Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(3): 2009–2037.
DOI: 10.3934/era.2026090
Received: 16 December 2025
Revised: 01 February 2026
Accepted: 12 February 2026
Published: 05 March 2026

Research article

PIP^2 Net: Physics-informed partition penalty deep operator network

Hongjin Mi[1], Huiqiang Lun[2], Changhong Mou[3] and Yeyu Zhang[1,*]

[1] School of Mathematics, Shanghai University of Finance and Economics, No. 777 Guoding Road, Shanghai 200433, China

[2] Faculty of Liberal Arts and Professional Studies, York University, 4700 Keele St, North York, ON M3J1P3, Canada

[3] Department of Mathematics and Statistics, Utah State University, 900 Old Main Hill, Logan, UT 84322, USA

* **Correspondence:** Email: zhangyeyu@mail.shufe.edu.cn.

Abstract: Operator learning has become a powerful tool for accelerating the solution of parameterized partial differential equations (PDEs), enabling rapid prediction of full spatiotemporal fields for new initial conditions or forcing functions. Existing architectures such as the deep operator network (DeepONet) and the Fourier neural operator (FNO) show strong empirical performance, but often require large training datasets, lack explicit physical structure, and may suffer from instability in their trunk-network features, where mode imbalance or collapse can hinder accurate operator approximation. Motivated by the stability and locality of classical partition-of-unity (PoU) methods, we investigate PoU-based regularization techniques for operator learning and develop a revised formulation of the existing POU–PI–DeepONet framework. The resulting physics-informed partition penalty deep operator network (PIP^2 Net) introduces a simplified and more principled partition penalty that improves the coordinated trunk outputs, which leads to more expressiveness without sacrificing the flexibility of DeepONet. We evaluate PIP^2 Net on three nonlinear PDEs: the viscous Burgers equation, the Allen–Cahn equation, and a diffusion–reaction system. The results show that it consistently outperforms DeepONet, PI-DeepONet, and POU-DeepONet in prediction accuracy and robustness.

Keywords: deep operator networks; physics-informed learning; partition-of-unity; nonlinear PDEs; spatiotemporal dynamics

1. Introduction

Many problems in engineering and physics are described by parameterized partial differential equations (PDEs) [1–3]. In such problems, we often need to solve the same PDE many times, for

different initial conditions, boundary conditions, source terms, or physical parameters [4, 5]. Standard numerical methods, such as finite difference, finite element, or spectral schemes, can be very accurate for a single solution, but they quickly become too expensive in multi-query settings like uncertainty quantification, inverse problems, design optimization, and real-time control [6–8]. The demand for such computational efficiency becomes especially clear in complex systems in engineering. For example, in autonomous maritime operations, real-time adaptive collision avoidance for unmanned surface vehicles (USVs) in perception-limited environments demands rapid simulation of guidance laws and surrounding traffic dynamics [9–11]. Similarly, in climate science, real-time forecasting and data assimilation under sparse and noisy observations require fast surrogate simulations of coupled multi-models to support timely risk assessment and decision-making [12–14]. This motivates the use of operator learning [15, 16], a framework that learns nonlinear mappings between infinite-dimensional function spaces, which further allows input data such as initial conditions to be mapped directly to the full PDE solution. Operator learning has therefore become a powerful tool for accelerating PDE-related simulations. Instead of predicting a single output for fixed inputs, operator learning approximate the solution operator itself with the mapping of an given function,e.g., an initial condition or forcing function to the corresponding spatiotemporal field of the governing equations [17, 18].

Operator learning can be broadly categorized into two different architectures: deep operator network (DeepONet) and Fourier neural operator. Both tends to learn solution operators between infinite-dimensional function spaces, but they differ in how the mapping is represented and evaluated. The deep operator network (DeepONet), introduced by Lu et al. [16] and supported by the universal operator approximation theorem of Chen and Chen [19], learns operators by splitting the representation into two components: a branch network that takes the input function, and a trunk network that includes spatial–temporal coordinates. The branch network outputs a set of coefficients derived from the input function, while the trunk network evaluates features at the desired coordinates. The final prediction is obtained by combining these two outputs. In this way, DeepONet can infer the entire solution field for new inputs that captures nonlinear operator mappings without requiring explicit spatial discretization of the underlying PDE. On the other hand, the Fourier neural operator (FNO) adopts a spectral approach [15]. Rather than working in physical space, FNO learns the operator by applying trainable convolution kernels in Fourier space that enables efficient representation of global interactions across the domain. By working in the frequency domain, FNO can efficiently capture broad spatial influence and still scale well as resolution increases, since updates are performed with fast Fourier transforms instead of pointwise operations. Purely data-driven operator networks, however, often require large quantities of high-fidelity training data and may generalize poorly outside the distribution they are trained on. A natural strategy to reduce this data burden is to incorporate physics information into the learning process. Physics-informed neural networks (PINNs) [20] introduce this idea by adding the governing PDE residual, along with initial and boundary conditions directly into the loss function through automatic differentiation. Building on this formulation, the physics-informed deep operator network (PI-DeepONet) [21] extends physics regularization to operator learning by adding the standard DeepONet data loss with physics residual terms. This physics-informed mechanism potentially improves data efficiency and robustness, and has motivated a growing number of research on physics-informed neural operator networks. Recent research has pushed PI-DeepONet forward in several important ways. Adaptive loss balancing

approaches have been introduced to mitigate competition between data and physics residuals and improve training stability [22–24]. Multiscale and domain-decomposed extensions [25–28] have also been shown to improve performance in problems with sharp gradients, heterogeneous media, or stiff dynamics.

