PDENNEval: A Comprehensive Evaluation of Neural Network Methods for Solving PDEs[*]

Ping Wei, **Menghan Liu**, **Jianhuan Cen**, **Ziyang Zhou**, **Liao Chen** and **Qingsong Zou**[†]
School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
Guangdong Province Key Laboratory of Computational Science, Guangzhou, China
{weip7, liumh59, cenjh3, zhouzy36, chenliao}@mail2.sysu.edu.cn, mcszqs@mail.sysu.edu.cn

Abstract

The rapid development of neural network (NN) methods for solving partial differential equations (PDEs) has created an urgent need for evaluation and comparison of these methods. In this study, we propose PDENNEval, a comprehensive and systematic evaluation of 12 NN methods for PDEs. These methods are classified into function learning type and operator learning type based on their different mathematical foundations. The evaluation is implemented using a diverse dataset comprising 19 distinct PDE problems selected from various scientific fields such as fluid, materials, finance, and electromagnetic. Several evaluation results are reported, aiming to provide guidance for further research in this field. Our code and data are publicly available at https://github.com/Sysuzqs/PDENNEval.

1 Introduction

Owing to their exceptional approximation capabilities [Hornik, 1991; Barron, 1994] and multi-level feature extraction abilities [LeCun *et al.*, 2015], neural networks (NN) based methods have been widely applied and have achieved great success in various disciplines, including computer vision [Krizhevsky *et al.*, 2012; He *et al.*, 2016; Dosovitskiy *et al.*, 2020] and natural language processing ([Vaswani *et al.*, 2017; Devlin *et al.*, 2018; Brown *et al.*, 2020]). Recently, NN methods have excelled in solving partial differential equations (PDEs), yielding significant breakthroughs in this field, e.g. [E *et al.*, 2017; E and Yu, 2018; Raissi *et al.*, 2019; Li *et al.*, 2020; Lu *et al.*, 2021a]. Compared to traditional numerical PDEs methods like finite element methods, NN methods exhibit unique advantages. One notable advantage is that they are mesh-free, making them immune to the *curse of dimensionality*. In addition, NN methods can learn patterns and features from data which might be governed by multiple types of equations[Raissi and Karniadakis, 2018]. This capability enables a single solver to handle various types of equations, offering a more versatile approach for solving PDEs.

The widespread development of NN methods for solving PDEs generates a pressing need for the assessment and comparative analysis of these techniques. In 2021, Lu and his colleagues compare the Physics-Informed Neural Networks (PINN) to a standard finite element method across various problems, including Poisson equation, Burgers' equation, Volterra integral differential equation, Lorenz system and Diffusion-Reaction systems in [Lu *et al.*, 2021b]. In 2022, three benchmarks of NN methods for PDEs are published. First, Lu and his colleagues[Lu *et al.*, 2022] present the relative performance of Deep Operator Network (DeepONet) and Fourier Neural Operator (FNO) by using 16 different benchmarks covering Burgers' equation, Darcy problem, Advection equation, Compressible Euler equations, electro-convection problem and Navier-Stokes equation, etc. Secondly, a side-by-side analysis, called PDEArena, is presented in [Gupta and Brandstetter, 2022] to show the performance of NN methods FNO, ResNet, and U-Net on Shallow-Water equations and Navier-Stokes equations. Thirdly, a versatile benchmark suite[Takamoto *et al.*, 2022b], called PDEBench, is proposed to compare NN methods FNO, U-Net, PINN, using datasets derived from 11 hydromechanical field PDEs, including Advection, Navier-Stokes equations, etc. In 2023, a benchmark[Hao *et al.*, 2023], called PINNacle, is designed to assess the performance of 10 PINN variants, encompassed tasks derived from over 20 different PDEs spanning various domains, such as heat conduction, fluid dynamics, biology, and electromagnetic.

Previous research primarily benchmarks some popular methods such as DeepONet, FNO, U-Net, and PINN. However, a comprehensive evaluation of some other important NN methods is lacking. Additionally, different types of NN methods are often compared in the same setting without being classified and evaluated based on their distinct mathematical foundations. On the other hand, despite the selection of various types of PDE problems in previous work, some crucial equations, such as phase-field equations, have not been included in the evaluation tasks. Furthermore, challenging problems like PDEs with singular solutions have also been omitted. In summary, the development of the evaluation and comparison works for NN methods have not kept pace with the advancement of NN methods themselves.

[*]An extendede version of this paper(with the Appendix included) can be find in https://github.com/Sysuzqs/PDENNEval.

[†]Corresponding author.

In this work, we propose a comprehensive and systematic evaluation of various NN methods for solving diversified PDE problems, considering accuracy, efficiency and robustness.

Our evaluations are integrated into a benchmarks suite called PDENNEval, with some key features listed below.

