properties of the PDE's solution. For example, for the Darcy flow problem $-\nabla\cdot\big(a(x)\nabla u(x)\big) = f(x)$ in porous media, we consider the nonlinear solution operator $\mathcal{G}: a(x) \to u(x)$, mapping from the permeability function $a(x)$ to the pressure function$u(x)$. If $\big(a(x),u(x)\big)$ is a pair of training data, then we can augment it by $\left(c_0a(x), \frac{1}{c_0}u(x)\right)$ with constant $c_0 \in \mathbb{R}, c_0 \neq 0$.

5 Experiments

We test our PGDA method on the solution operator learning of different linear differential equations, from a simple 1D linear ODE system to the 2D PDE dynamics and singular perturbation problems. We compare the testing error for neural operator models trained with/without PGDA and demonstrate that adding PGDA can improve generalization performance effectively.

Evaluation Metrics. The training and testing errors are measured by the following MSE error:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\big\|\mathcal{G}_\mathcal{N}\big(v^{(i)}\big) - u^{(i)}\big\|^2, \tag{10}$$

where $\|\cdot\|$ denotes the discrete L^2 norm in the output space, $v^{(i)}$ is the input function, $u^{(i)}$ is the target and $\mathcal{G}_\mathcal{N}\big(v^{(i)}\big)$ is the DeepONet/FNO output.

Experimental Setup. We train DeepONet and FNO models with/without PGDA on different linear differential equations. All models are trained by Adam optimizer with an initial learning rate $\alpha = 0.001$ that is halved every 100 epochs. The training instances before augmentation are $N = 1{,}000$ and testing instances are 200. The DeepONet's trunk net and brunch net are 4 layers FNN with $p = 128$ and $m = 32$. The FNO has 4 Fourier layers with $k_{max} = 16$ and $d_v = 32$. The linear coefficients c_0, c_1, c_2 in the augmentation technique (9) are randomly sampled constants.

Data Generalization. The input function $v(x)$ is generated from the Gaussian Random Field (GRF)

$$v \sim \mathcal{G}\big(A, k_l(x_1,x_2)\big),$$

where the constant A is the mean value of the sample function, and the covariance kernel $k_l(x_1,x_2) = \exp\left(-\frac{\|x_1-x_2\|^2}{2l^2}\right)$ is the radial-basis function kernel with a length-scale parameter $l > 0$. We train DeepONet and FNO on one pair (A_0,l_0) and test them both on the same pair (A_0,l_0) and distributional shifted pairs $(A',l') \neq (A_0,l_0)$ with/without PGDA on different linear differential equations.

5.1 A Simple 1D Dynamic System

A 1D dynamic system is described by

$$\frac{du(x)}{dx} = g(u(x), v(x), x), \quad x \in [0,1], \tag{11}$$

with an initial condition $u(0) = 0$. Our goal is to predict $u(x)$ over the whole domain $[0,1]$ for any given $v(x)$. For simplicity we set $g(s(x), u(x), x) = v(x)$, then it is equivalent to learning the antiderivative operator $\mathcal{G}: v(x) \to u(x) = \int_0^x v(\tau)d\tau$.

We train DeepONet and FNO on the training data for $(A, l) = (0,0.2)$ with and without PGDA, then we test the model on distribution shift data with different pairs (A, l), see Table I. Both the DeepONet and FNO trained with PGDA can generalize very well on different test data, and the FNO with PGDA shows best performance.

5.2 Poisson Equation

The Poisson equation is an elliptic partial differential equation of broad utility in theoretical physics. We consider the Poisson equation on the domain $\Omega = [0,1] \times [0,1]$:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x,y), \quad (x,y) \in \Omega, \tag{12}$$

with the Dirichlet boundary condition $u|_{\partial\Omega} = 0$. We want to learn the mapping $\mathcal{G}: f(x,y) \to u(x,y)$. Similar results as described in Section 5.1 can be obtained from Table I. An instance of the Poisson equation with a referenced solution and the learned neural operator solution are plotted in Figure 2.

Table I Test of PGDA on different tasks for improving generalization. MSE errors on different test sets are reported.