A key difficulty underlying these challenges lies in how the operator is represented;specifically, in the features generated by the trunk network. In DeepONet, the trunk outputs act as coordinate-dependent modes that combine with branch coefficients to form the predicted solution field. When no structural guidance is imposed, these modes may become unbalanced: certain components grow disproportionately while others collapse toward zero. This imbalance can degrade conditioning, increase sensitivity to hyperparameters, and reduce interpretability of the learned operator. In contrast, classical numerical methods often maintain stability through structured local representations. Among them, the *partition-of-unity* (PoU) framework is especially notable: PoU basis constructions have long been used in meshfree and radial-basis-function (RBF) approximations [29–34] to assemble globally smooth solutions from balanced local components. Motivated by these advantages, PoU concepts have recently begun to appear in learning-based models as well, including probabilistic regression [35] and PoU-augmented DeepONets [36]. These developments show that the trunk network's representation is not just a side effect of training, it in fact directly affects how well the operator is learned.

Building on these developments, in this work, we propose a more general way to bring PoU structure into physics-informed operator learning. In particular, we propose the physics-informed partition penalty deep operator network (PIP^2 Net). PIP^2 Net retains the branch–trunk architecture of DeepONet and the physics-based loss used in PI-DeepONet, and introduces a partition penalty (P^2) applied to the trunk outputs. This penalty encourages the trunk outputs to satisfy a global normalization condition, either on their values or on their magnitudes. Intuitively, this makes the collection of trunk outputs behave more like a coordinated family of basis functions, similar in spirit to PoU used in classic numerical methods. This allows us to separate the effects of physics-based losses and partition-based regularization, and to see how they interact in the full PIP^2 Net. We then study these models on three representative nonlinear PDEs: the viscous Burgers equation, the Allen–Cahn equation, and a diffusion–reaction system.

The rest of the paper is organized as follows. In Section 2, we first review DeepONet and PI-DeepONet and then introduce the partition penalty and the resulting PIP^2 Net architecture. We then show numerical results for the Burgers, Allen–Cahn, and diffusion–reaction equations, comparing DeepONet, PI-DeepONet, POU-DeepONet, and PIP^2 Net in Section 3. Finally, in Section 4, we summarize the findings and discuss possible extensions in the future work.

2. Physics-informed partition penalty deep operator network (PIP^2 Net)

2.1. Physics-informed deep operator network (PI-DeepONet)

Lu et al. [16] proposed the general deep operator network (DeepONet) that relies on the universal operator approximation theorem [19] that describes the learnability of nonlinear operator mappings between infinite-dimensional function spaces. To illustrate the DeepONet framework for a general

partial differential equation (PDE), consider a PDE of the form

$$\mathcal{F}(u(x,t);\kappa) = 0, \qquad (x,t) \in \Omega \times [0,T], \tag{2.1}$$

where $\mathcal{F}$ denotes a (possibly nonlinear) differential operator, and κ represents the input information associated with the problem such as initial or boundary data, forcing terms, or physical parameters. DeepONet aims to learn the corresponding solution operator

$$\mathcal{G}(\kappa) = u(\cdot;\kappa), \tag{2.2}$$

which maps the input κ to the full solution field u. A DeepONet can be interpreted as a class of neural network models to approximate $\mathcal{G}$. In particular, we choose the notation G_θ to represent the DeepONet approximation of $\mathcal{G}$,

$$\mathcal{G} \approx G_\theta, \tag{2.3}$$

where θ is the set of trainable network parameters. Then, a DeepONet usually consists of two subnetworks [16, 37]: the branch net and the trunch net, where the estimator at $\mathbf{x} \in M$ is obtained as the inner product of their outputs:

$$G_\theta(\kappa(\Xi))(\mathbf{x}) = \sum_{k=1}^{p} br_k(\kappa(\xi_1), \kappa(\xi_2), \ldots, \kappa(\xi_m))\, tr_k(\mathbf{x}) + br_0, \tag{2.4}$$

where $br_0 \in \mathbb{R}$ is a bias, $\{br_1, br_2, \ldots, br_p\}$ are the p outputs of the branch net, and $\{tr_1, tr_2, \ldots, tr_p\}$ are the p outputs of the trunk net. More specifically, the branch network represents κ in a discrete format is given by

$$\kappa(\Xi) = \{\kappa(\xi_1), \kappa(\xi_2), \ldots, \kappa(\xi_m)\}, \tag{2.5}$$

where $\kappa(\Xi)$ is a vector which consists the evaluation of input functions at different sensors $\Xi = \{\xi_1, \xi_2, \ldots, \xi_m\}$. The trunk network may take as input a spatial location $\mathbf{x} \in M$ [16, 21, 37] which yields the following:

$$\mathrm{tr}(\mathbf{x}) = (\mathrm{tr}_1(\mathbf{x}), \ldots, \mathrm{tr}_p(\mathbf{x})) \in \mathbb{R}^p, \tag{2.6}$$

where $\mathbf{x}$ denotes the spatial–temporal coordinate. Our goal is to predict the solution corresponding to a given κ, which is also evaluated at the same input location $\mathbf{x} \in M$. The representation of κ through pointwise evaluations at arbitrary sensor locations provides additional flexibility, both during training and in prediction, particularly when κ is available only through its values at these sensor points.