- We provide a code reference that integrates 12 advanced NN methods, including six function learning-based NN methods: the Deep Ritz Method(DRM) [E and Yu, 2018], PINN [Raissi *et al.*, 2019], Weak Adversarial Network (WAN) [Zang *et al.*, 2020], Derivative-Free Loss Method (DFLM)[Han *et al.*, 2020], Random Feature Method (RFM) [Chen *et al.*, 2022], Deep Finite Volume Method (DFVM) [Cen and Zou, 2023], and six operator learning-based NN methods: U-Net[Ronneberger *et al.*, 2015], Message Passing Neural PDE Solvers (MPNN)[Brandstetter *et al.*, 2022], FNO [Li *et al.*, 2020], DeepONet[Lu *et al.*, 2021a], Physics-Informed Neural Operator (PINO)[Li *et al.*, 2021], U-shaped Neural Operators (U-NO)[Rahman *et al.*, 2022]. We systematically evaluate the performance of these two categories in different setting.

- We carefully select 19 representative PDE problems from various fields such as fluid dynamics, materials science, finance, and electromagnetism to evaluate the performance of the aforementioned NN methods. These equations encompass some core mathematical challenges, including nonlinearity, singularity, high dimensionality and complex geometry. It's may worth mentioning that it is the first time NN methods have been evaluated by using a PDE with a singular solution. More detailed descriptions for all PDEs are provided in Appendix A.

- We design a diverse and high-fidelity dataset, generated using high-precision conventional scientific-computation methods. For instance, to generate the data for the phase-field equations, spatial discretization is accomplished using the spectral method, and time discretization is performed using the fourth-order Runge-Kutta method.

- We select a set of metrics that are better suited for each type of NN methods, including L_2 norm relative error and maximum error, to assess accuracy, Lipschitz constant to evaluate robustness, and efficiency metrics encompassing training time, inference time, and convergence time.

The rest of the paper is organized as below. In Section 2, we outline the 12 representative NN methods and present 19 distinct PDE tasks which will be evaluated in PDENNEval. In Section 3, we present an overview of the datasets and metrics used in PDENNEval. In Section 4, we illustrate a standardized testing procedure, and present the results of the pertinent tests. Some brief concluding remarks are given in Section 5.

2 An Overview of Selected PDEs and NN Methods

In this section, we briefly introduce the PDE problems and NN methods integrated into the PDENNEval.

2.1 Selected PDE Tasks

A well-posed PDE system often involves three fundamental components: the equation itself, boundary conditions, and initial values. Let $\Omega \subset \mathbb{R}^n$ be the spatial domain with boundary $\partial\Omega$, and let $[0,T]$ be the time interval. A general PDE system can be represented as:

$$\mathcal{F}(u; \mathbf{x}, t; \lambda) = \mathbf{f}(\mathbf{x}, t), \qquad (\mathbf{x}, t) \in \Omega \times [0, T], \tag{1}$$

where $\mathcal{F}$ represents a differential operator governing the behavior of the unknown function u, λ is an optional vector of parameters, appearing often as constant or variable coefficients of the differential operator, and $\mathbf{f}(\mathbf{x}, t)$ is a vector-valued function. This equation encapsulates the evolution of the system over the spatiotemporal domain. Boundary conditions are often expressed as:

$$\mathcal{B}(u; \mathbf{x}, t) = \mathbf{b}(\mathbf{x}, t), \qquad \mathbf{x} \in \partial\Omega, \tag{2}$$

where $\mathbf{b}$ is a given vector-valued function defined on $\partial\Omega$. The boundary conditions $\mathcal{B}$ define how the solution behaves or is constrained at the boundary of the domain. At $t = 0$, the system's state is often specified by initial values:

$$\mathcal{I}(u; (\mathbf{x}, t)|_{t=0}) = \mathbf{i}(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{3}$$

where $\mathbf{i}$ is a given vector-valued function defined only on Ω.

Usually, we call it as a *forward* problem when seeking a function u to satisfy (1), (2) and (3), given the information $\Omega, \lambda, \mathbf{f}, \mathbf{b}, \mathbf{i}$. Conversely, we call it as an *inverse* problem if we aim to seek parameters λ, or the shape of Ω, or initial conditions, or other essential information about the system based on some observed data or outcomes. In this bench, since an inverse problem might be solved by training a NN using the same loss function as its corresponding forward problem, we will focus only on solving forward problems and encourage subsequent researchers to benchmark their respective inverse problems.

In our study, we aim to benchmark 19 forward problems of PDEs, representing pivotal mathematical models in physics and engineering that span various fields, including fluid dynamics, materials science, finance, and electromagnetism. Within the realm of fluid dynamics, our benchmark incorporates 11 equations: 1D Advection, 1D and 2D Burgers', 1D Diffusion-Reaction, 1D Diffusion-Sorption, 2D Darcy Flow, 2D Shallow-Water, 1D, 2D, and 3D Navier-Stokes, and 3D Euler equations. These equations model gas and liquid flow phenomena, finding applications across aerospace, weather prediction, oceanography, chemistry, and hydraulic engineering fields. Notably, while nine fluid-related PDEs have been evaluated in PDEBench, they were restricted to only three methods, whereas our assessment extends to twelve methods in total. In materials science, our selection includes the 1D and 2D Allen-Cahn equations and the 1D Cahn-Hilliard equation, employed in understanding material phase transitions, crystal growth, and surface evolution phenomena. Regarding finance, we encompass the 2D Black-Scholes-Barenblatt equation, crucial in the pricing of European options. In the domain of electromagnetism, our benchmark involves the Poisson equation and 3D Maxwell's equations, which model the behaviors of electromagnetic field and serve as fundamental elements in the realms of communication technology, electromagnetic device design, and optical systems. Specifically for the Poisson equation, we construct three distinct cases.