Task	Test data	***Without*** PGDA		***With*** PGDA	
		DeepONet	FNO	DeepONet	FNO
Section 5.1	$(A, l) = (0,0.2)$	4.65×10^{-4}	1.68×10^{-5}	1.77×10^{-3}	$\mathbf{9.44\times10^{-6}}$
	$(A, l) = (0,2)$	6.32×10^{-4}	1.45×10^{-4}	9.09×10^{-4}	$\mathbf{1.27\times10^{-5}}$
	$(A, l) = (10,0.2)$	3.67×10^{-3}	3.85×10^{-1}	2.06×10^{-4}	$\mathbf{3.33\times10^{-5}}$
	$(A, l) = (10,2)$	3.32×10^{-3}	3.86×10^{-1}	5.82×10^{-5}	$\mathbf{1.42\times10^{-5}}$
Section 5.2	$(A, l) = (0,0.2)$	1.52×10^{-4}	4.79×10^{-6}	1.75×10^{-4}	$\mathbf{4.04\times10^{-6}}$
	$(A, l) = (0,2)$	4.19×10^{-5}	7.01×10^{-6}	4.03×10^{-5}	$\mathbf{1.08\times10^{-6}}$
	$(A, l) = (10,0.2)$	3.23×10^{-3}	2.29×10^{-1}	1.03×10^{-3}	$\mathbf{1.99\times10^{-5}}$
	$(A, l) = (10,2)$	3.03×10^{-3}	2.18×10^{-1}	9.05×10^{-4}	$\mathbf{1.11\times10^{-5}}$
Section 5.3	$(A, l) = (0,0.2)$	4.72×10^{-3}	4.01×10^{-5}	3.99×10^{-3}	$\mathbf{1.21\times10^{-5}}$
	$(A, l) = (0,2)$	7.49×10^{-3}	2.81×10^{-4}	2.42×10^{-3}	$\mathbf{1.68\times10^{-5}}$
	$(A, l) = (10,0.2)$	1.75×10^{-1}	4.29×10^{-1}	7.41×10^{-3}	$\mathbf{4.76\times10^{-5}}$
	$(A, l) = (10,2)$	1.74×10^{-1}	4.28×10^{-1}	7.30×10^{-5}	$\mathbf{5.07\times10^{-5}}$

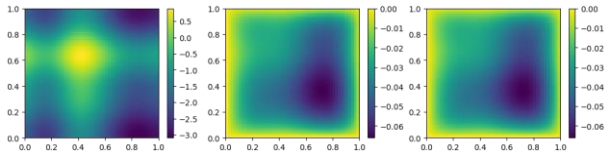

Figure 2 The neural operator learning of the Poisson solution operator $\mathcal{G}: f(x,y) \mapsto u(x,y)$. (**Left**) An instance of input function $f(x,y)$. (**Middle**) Reference solution $(\mathcal{G}f)(x,y)$. (**Right**) Learned neural operator $(\mathcal{G}_N f)(x,y)$ with PGDA.

5.3 Singularly Perturbed Advection-Diffusion Equation

Singularly perturbed problems have been successfully applied to many fields including gas dynamics, chemical reaction, fluid mechanics, elasticity, etc. To find the solution is a hot and difficult problem because it contains a very small parameter $0 < \epsilon \ll 1$. We consider the second-order linear singularly perturbed advection-diffusion equation

$$-\epsilon u''(x) + u'(x) = f(x), \quad x \in (0,1), \tag{13}$$

with Dirichlet boundary conditions $u(0) = u(1) = 0$. We want to learn the mapping $\mathcal{G}: f(x) \to u(x)$. As seen in Table I, MSE errors of DeepONet and FNO without PGDA increase for the appearance of this small parameter ϵ, but our PGDA method still generalize well for different test data in the small parameter regime.

6 Conclusion and Future Work

In this paper we developed a physics-guided data augmentation (PGDA) technique for the neural operator model training in the application of differential equations to improve generalization. Training data is augmented based on the physical properties of differential equations such as linearity and translation. In the case with out-of-distribution test data, our PGDA methods generalize significantly better than models trained without data augmentation. Future work includes continuing to find other physical-properties for data augmentation and active sampling strategies to accelerate training process.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.62106103), Fundamental

Research Funds for the Central Universities (No.ILA22023, No.90YAH20131), 173 Program Technical Field Fund (No.2021-JCJQ-JJ-0018).