Given the training data with labels in the two sets of inputs and outputs,

$$\text{input:} \qquad \left\{\kappa^{(k)}, \Xi, (\mathbf{x}_i^{(k)})_{i=1,\ldots,N}\right\}_{k=1,\ldots,N_{\text{data}}}, \tag{2.7}$$

$$\text{output:} \qquad \{u(\mathbf{x}_i^{(k)}; \kappa^{(k)})\}_{i=1,\ldots,N,\ k=1,\ldots,N_{\text{data}}}. \tag{2.8}$$

We minimize a loss function that measures the discrepancy between the true solution operator $G(\kappa^{(k)})$ and its DeepONet approximation $G_\theta(\kappa^{(k)})$:

$$\widehat{\mathcal{L}}_{\text{data}}(\theta) = \frac{1}{N_{\text{data}} N} \sum_{k=1}^{N_{\text{data}}} \sum_{i=1}^{N} \left| G_\theta(\kappa^{(k)}(\Xi))(\mathbf{x}_i^{(k)}) - G(\kappa^{(k)})(\mathbf{x}_i^{(k)}) \right|^2, \tag{2.9}$$

where $X^{(k)} = \{\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_N^{(k)}\}$ denotes the set of N evaluation points in the domain of $G(\kappa^{(k)})$. In practice, however, the analytical solution of the PDE is not available. Instead, we rely on numerical simulation obtained from high fidelity numerical solvers, which can be denoted as:

$$\{\hat{u}(\mathbf{x}_i^{(k)}; \kappa^{(k)})\}_{i=1,\ldots,N,\ k=1,\ldots,N_{\text{data}}}. \tag{2.10}$$

Then, the practical training tends to minimize the loss function

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N_{\text{data}}N} \sum_{k=1}^{N_{\text{data}}} \sum_{i=1}^{N} \left| G_\theta(\kappa^{(k)}(\Xi))(\mathbf{x}_i^{(k)}) - \hat{u}(\mathbf{x}_i^{(k)}; \kappa^{(k)}) \right|^2. \tag{2.11}$$

The physics-informed deep operator network (PI-DeepONet) [21] integrates the idea of physics-informed neural networks (PINN) [20] with the DeepONet framework. The PI-DeepONet introduces the additional PDE residual loss function and boundary condition loss function. Consequently, the loss function for PI-DeepONet is defined as

$$\mathcal{L}(\theta) = w_{\text{data}}\mathcal{L}_{\text{data}}(\theta) + w_{physics}\mathcal{L}_{physics}(\theta) + w_{\text{bc}}\mathcal{L}_{\text{bc}}(\theta), \tag{2.12}$$

where $\mathcal{L}_{\text{physics}}(\theta)$ enforces the PDE residual, $\mathcal{L}_{\text{bc}}(\theta)$ enforces the boundary conditions, and $\mathcal{L}_{\text{data}}(\theta)$ accounts for available observational data, which also includes the initial condition. The weights $w_{\text{data}}, w_{\text{physics}}$ and w_{bc} are tunable hyperparameters that balance the contributions of the individual loss terms. The boundary-condition loss yields the following:

$$\begin{aligned} \mathcal{L}_{BC}(\theta) = &\frac{1}{NP} \sum_{i=1}^{N} \sum_{j=1}^{P} \left| G_\theta(u^{(i)})(0, t_{bc,j}^{(i)}) - G_\theta(u^{(i)})(1, t_{bc,j}^{(i)}) \right|^2 \\ &+ \frac{1}{NP} \sum_{i=1}^{N} \sum_{j=1}^{P} \left| \partial_x G_\theta(u^{(i)})(0, t_{bc,j}^{(i)}) - \partial_x G_\theta(u^{(i)})(1, t_{bc,j}^{(i)}) \right|^2, \end{aligned} \tag{2.13}$$

which enforces periodicity and consistency of spatial derivatives, while the PDE residual term

$$\mathcal{L}_{physics}(\theta) = \frac{1}{NR} \sum_{i=1}^{N} \sum_{j=1}^{R} \left| \mathcal{F}(G_\theta(u^{(i)});\ x_{r,j}^{(i)}, t_{r,j}^{(i)}) \right|^2, \tag{2.14}$$

guarantees that the predicted solution satisfies the PDEs (2.1) at a set of residual points.