We select these PDEs to ensure that the benchmark introduces a wide range of mathematical challenges, including

- **Nonlinear behavior**: Numerous PDEs showcase nonlinear or even chaotic dynamics, wherein minor alterations in initial conditions result in significant divergence within the outcomes. Addressing these nonlinearities illuminates a method's capacity to capture intricate and sensitive dependencies within the system's behavior. In particular, our set of equations includes primarily nonlinear equations, such as the Navier-Stokes equations, phase field equations, and the Black-Scholes-Barenblatt equation.
- **High dimensionality**: Various fields, including materials, finance, and quantum mechanics, present high-dimensional PDEs. To evaluate the capability of NN methods in handling high-dimensional PDEs, we utilize high-dimensional ($d > 3$) Poisson equations to compare the performance of function learning-based NN methods.
- **Singularity**: Singularities in partial differential equations represent unconventional solution properties, often marked by unboundedness, discontinuities, or lack of differentiability at specific points. In our PDENNEval, we intentionally create a Poisson equation with a singular solution for assessment purposes.
- **Complex geometry**: PDEs of practical significance often extend beyond regular domains, underscoring the importance of evaluating NN methods' capability to handle irregular domain scenarios. In our study, focusing on the Poisson equation, we established a specific case involving the resolution of the equation over an L-shaped domain.

2.2 Selected NN Methods

An NN method heavily relies on its definition of the loss function $\mathcal{L}(\theta)$ which depends on θ, the trainable parameters of the NN. Generally, a loss $\mathcal{L}(\theta)$ for PDEs consolidates four essential components into a unified expression as follows:

$$\begin{aligned}\mathcal{L}(\theta) = \omega_{\text{PDE}}\, \mathcal{L}_{\text{PDE}}(\theta) + \omega_{\text{BC}}\, \mathcal{L}_{\text{BC}}(\theta) \\ + \omega_{\text{IC}}\, \mathcal{L}_{\text{IC}}(\theta) + \omega_{\text{Data}}\, \mathcal{L}_{\text{Data}}(\theta),\end{aligned} \tag{4}$$

where $\omega_{\text{PDE}}, \omega_{\text{BC}}, \omega_{\text{IC}}, \omega_{\text{Data}}$ are the weights of different loss terms, and a breakdown of each loss term is explained as below:

- **PDE constraint loss ($\mathcal{L}_{\text{PDE}}$)** enforces the PDE constraints within the domain Ω, defined as

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_1} \sum_{(\mathbf{x},t)\in\mathcal{S}_{\text{PDE}}} \|\widehat{\mathcal{F}}(u_\theta, \mathbf{f}; \mathbf{x}, t; \lambda)\|^2, \tag{5}$$

where $\widehat{\mathcal{F}}$ is a variant of the original equations (1), including its strong, weak, stochastic forms; $\mathcal{S}_{\text{PDE}}$ is the set of training points in the interior of $\Omega \times [0, T]$, and $N_1 = \#\mathcal{S}_{\text{PDE}}$.

- **Boundary condition loss ($\mathcal{L}_{\text{BC}}$)** ensures adherence to specified boundary conditions at $\partial\Omega$, defined as

$$\mathcal{L}_{\text{BC}}(\theta) = \frac{1}{N_2} \sum_{(\mathbf{x},t)\in\mathcal{S}_{\text{BC}}} \|\left(\mathcal{B}(u_\theta) - \mathbf{b}\right)(\mathbf{x}, t)\|^2, \tag{6}$$

where $\mathcal{S}_{\text{BC}}$ is the set of training points sampled from the boundary $\partial\Omega \times [0, T]$, and $N_2 = \#\mathcal{S}_{\text{BC}}$.

- **Initial condition loss ($\mathcal{L}_{\text{IC}}$)** enforces accuracy at the initial time $t = 0$, defined as

$$\mathcal{L}_{\text{IC}}(\theta) = \frac{1}{N_3} \sum_{\mathbf{x}\in\mathcal{S}_{\text{IC}}} \|\mathcal{I}(u_\theta(\mathbf{x}, 0)) - \mathbf{i}(\mathbf{x})\|^2, \tag{7}$$

where $\mathcal{S}_{IC}$ is the set of training points sampled from the Ω, and $N_3 = \#\mathcal{S}_{\text{IC}}$.

- **Data loss ($\mathcal{L}_{\text{Data}}$)** measures the discrepancy between predicted and observed data at designated points, defined as

$$\mathcal{L}_{\text{Data}}(\theta) = \frac{1}{N_4} \sum_{(\mathbf{x},t)\in\mathcal{S}_{\text{Data}}} \|u_\theta(\mathbf{x}, t) - u(\mathbf{x}, t)\|^2, \tag{8}$$

where $\mathcal{S}_{\text{Data}}$ is the set of training points at which the value of the true solution u is known and $N_4 = \#\mathcal{S}_{\text{Data}}$.