References

[1] Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25 (2012)

[2] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature, 521(7553), 436–444 (2015)

[3] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nature Reviews Physics, 3(6), 422–440 (2021)

[4] Ames, W. F.: Numerical methods for partial differential equations. Academic Press (2014)

[5] Bar-Sinai, Y., Hoyer, S., Hickey, J., Brenner, M. P.: Learning data-driven discretizations for partial differential equations. Proceedings of the National Academy of Sciences, 116(31), 15344–15349 (2019)

[6] Thuerey, N., Holl, P., Mueller, M., Schnell, P., Trost, F., Um, K.: Physics-based deep learning. arXiv preprint arXiv:2109.05237 (2021)

[7] Greenfeld, D., Galun, M., Basri, R., Yavneh, I., Kimmel, R.: Learning to optimize multigrid PDE solvers. In International Conference on Machine Learning, pp. 2415–2423, PMLR (2019, May)

[8] Lu, L., Jin, P., Karniadakis, G. E.: DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:1910.03193 (2019)

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

[10] Wang, R., Walters, R., Yu, R.: Incorporating symmetry into deep dynamics models for improved generalization. In International Conference on Learning Representations (ICLR) (2021, January)

[11] Dehmamy, N., Walters, R., Liu, Y., Wang, D., Yu, R.: Automatic symmetry discovery with Lie algebra convolutional network. Advances in Neural Information Processing Systems, 34 (2021)

[12] Shorten, C., Khoshgoftaar, T. M.: A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 1–48 (2019)

[13] Brandstetter, J., Welling, M., Worrall, D. E.: Lie point symmetry data augmentation for neural PDE solvers. arXiv preprint arXiv:2202.07643 (2022)

[14] Luo, W., Yan, Z., Song, Q., Tan, R.: PhyAug: Physics-directed data augmentation for deep sensing model transfer in cyber-physical systems. In Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021), pp. 31–46 (2021, May)

[15] Cohen, T., Welling, M.: Group equivariant convolutional networks. In International Conference on Machine Learning, pp. 2990–2999, PMLR (2016, June)

[16] Cohen, T., Weiler, M., Kicanaoglu, B., Welling, M.: Gauge equivariant convolutional networks and the icosahedral CNN. In International Conference on Machine Learning, pp. 1321–1330, PMLR (2019, May)

[17] Finzi, M., Stanton, S., Izmailov, P., Wilson, A. G.: Generalizing convolutional neural networks for equivariance to Lie groups on arbitrary continuous data. In International Conference on Machine Learning, pp. 3165–3176, PMLR (2020, November)

[18] Weiler, M., Cesa, G. General E(2)-equivariant steerable CNNs. Advances in Neural Information Processing Systems, 32 (2019).

[19] Wang, R., Walters, R., Yu, R.: Approximately equivariant networks for imperfectly symmetric dynamics. arXiv preprint arXiv:2201.11969 (2022)

[20] Mattheakis, M., Protopapas, P., Sondak, D., Di Giovanni, M., Kaxiras, E.: Physical symmetries embedded in neural networks. arXiv preprint arXiv:1904.08991 (2019)

[21] Markidis, S.: The old and the new: Can physics-informed deep-learning replace traditional linear solvers? Frontiers in big Data, 92 (2021)

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

[23] Sirignano, J., Spiliopoulos, K.: DGM: A deep learning algorithm for solving partial differential equations. Journal of Computational Physics, 375, 1339–1364 (2018)

[24] E, W., Yu, B.: The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. Communications in Mathematics and Statistics, 1(6), 1–12 (2018)

[25] Khoo, Y., Lu, J., Ying, L.: Solving parametric PDE problems with artificial neural networks. European Journal of Applied Mathematics, 32(3), 421–435 (2021).

[26] Raissi, M.: Deep hidden physics models: Deep learning of nonlinear partial differential equations. The Journal of Machine Learning Research, 19(1), 932–955 (2018)

[27] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)

[28] Wang, S., Wang, H., Perdikaris, P.: Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Science Advances, 7(40), eabi8605 (2021)

[29] Wang, S., Bhouri, M. A., Perdikaris, P.: Fast PDE-constrained optimization via self-supervised operator learning. arXiv preprint arXiv:2110.13297 (2021)

[30] Jin, P., Meng, S., Lu, L.: MIONet: Learning multiple-input operators via tensor product. arXiv preprint arXiv:2202.06137 (2022)

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

[32] Hsieh, J. T., Zhao, S., Eismann, S., Mirabella, L., Ermon, S.: Learning neural PDE solvers with convergence guarantees. arXiv preprint arXiv:1906.01200 (2019)

[33] Lichtenstein, M., Pai, G., Kimmel, R.: Deep eikonal solvers. In International Conference on Scale Space and Variational Methods in Computer Vision, pp. 38–50, Springer, Cham (2019, June).

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

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