2.2. The partition penalty (P^2) in trunk net

The partition of unity (PoU) framework has a long and influential history in scientific computing [29–34], where it provides a principled mechanism for blending local approximations into globally smooth representations. Its flexibility, numerical stability, and compatibility with meshfree discretizations have also motivated a growing number of applications in scientific machine learning. Notably, [35] adopts PoU ideas for probabilistic regression, while [36] incorporates PoU constructions within the deep operator network (DeepONet) architecture to enhance expressivity and local adaptability.

Motivated by these developments, we introduce the *partition penalty* (P^2), a mechanism that extends the PoU philosophy to physics informed operator learning framework by explicitly regularizing the trunk network. Conceptually, P^2 encourages the trunk outputs to behave like a coordinated collection of basis functions, analogous to PoU weight functions. Practically, this is achieved by augmenting the loss function (2.12) with an additional penalty term applied to the trunk network (2.6). The goal is to promote a global structure among the trunk outputs that supports stable and interpretable operator approximation. Formally, the partition penalty enforces a normalization constraint of the form

$$\sum_{j=1}^{p} \mathrm{tr}_j(\mathbf{x}) = c(\theta), \tag{2.15}$$

where the normalization factor $c(\theta)$ may be prescribed a priori or depend on trainable parameters θ. A widely used and practically effective choice is the unit normalization

$$c(\theta) \equiv 1, \tag{2.16}$$

which forces the collection $\{\mathrm{tr}_j\}_{j=1}^{p}$ to collectively form a partition over the evaluation domain. In situations where sign changes in the trunk outputs are desirable, for example, when representing mixed-mode or oscillatory structures, a natural surrogate of the penalty enforces normalization on their magnitudes yields the following:

$$\sum_{j=1}^{p} \left\| \mathrm{tr}_j(\mathbf{x}) \right\| = c(\theta), \tag{2.17}$$

where $\|\cdot\|$ denotes the norm of interest. This preserves the essential balance of the normalization while allowing each trunk component to take positive or negative values.

The introduction of P^2 introduces several important advantages. First, by enforcing a global normalization, it mitigates pathological behaviors in which a subset of trunk outputs dominates while others collapse toward zero; such imbalance is a known issue in operator networks, especially for operators with multiscale or spatially heterogeneous structure. Second, a near-partition constraint improves the conditioning of the operator map by ensuring that all trunk outputs contribute meaningfully and comparably during training. Finally, the P^2 mechanism encourages the trunk network to generate basis-like functions whose collective behavior resembles classical PoU systems.

2.3. The Physics-informed partition-penalty deep operator network (PIP^2 Net)

Building on the PI-DeepONet framework and the partition-penalty (P^2) regularization of the trunk network, we propose the *p*hysics-*i*nformed *p*artition-*p*enalty deep operator network (PIP^2 Net). This architecture combines a physics-informed operator learning framework with PoU inspired by structural regularization. Figure 1 illustrates the PIP^2 Net framework. In general, the PIP^2 Net retains the standard DeepONet representation

$$G_\theta(\kappa(\Xi))(\mathbf{x}) = \sum_{k=1}^{p} br_k(\kappa(\Xi))\, tr_k(\mathbf{x}) + br_0, \tag{2.18}$$

where the branch network encodes the input function κ and the trunk network provides a set of basis-like outputs $\{tr_k(\mathbf{x})\}_{k=1}^{p}$. The physics-informed component imposes the PDE residual, boundary

conditions, and available data through the loss (2.12), ensuring that the predicted operator adheres to the governing equation (2.1).

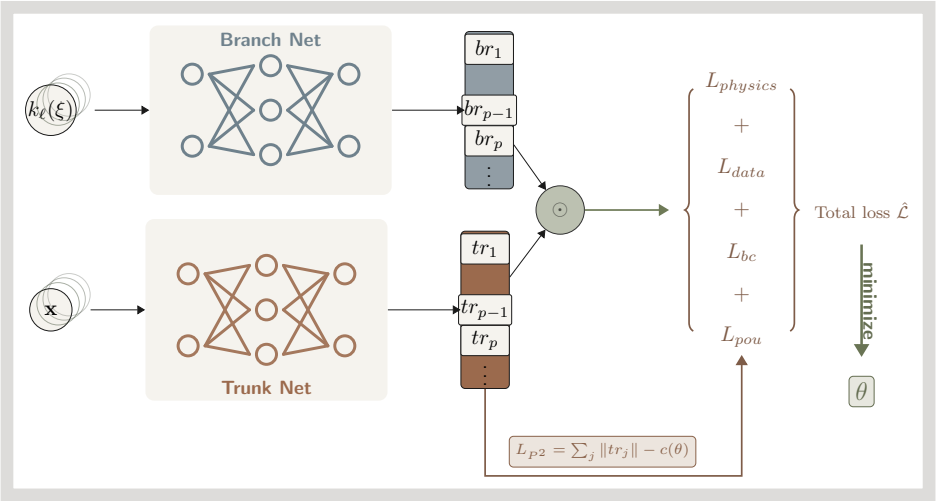


Figure 1. Illustration of PIP^2 Net.