In general, physics-driven methods typically encompass only the first three components of the loss function, i.e., $\omega_{\text{Data}} = 0$, while data-driven methods exclude the PDE constraint loss. However, in practice, certain physics-driven approaches encompass all loss components, blurring the distinction between data-driven and physics-driven paradigms. Hence, categorizing methods strictly into physics-driven and data-driven techniques becomes inadequate, given the strong coupling between these methodologies. The demarcation between these two approaches tends to be ambiguous.

In PDENNEval, we attempt to categorize NN methods for solving PDEs into two principal categories distinguished by their mathematical foundations. The first category employs elaborated NN to fit the solution function of the equation, rooted in the universal approximation theory[Hornik, 1991; Barron, 1994; He *et al.*, 2022]. The second category utilizes NN to fit solution operators for a spectrum of PDEs, based on the universal approximation of linear or nonlinear operators by NN[Lu *et al.*, 2021a]. Some details of these two methods are described as below.

Function Learning Based NN Methods

NN methodologies for function learning largely fall under the domain of physics-driven methods, resembling traditional approaches in their goal of approximating solutions to PDEs. While traditional numerical methods often utilize piece-wise polynomials, NN methods leverage networks for this purpose. Typically, these methods involve training the networks using the four aforementioned loss components. However, similar to a traditional numerical method which solves a PDE only using the equation itself, the boundary and initial conditions, a function learning type NN method often does not need to use exceptional data to train the NN. That is, ω_{Data} is often set to 0. Certainly, data might contribute to the boundary and initial value losses, but they are typically treated separately from the data loss component.

Function learning based methods differ mainly in their treatment of the PDE constraint loss item. Strong form-based methods like PINN and RFM integrate the PDE constraint loss utilizing the *residual* $\widehat{\mathcal{F}} = \mathcal{F} - f$. PINN and RFM both employ strong form-based loss functions, yet their optimization processes differ. PINN utilizes deep learning optimizers, while RFM, employing linear algebraic equations, fixes the parameters of the preceding layers to compute the final layer's parameters. Weak form-based methods encompass DRM, WAN, DFLM, and DFVM. DRM is rooted in the variational format, emphasizing the principle of minimal potential energy. WAN adopts the virtual-work-principle form, focusing on finding weak solutions to the equations. DFLM tailors its loss to match the equivalent stochastic-differential form of the original PDE. DFVM aligns its loss with the local-conservation form of the PDE.

Operator Learning Based NN Methods

Operator learning-based methods tackle solving a set of PDEs by learning some operators, categorized into autoregressive and non-autoregressive approaches. Autoregressive methods focus on learning mappings from solutions at preceding time steps to subsequent time steps, including FNO, MPNN, U-Net, and U-NO. Non-autoregressive methods focus on learning mappings from auxiliary functions to solution functions, including DeepONet and PINO.

Autoregressive methods learn mappings from preceding time-step solutions to subsequent time-step solutions. These methods fundamentally simulate an image-to-image mapping, forecasting the future state based on the current state, which can be described as below :

$$\mathcal{G} : u(\mathbf{x}, t)|_{t=t_0} \mapsto u(\mathbf{x}, t)|_{t=t_1}. \tag{9}$$

These methods involve formulating the loss function to minimize the difference between the predicted solution at the next time step and the actual solution. This enables the network to learn the temporal evolution of the system. For instance, FNO focuses on parameterizing the integral kernel directly in Fourier space, enabling an expressive and efficient architecture for learning solution operators. MPNN operates on graph data by iteratively generating messages, aggregating neighbor information, and updating node representations. U-Net adapts an autoregressive approach for solving time-dependent PDEs. Its architecture predicts subsequent time step solutions based on preceding sequences of solutions, making it adept at capturing both local and global patterns for progressive solutions. U-NO employs a U-shaped architecture, mitigating memory constraints while efficiently mapping functions across different domains. By handling encoding and decoding with skip connections, it offers an effective solution for complex neural operator modeling.

Non-autoregressive methods involve learning an operator which maps the known information $(\lambda, \mathbf{f}, \mathbf{b}, \mathbf{i})$ to solution

PDE	Dataset	PDE	Dataset
Advection	Ad1	Diffusion-Reaction	DR1
Burgers 1D	Bu1	Diffusion-Sorption	DS1
Allen-Cahn 1D	AC1	Cahn-Hilliard	CH1
Comp Navier-Stokes 1D	NS1	Comp Navier-Stokes 2D	NS2
Allen-Cahn 2D	AC2	Burgers 2D	Bu2
Darcy Flow	DF2	Shallow-Water	SW2
Black-Scholes-Barenblatt	BS2	Comp Navier-Stokes 3D	NS3
Maxwell's	Ma3	Euler	Eu3

Table 1: The name of datasets corresponding to PDEs

functions $u \in \mathcal{U}$, described as

$$\mathcal{G} : (\lambda, \mathbf{f}, \mathbf{b}, \mathbf{i}) \mapsto u. \tag{10}$$

In other words, these methods aim to learn mappings from right-hand side functions, boundary conditions, or other auxiliary information to the solution function. The loss function formulation for such mappings varies based on specific model designs and target functions, and may or may not involve the explicit PDE constraint loss term. For instance, DeepONet utilizes separate feed-forward neural networks (FNNs) to encode input functions and query locations, effectively mapping auxiliary functions directly to solutions. PINO blends data and physics information to learn resolution-invariant operators in Fourier space. It efficiently handles temporal PDEs by combining data-driven and physics-driven methodologies for accurate solutions across various resolutions and dimensions.

3 An Overview of Datasets and Metrics

In this section, we present two fundamental components of PDENNEval: the datasets and the metrics.

3.1 Datasets

The PDENNEval datasets consist of 16 data files, each representing a distinct PDE, excluding three instances of the Poisson equation which are solved using only the first two loss components in (4) (i.e. $\omega_{\text{IC}} = \omega_{\text{Data}} = 0$). These 16 data files are named using the following abbreviations of PDEs.

In each data file, we store multiple samples, each of which is a solution of the related PDE. All solution samples are obtained by using traditional numerical methods to solve the PDE with different initial functions, sampled in a certain function space according to a certain distribution. Each sample contains the spatiotemporal coordinates of the underlying mesh and the values of the solution at those locations. Listed in Table 2 is a summary of all 16 data files. In this table, N_d is the number of spatial dimensions of the equation, and N_S is the size of the data file. Moreover, we use N_{sp}, N_t, N_{sa} to denote the spatial resolution, the temporal resolution of the underlying mesh, and the number of samples, respectively. Note that, since the Darcy flow equation is time-independent, we have not provided the number N_t for the dataset DF2.

Some details about the traditional numerical methods for generating datasets are reported below. First, the following four datasets: AC1, AC2, CH1, and Bu2 are generated using the Matlab R2022a Software. More precisely, the samples in AC1, AC2 and CH1 are obtained by discretizing the spatial

Dataset	N_d	N_S	N_{sp}	N_t	N_{sa}
Ad1	1	7.7 G	1024	201	10000
DR1	1	3.9 G	1024	101	10000
Bu1	1	7.7 G	1024	201	10000
DS1	1	4.0 G	1024	101	10000
AC1	1	3.9 G	1024	101	10000
CH1	1	3.9 G	1024	101	10000
NS1	1	12 G	1024	101	10000
Bu2	2	13 G	128×128	101	1000
NS2	2	52 G	128×128	21	10000
DF2	2	1.3 G	128×128	-	10000
SW2	2	6.2 G	128×128	101	1000
AC2	2	6.2 G	128×128	101	1000
BS2	2	6.2 G	128×128	101	1000
Ma3	3	5.9 G	$32 \times 32 \times 32$	8	1000
Eu3	3	83 G	$128 \times 128 \times 128$	21	100
NS3	3	83 G	$128 \times 128 \times 128$	21	100

Table 2: Summary of datasets

variable with a spectral method and discretizing the temporal variable with a fourth-order Runge-Kutta scheme. The samples in the dataset Bu2 are obtained by discretizing the spatial first-order derivatives with a first-order upwind scheme and discretizing the spatial second-order derivatives with a central difference scheme. Secondly, the datasets Ma3, BS2, Eu3 are generated using Python. To be more specific, the samples in Ma3 are obtained by discretizing the 3D Maxwell equation using the finite-difference time-domain method (FDTD, [Kunz and Luebbers, 1993]), and the resulting discrete algebraic system is solved with the leapfrog method. The BS2 dataset is sampled from some exact solutions of the Black-Scholes-Barenblatt equation given in [Raissi, 2024]. The samples in Eu3 are obtained by solving the Euler equation utilizing the second-order HLLC scheme [Toro *et al.*, 1994] for the inviscid part, the MUSCL schemes [Van Leer, 1979] for its inviscid counterpart, and a central difference scheme for the viscous part. The remaining nine datasets are related to PDEs in fluid dynamics, their samples are generated using the numerical methods which been used to generate the dataset in the PDEBench [Takamoto *et al.*, 2022a].

All datasets are stored in HDF5 format, which is convenient to use and share and widely supported in a host of programs including Python, R and Matlab. The data and usage example can be found in https://github.com/Sysuzqs/PDENN Eval. More details of datasets can be found in Appendix C.

3.2 Metrics

In this section, we briefly introduce the metrics used in PDENNEval. To evaluate the accuracy, efficiency, and robustness of our selected NN methods, we collect 7 metrics which can be categorized into two groups: performance metrics and functional metrics. They are summarized in Table 3. Furthermore, we provide the definition of one metric from each of the aforementioned groups. The L_2 norm relative error is defined as L2RE $= \|u_\theta - u\|_2/\|u\|_2$. The Lipschitz constant **Lips** is defined as the smallest L satisfying

$$\|u_\theta(\mathbf{x}) - u_\theta(\mathbf{y})\|_2 \le L\|\mathbf{x} - \mathbf{y}\|_2, \quad \forall \mathbf{x}, \mathbf{y}.$$

Group	Metrics	Symbolic meanings
Performance	L2RE	L_2 norm relative error
	mERR	L_∞ norm error
	t_{train}	Model training time
	t_{infer}	Model inference time for one sample
	t_{conv}	The time of model convergence
Functional	Cove	Model's coverage ability on multiple tasks
	Lips	The Lipschitz constant

Table 3: Metrics of PDENNEval

Here $\|\cdot\|_2$ is the L_2 norm, $\mathbf{u}_\theta$ is the predictive solution, and $\mathbf{u}$ is the reference solution. The definition of other metrics and their detailed description are given in Appendix D.