To further improve the stability and representational structure of the trunk outputs, PIP^2 incorporates the partition penalty (P^2) introduced in the previous subsection. Specifically, at each spatial–temporal coordinate $\mathbf{x}$, the trunk outputs are constrained by the P^2 loss:

$$\mathcal{L}_{P^2} = \sum_{j=1}^{p} \|tr_j(\mathbf{x})\| - c(\theta), \tag{2.19}$$

p denotes the output dimension of the trunk network, each $tr_k(\mathbf{x})$ represents the kth trunk component at $\mathbf{x}$, which is paired with the corresponding branch coefficient in the operator representation.

With $c(\theta)$ either a prescribed constant or a learnable parameter. This penalty loss includes the PoU-like structure among the trunk outputs, preventing mode collapse and reducing ill-conditioning in the learned operator representation. Then, the combined loss function for PIP^2 Net therefore takes the form

$$\mathcal{L}_{\text{PIP}^2}(\theta) = w_{\text{data}}\mathcal{L}_{\text{data}}(\theta) + w_{physics}\mathcal{L}_{physics}(\theta) + w_{\text{bc}}\mathcal{L}_{\text{bc}}(\theta) + \lambda_{P^2}\mathcal{L}_{P^2}(\theta), \tag{2.20}$$

where $w_{\text{data}}, w_{physics}$ and w_{bc} are pre-determined or adaptive coefficients, and λ_{P^2} is the regularization coefficient. By integrating physics-informed losses with partition-based trunk regularization, the PIP^2 Net provides a consistent operator-learning framework that enforces the governing PDE and boundary conditions with specific stabilization of the trunk representation.

3. Numerical tests

To evaluate the proposed PIP^2 Net, we test a series of numerical examples for three representative nonlinear PDEs: the Burgers equation, the Allen–Cahn equation, and a diffusion—reaction equation. As summarized in Table 1, our comparisons include four different operator learning frameworks:

DeepONet, PI-DeepONet, POU-DeepONet, and the proposed PIP^2 Net. In all test problems, we provide quantitative comparisons of different models using the standard relative L^2 error between the predicted solution $\widehat{u}$ and the benchmark solution u:

$$\mathcal{E}_{L^2} = \frac{\left(\int_0^T \int_\Omega |u(x,t) - \widehat{u}(x,t)|^2 dxdt\right)^{1/2}}{\left(\int_0^T \int_\Omega |u(x,t)|^2 dxdt\right)^{1/2}}, \tag{3.1}$$

where Ω denotes the spatial domain and $[0, T]$ is the time interval over which the solution is evaluated.

Table 1. Summary of models used in numerical tests.

Acronym	Full Name	Physics-informed	Partition	Normlization
DeepONet	Deep Operator Network	N/A	N/A	N/A
PI-DeepONet	Physics-informed Deep Operator Network	Yes	N/A	N/A
POU-DeepONet	Partition Of Unity Deep Operator Network	N/A	Yes	N/A
PIP^2 Net	Physics-informed Partition Penalty Deep Operator Network	Yes	Yes	Yes

3.1. One-dimensional Burgers' equation

We first consider the one-dimensional viscous Burgers' equation, which is a standard model for nonlinear advection and shock dynamics [15, 37–39]. Let $u(x, t)$ denote the velocity field that evolves from an initial profile $u_0(x)$ with viscosity ν. The dynamics are governed by the following PDE with initial condition (IC) and periodic boundary conditions (BCs):

Operator learning seeks to approximate the solution operator that maps an initial condition u_0 to the corresponding solution $u(x, t)$ of (3.3). In the numerical experiments, we fix the viscosity parameter to $\nu = 0.01$. Following [15, 37], the initial fields $u_0(x)$ are sampled from a Gaussian random field (GRF) with zero mean and a Matérn-type covariance operator,

$$u_0 \sim \mathcal{N}\Big(0,\ \sigma^2 (-\Delta + \tau^2 I)^{-\gamma}\Big), \tag{3.2}$$

which provides a rich family of spatially correlated and potentially rough initial profiles for training and testing the operator-learning model.

Training and testing data For each initial function $u^{(i)}(x)$, collocation points are sampled as follows. At the spatial grid points $x_{ic,j}^{(i)} = x_j$, we randomly sample temporal points to form the boundary sets $\{(0, t_{ic,j}^{(i)})\}_{j=1}^P$ and $\{(1, t_{ic,j}^{(i)})\}_{j=1}^P$, as well as the interior residual set $\{(x_{r,j}^{(i)}, t_{r,j}^{(i)})\}_{j=1}^Q$, where $P = 100$ and $Q = 2500$. A total of 1000 initial functions are randomly generated from a Gaussian random field $\mathcal{N}(0, 5^2(-\Delta + 5^2 I)^{-4})$. For each realization, training and testing data are obtained by

numerically solving the governing equation with periodic boundary conditions and initial condition $s(x, 0) = u(x)$ for $x \in [0, 1]$. The numerical solutions are computed using the Chebfun package [40] in MATLAB, which employs a Fourier spectral discretization in space together with a fourth-order exponential time-differencing scheme for temporal integration. The Adam optimizer is used to minimize the loss function during training. From the 1000 generated initial functions, 200 are selected for training.