4 A Selection of Evaluation Results

In this section, we separately present partial experimental results for function learning-based NN methods and operator learning-based NN methods. Additional results can be found in Appendix F. For a fair comparison, all our experiments are conducted using the following system configuration: CPU : $2\times$ Intel Xeon Gold 6230R @ 2.10GHz; GPU : $1 \times$ NVIDIA TITAN RTX; Software : PyTorch@1.13.1, CUDA@11.6.

Recall that we have introduced 12 NN methods and 19 PDE tasks in Section 2. In our experiments, to test the performance of function learning NN methods, we introduce three additional PDE tasks P_S, P_L, P_H which will be explained in details in Section 4.1. Apparently, in PDENNEval, not every method has been applied to all 19 PDE tasks. Listed in Table 4 is a summary of tasks coverage by our selected 12 NN methods.

From this table, we observe that PINN, DFVM, and RFM have been used to solve all 19 PDE problems; Meanwhile, the function learning methods WAN, DRM, and DFLM have only been applied to a part of these PDE problems. Following the vanilla methods, we use DRM [E and Yu, 2018] and DFLM [Han *et al.*, 2020] only to solve symmetric and time-independent problems P_S, P_L, P_H and DF2. In the same way, we use WAN [Zang *et al.*, 2020] only to solve scalar problems. In contrast, all operator learning type methods have not been used to solve the three Poisson tasks. Moreover, DeepONet, PINO and MPNN do not solve 3D problems, while the MPNN does not solve the time-independent problem DF2 either. It is worth mentioning that we do not mean that a specific NN method can not be applied to solve some specific PDE problems if it has not been done in our PDENNEval, it simply suggests that implementing this NN method for individual equation problems may pose challenges.

4.1 Results on Function Learning Methods

As already pointed out in Table 4, function learning-based NN methods can solve almost all kinds of PDE tasks. Here, we primarily present the experimental results of three tasks P_H, P_S and P_L, which tackle three special mathematical challenges high-dimensionality, singularity, and complex geometry. The governing PDE of these three tasks is the Poisson

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
P_S	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×
P_L	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×
P_H	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×
Ad1	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
DR1	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
Bu1	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
DS1	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
AC1	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
CH1	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓
NS1	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓
Bu2	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓
NS2	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓
DF2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
SW2	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓
AC2	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
BS2	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
Ma3	✓	✓	✓	×	×	×	×	×	✓	✓	✓	×
Eu3	✓	✓	✓	×	×	×	×	×	✓	✓	✓	×
NS3	✓	✓	✓	×	×	×	×	×	✓	✓	✓	×

Table 4: Tasks coverage of NN methods. Here M1-M12 denote the NN methods PINN, DFVM, RFM, WAN, DRM, DFLM, DeepONet, PINO, U-NO, FNO, U-Net, MPNN, respectively.

	d	PINN	DFVM	RFM	WAN	DRM	DFLM
L2RE	3	0.0019	0.0019	0.0074	0.0672	0.0176	0.0311
	5	0.0017	0.0017	0.0479	0.0431	0.0035	0.0034
	10	0.0026	0.0025	0.3079	0.0573	0.0047	0.0038
	20	0.0035	0.0035	0.5133	0.2503	0.0062	0.0043
	40	0.0037	0.0036	0.5108	0.1316	0.0090	0.0057
	80	0.0067	0.0062	0.5678	0.0159	0.0378	0.0296
	120	0.0129	0.0112	0.6654	0.0230	0.1066	0.1053
t_{train}(s)	3	395.5	145.3	-	115.7	147.1	302.8
	5	549.7	156.0	-	135.1	150.1	303.1
	10	1,289	221.6	-	171.4	299.6	320.6
	20	2,512	290.3	-	291.8	314.2	324.6
	40	4,526	412.7	-	444.4	314.3	325.7
	80	7,472	776.0	-	818.7	328.2	329.7
	120	10,701	1,151	-	1,272	344.3	370.1

Table 5: L2RE and t_{train} of function learning methods for problem P_H.

equation

$$-\Delta u = f, \text{in } \Omega, \quad u = g, \text{on } \partial\Omega. \tag{11}$$

where Ω, f, g will be specified later.

1) High dimensional case (P_H). In the first case, we evaluate the performance of function learning-based NN methods on high dimensional Poisson equations. Let $\Omega = [-1,1]^d$, here $d = 3, 5, 10, 20, 40, 80, 120$ is the spatial dimension. The functions f and g are chosen so that $u(\mathbf{x}) = \left(\frac{1}{d}\sum_{i=1}^{d} x_i\right)^2 + \sin\left(\frac{1}{d}\sum_{i=1}^{d} x_i\right), \forall \mathbf{x} \in \mathbf{R}^d$, which is sufficiently smooth.

Listed in Table 5 are L_2 norm relative error and training times of function learning methods for P_H. From Table 5, we observe that all methods are immune to the *curse of dimensionality*. That is, the computation time of no method increases exponentially in terms of the dimension. Moreover, PINN exhibits superior performance in terms of the L2RE metric. The L2RE of the DFVM is almost as good as the PINN. In addition, on evaluation on the training time metric t_{train}, the WAN excels for the $3, 5, 10, 20$ dimensional cases, while DRM excels for the 40, 80 and 120 dimension cases. It is worth mentioning that since the RFM solution is obtained by solving a linear system of algebraic equations instead of training, no training time has been reported for this method.

	PINN	DFVM	RFM	WAN	DRM	DFLM
L2RE	1.0793	0.0209	1.0298	0.0543	1.0050	0.0207
t_{train}(s)	235.7	134.0	-	175.4	141.2	228.9

Table 6: L2RE and t_{train} of function learning methods for problem P_S.

2) Singular case (P_S). In the second case, we consider the equation 11 with a singular solution. Precisely, let $\Omega = [0,1]^2$ and f, g be selected to ensure $u(x_1, x_2) = x_1^2$ if $0 \leq x_1 \leq \frac{1}{2}$ and $u(x_1, x_2) = (1 - x_1)^2$ otherwise. Note that the exact solution u is continuous but not smooth at the location $x_1 = \frac{1}{2}$.

Listed in Table 6 are L2RE and t_{train} of function learning methods for this Poisson singular problem. We observe that in this case, the error of the DFLM is the smallest, followed by the DFVM. Moreover, the training time of the DFVM is the shortest. The error of the PINN and the RFM seems to be too big, which might be caused by the fact that they use the strong form PDE to construct their loss.

3) L-shape domain case (P_L). In the third case, we choose $\Omega = [-1,1]^2 \backslash (0,1]^2$ and $f = 1, g = 0$. There is no analytical solution for this problem. We utilize the software COMSOL to generate a numerical solution whose configuration is shown in Fig1.

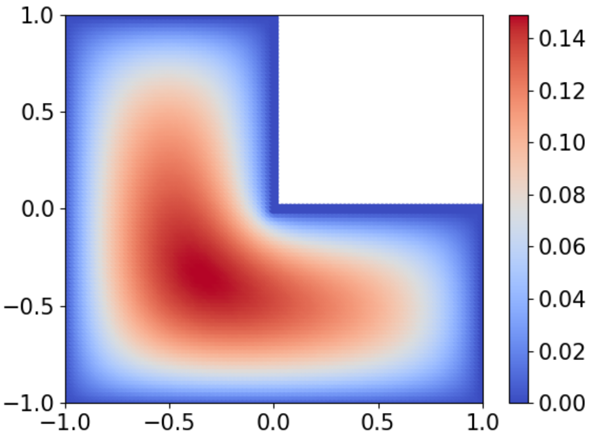

Figure 1: The solution of the problem P_L.

	PINN	DFVM	RFM	WAN	DRM	DFLM
L2RE	0.0042	0.0043	0.0189	0.0539	0.0216	0.5576
t_{train}(s)	233.9	289.9	-	182.1	94.80	173.5

Table 7: L2RE and t_{train} of function learning methods for problem P_L.

Listed in Table 7 are the L_2 norm relative errors and training times for the problem P_L. We observe that the error of the PINN is the smallest, followed by the DFVM. The training time of the DRM is the shortest.

Note that here, only results concerning L2RE and t_{train} are reported, those concerning other metrics such as mERR, t_{infer} and t_{conv} will be presented in the Appendix F.

4.2 Results on Operator Learning Methods

The six operator learning methods to be measured in PDENNEval are DeepONet, PINO, U-NO, FNO, U-Net, and MPNN, which are data-driven. Therefore, they are applied to solve 16 PDE problems except the three Poisson problems in the previous section, as shown in Table 4. Due to the space limitation, here we only present the evaluation results of the following three equations: the 1D Advection equation (Ad1), the 2D Allen-Cahn equation (AC2), and the 2D Darcy Flow equation (DF2). In our experiments, 90% of the data in each dataset is used for training, and 10% is used for testing. Moreover, here we only report the evaluation results in terms of the accuracy metric L2RE and three time metrics: the training time t_{train}, the inference time t_{infer}, and the convergence time t_{conv}.

Listed in Table 8 are the L_2 norm relative error of the six operator learning methods. We observe that the U-NO achieves the best computational accuracy for problems Ad1 and AC2, while PINO achieves the best accuracy for the problem DF2. Note that we did not employ MPNN to solve the DF2.

Listed in Table 9 are the training, inference and convergence time. We observe that for all three PDE problems, DeepONet demonstrates noteworthy efficiency.

	DeepONet	PINO	U-NO	FNO	U-Net	MPNN
Ad1	0.0792	0.0191	0.0069	0.0128	0.0699	0.0451
AC2	0.9999	0.0191	0.0054	0.0084	0.1117	0.0079
DF2	0.3971	0.0709	0.0702	0.1328	0.0828	-

Table 8: L2RE of operator learning methods for Ad1,AC2 and DF2.