Neural network architectures Both the branch and trunk networks are implemented as seven-layer fully connected neural networks, with 100 neurons in each hidden layer and the hyperbolic tangent (tanh) activation function. The branch network takes the discretized initial condition as input, while the trunk network takes the spatial coordinates as input.

Neural network parameters We perform a grid search over $w_{\text{data}} \in \{1, 5, 10, 20, 50, 100\}$ and $\lambda_{p^2} \in \{0.25, 0.5, 0.75, 1\}$ by minimizing the loss function. The optimal values are found to be $w_{\text{data}} = 20$ and $\lambda_{p^2} = 0.5$. Following the settings in [21], the weights for the physics loss and boundary-condition loss are set to $w_{\text{physics}} = 1$ and $w_{\text{bc}} = 1$, respectively. The batch size for both training and testing is set to 10, and a fixed random seed is used for data generation and model initialization to ensure reproducibility. The same loss weights and coefficients are used for all test cases to keep consistency. The model is trained using a learning rate of 10^{-3} for 10,000 iterations.

Table 2. Comparison of the relative L^2 error $\mathcal{E}_{L^2}$ for different operator-learning models for the one-dimensional Burgers' equation.

Model	DeepONet	POU-DeepONet	PI-DeepONet	PIP^2 Net
$\mathcal{E}_{L^2}$	2.44×10^{-1}	2.05×10^{-1}	1.01×10^{-1}	9.94×10^{-2}

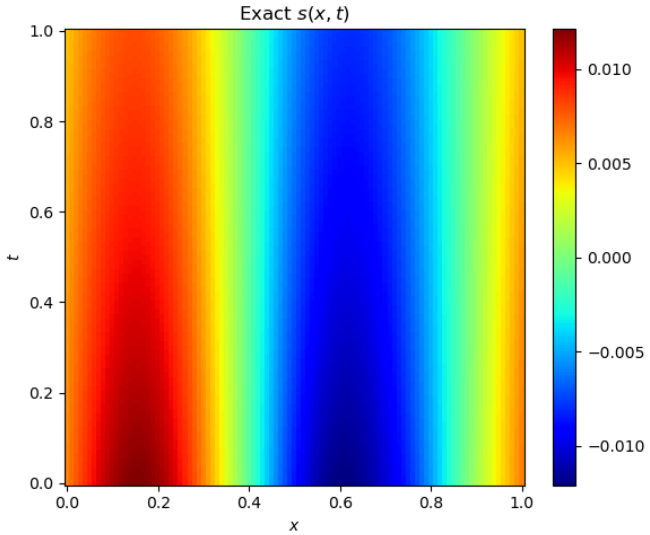


Figure 2. Benchmark solution of Burgers' equation with one initial condition.

Figure 2 presents the benchmark solution of Burgers' equation with one initial condition. Table 2 compares the relative L^2 error $\mathcal{E}_{L^2}$ of different operator-learning models for the one-dimensional Burgers' equation. The baseline DeepONet shows the largest error, indicating limited capability in accurately capturing nonlinear spatiotemporal dynamics using data alone. Introducing a partition-of-unity structure in POU-DeepONet yields a moderate reduction in error, suggesting that localized representations improve approximation quality but remain insufficient without additional physical guidance. Enforcing physics constraints through PI-DeepONet leads to a substantial improvement, reducing the relative error by approximately a factor of two compared to POU-DeepONet. The proposed PIP^2 Net achieves the smallest L^2 error among all models which shows that the combination of physics-informed learning and partition-penalty regularization further improves the results.

Table 3 reports pointwise absolute errors at selected spatial locations at the final time $t = 1$ that provides a localized comparison of model performance. Consistent with Table 2, DeepONet exhibits increasing errors toward the right boundary due to accumulated inaccuracies in nonlinear transport regions. POU-DeepONet reduces pointwise errors over the domain and physics-informed models further improve accuracy, with PI-DeepONet achieving uniformly lower errors at all locations. The proposed PIP^2 Net attains the smallest average error and maintains low pointwise errors, particularly near $x = 1$.

Table 3. Pointwise absolute errors at selected spatial locations x at time $t = 1$ for different operator-learning models on the one-dimensional Burgers' equation.

Model	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1$	Avg. Error
DeepONet	2.47×10^{-3}	5.10×10^{-3}	7.61×10^{-3}	9.93×10^{-3}	1.21×10^{-2}	7.40×10^{-3}
POU-DeepONet	1.58×10^{-3}	1.92×10^{-3}	2.02×10^{-3}	1.92×10^{-3}	1.66×10^{-3}	2.01×10^{-3}
PI-DeepONet	1.00×10^{-4}	4.19×10^{-4}	5.24×10^{-4}	6.81×10^{-4}	9.65×10^{-4}	5.74×10^{-4}
PIP^2 Net	1.86×10^{-4}	3.65×10^{-4}	5.83×10^{-4}	7.67×10^{-4}	8.49×10^{-4}	5.56×10^{-4}

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} = \nu^2\,\frac{\partial^2 u}{\partial x^2}, \qquad (x,t) \in [-\pi,\pi] \times [0,T], \tag{3.3a}$$

$$\text{IC:} \qquad u(x,0) = u_0(x), \qquad x \in [-\pi,\pi], \tag{3.3b}$$

$$\text{BCs:} \qquad u(-\pi,t) = u(\pi,t), \tag{3.3c}$$

$$\frac{\partial u}{\partial x}(-\pi,t) = \frac{\partial u}{\partial x}(\pi,t), \qquad t \in [0,T]. \tag{3.3d}$$

As illustrated in Figure 3, all models capture the solution reasonably well at early time, while discrepancies become larger at later times. Among all models, DeepONet exhibits noticeable phase shifts and amplitude errors, particularly near regions of steep gradients. POU-DeepONet improves local accuracy, but still shows visible deviations near the right boundary. Physics-informed models achieve closer agreement with the benchmark across all time instants, with PI-DeepONet reducing phase and amplitude errors. The proposed PIP^2 Net most accurately reproduces the benchmark solution at all three time instances that are consistent with Tables 2 and 3.

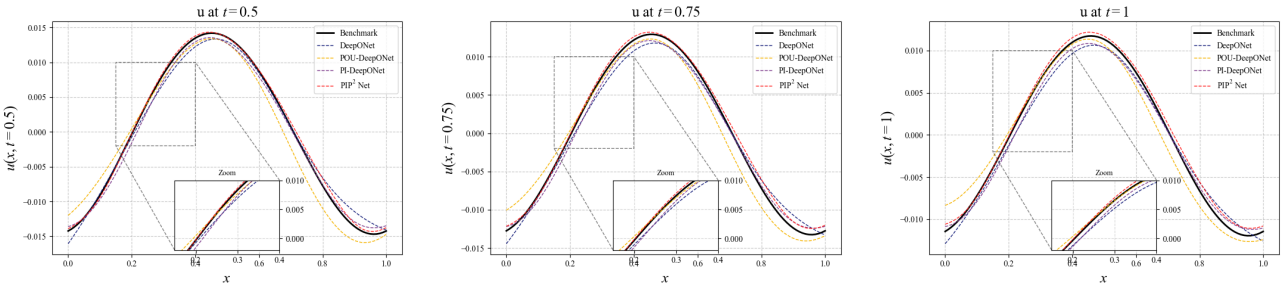


Figure 3. Comparison of the benchmark and model solutions in Table 1 for Burgers' equation at three different time instants: (left) $t = 0.05$, (middle) $t = 0.75$, and (right) $t = 1.0$.

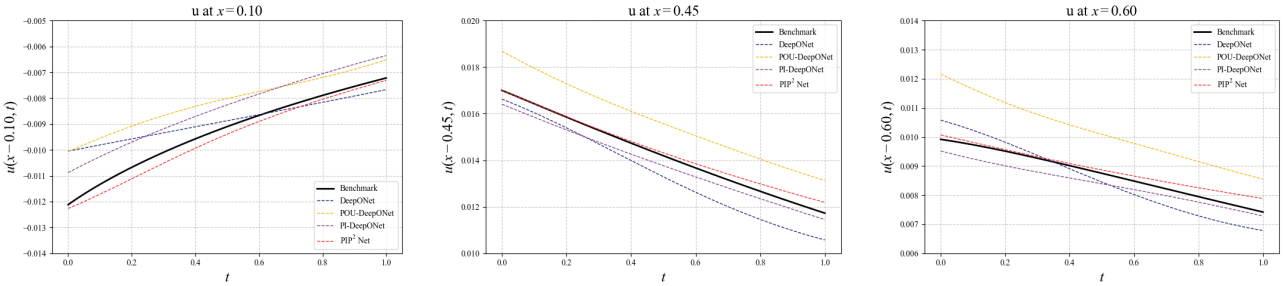
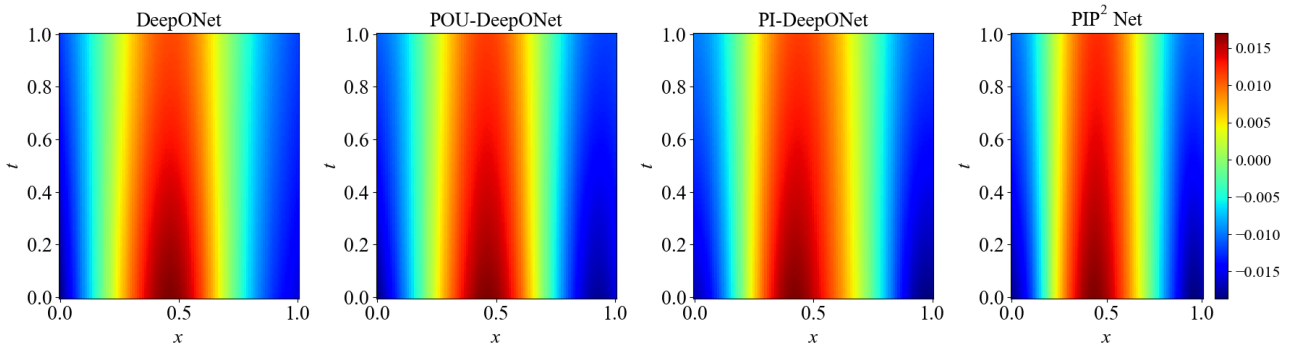


Figure 4. Comparison of the benchmark and model solutions in Table 1 for Burgers' equation at three different locations: $x = 0.1$ (left), $x = 0.45$ (middle), and $x = 0.60$ (right).