		DeepONet	PINO	U-NO	FNO	U-Net	MPNN
t_{train}	Ad1	6.6e02	9.0e03	2.9e04	1.1e04	2.4e04	2.6e04
	AC2	5.3e03	8.5e05	2.0e05	9.0e04	6.7e04	3.1e04
	DF2	1.1e03	1.4e05	2.2e04	8.4e03	1.3e04	-
t_{infer}	Ad1	8.0e-04	2.1e-03	2.9e-01	2.5e-01	7.7e-02	1.8e-02
	AC2	6.1e-02	1.7e-02	4.3e-01	3.6e-01	3.7e-01	3.8e-01
	DF2	1.3e-03	2.1e-03	2.3e-01	2.1e-01	3.6e-03	-
t_{conv}	Ad1	5.7e02	7.2e03	2.7e04	8.7e03	2.3e04	2.6e04
	AC2	3.2e01	8.5e04	4.2e04	1.9e04	6.7e04	3.1e04
	DF2	8.8e02	8.2e03	1.8e04	5.1e03	1.2e04	-

Table 9: Training, inference and convergence time of operator learning methods.

5 Concluding Remarks

In this study, we evaluate 6 function learning and 6 operation learning NN methods for solving 19 PDEs. Our experiments with function learning methods demonstrate that, for the PDEs with smooth solution, the PINN exhibits highest accuracy, and the DRM and WAN exhibits excellent training speed. In the case of PDE with non-smooth solution, the DFVM and WAN perform very well. Our experiments with operator learning methods reveal that, for autoregressive tasks, both U-NO and FNO perform admirably, while for non-autoregressive tasks, DeepONet exhibits the best performance.

Certainly, many other important NN methods, including variants of PINN, and many other fundamental PDE tasks such as multiscale problems in various fields have not been covered yet in PDENNEval. Our evaluation system is scalable and easily accessible, everyone is welcome to contribute data and/or codes to the https://github.com/Sysuzqs/PDENNEval.

Developing PDE solvers with high accuracy, efficiency, and strong generalization is the pursuit of many computational scientists. The combination of traditional numerical methods and NN methods provides technical possibilities for achieving this pursuit. It can be anticipated that in the near future, there will likely be the emergence of many large NN models capable of efficiently solving lots of complex PDEs simultaneously (see e.g. [Yang *et al.*, 2023a; Yang *et al.*, 2023b]). Therefore, in the future, we will pay much attention to the evaluation of large NN models for solving PDE problems.

Acknowledgments

This work was supported by the National Science and Technology Major Project (2022ZD0117805), by the National Natural Science Foundation of China under grants 92370113 and 12071496, and by the Natural Science Foundation of the Guangdong Province grant 2023A1515012097.

We thank graduate students Haolong Fan, Hongji Li and Changye He, all from Sun Yat-sen University, for their helps with numerical experiments.

References

[Barron, 1994] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14:115–133, 1994.

[Brandstetter *et al.*, 2022] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Cen and Zou, 2023] Jianhuan Cen and Qingsong Zou. Deep finite volume method for high-dimensional partial differential equations. *arXiv preprint arXiv:2305.06863*, 2023.

[Chen *et al.*, 2022] Jingrun Chen, Xurong Chi, Zhouwang Yang, et al. Bridging traditional and machine learning-based algorithms for solving PDEs: The random feature method. *arXiv preprint arXiv:2207.13380*, 2022.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dosovitskiy *et al.*, 2020] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

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

[Gupta and Brandstetter, 2022] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

[Han *et al.*, 2020] Jihun Han, Mihai Nica, and Adam R Stinchcombe. A derivative-free method for solving elliptic partial differential equations with deep neural networks. *Journal of Computational Physics*, 419:109672, 2020.

[Hao *et al.*, 2023] Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, et al. PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs. *arXiv preprint arXiv:2306.08827*, 2023.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[He *et al.*, 2022] Juncai He, Lin Li, and Jinchao Xu. Approximation properties of deep relu cnns. *Research in the mathematical sciences*, 9(3):38, 2022.

[Hornik, 1991] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[Kunz and Luebbers, 1993] Karl S Kunz and Raymond J Luebbers. *The finite difference time domain method for electromagnetics*. CRC press, 1993.

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

[Li *et al.*, 2020] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[Li *et al.*, 2021] Zongyi Li, Hongkai Zheng, Nikola B. Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *CoRR*, abs/2111.03794, 2021.

[Lu *et al.*, 2021a] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[Lu *et al.*, 2021b] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[Lu *et al.*, 2022] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[Rahman *et al.*, 2022] Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

[Raissi and Karniadakis, 2018] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[Raissi *et al.*, 2019] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[Raissi, 2024] Maziar Raissi. Forward-backward stochastic neural networks: deep learning of high-dimensional partial differential equations. In *Peter Carr Gedenkschrift: Research Advances in Mathematical Finance*, pages 637–655. World Scientific, 2024.

[Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[Takamoto *et al.*, 2022a] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench Datasets, 2022.

[Takamoto *et al.*, 2022b] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[Toro *et al.*, 1994] Eleuterio F Toro, Michael Spruce, and William Speares. Restoration of the contact surface in the hll-riemann solver. *Shock waves*, 4:25–34, 1994.

[Van Leer, 1979] Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Yang *et al.*, 2023a] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J. Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

[Yang *et al.*, 2023b] Liu Yang, Siting Liu, and Stanley J. Osher. Fine-tune language models as multi-modal differential equation solvers. *arXiv preprint arXiv:2308.05061*, 2023.

[Zang *et al.*, 2020] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.