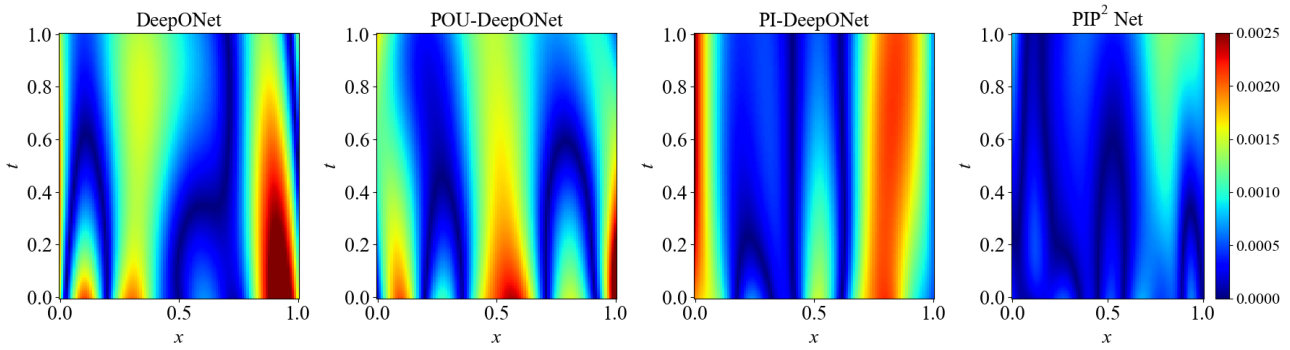
Consistent with the temporal slices in Figure 3, Figure 4 shows that discrepancies among models become more remarkable at locations with step gradient, i.e., $x = 0.45$ and 0.6. While all models perform similarly at $x = 0.1$, DeepONet shows larger temporal error at $x = 0.45$ and $x = 0.60$, and POU-DeepONet shows moderate improvement. Physics-informed models maintain closer agreement with the benchmark across all locations, with PIP^2 Net most accurately matching the reference solution, consistent with Tables 2 and 3.

Figure 5 compares the predicted spatiotemporal fields and corresponding absolute error. DeepONet and POU-DeepONet show noticeable larger spatiotemporal errors particularly near regions with strong gradients. Physics-informed models substantially reduce these errors. Between the two, the proposed PIP^2 Net most accurately captures the spatiotemporal evolution and yields the smallest error level that is consistent with Figures 3 and 4 and Tables 2 and 3.

All models were trained for the same number of epochs on the same computing device. In particular, DeepONet and POU-DeepONet require an average 60 s and 80 s per epoch, respectively. Due to the evaluation of PDE residuals, PI-DeepONet takes about 54 min per epoch, and the PIP^2 takes approximately 60 min.



(a) Comparison of model solutions in Table 1 for the spatiotemporal field of Burgers' equation.



(b) Comparison of the model solutions in Table 1 for the absolute error of Burgers' equation.

Figure 5. Predicted spatiotemporal fields and absolute errors for Burgers' equation obtained from the models summarized in Table 1.

3.2. One-dimensional Allen-Cahn equation

Next,we consider the one-dimensional Allen–Cahn equation, a prototypical model for phase separation and interface dynamics in materials science. Let $u(x,t)$ denote the order parameter, i.e., the local phase state, with prescribed initial condition $u_0(x)$, governed by the following PDE subject to Dirichlet boundary conditions (BCs):

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \frac{1}{c^2} f'(u) \qquad (x,t) \in [-\pi,\pi] \times [0,1], \tag{3.4a}$$

$$\text{IC:} \qquad u(x,0) = u_0(x), \qquad x \in [-\pi,\pi], \tag{3.4b}$$

$$\text{BC:} \qquad u(-\pi,t) = u(\pi,t) = 0, \qquad t \in [0,1]. \tag{3.4c}$$

Here, c is a model parameter and the nonlinear function $f(u)$ is given by the double-well potential

$$f(u) = \frac{(u^2-1)^2}{4}. \tag{3.5}$$

The Allen–Cahn equation models the evolution of an order parameter during phase transitions by minimizing a free-energy functional composed of gradient and bulk energy terms. Similar to the

one-dimensional Burgers equation in Section 3.1, operator learning seeks to approximate the solution operator that maps an initial condition u_0 (and, when applicable, the parameter c) to the corresponding spatiotemporal solution $u(x, t)$ of (3.4). The benchmark solution of the Allen–Cahn equation is shown in Figure 6.

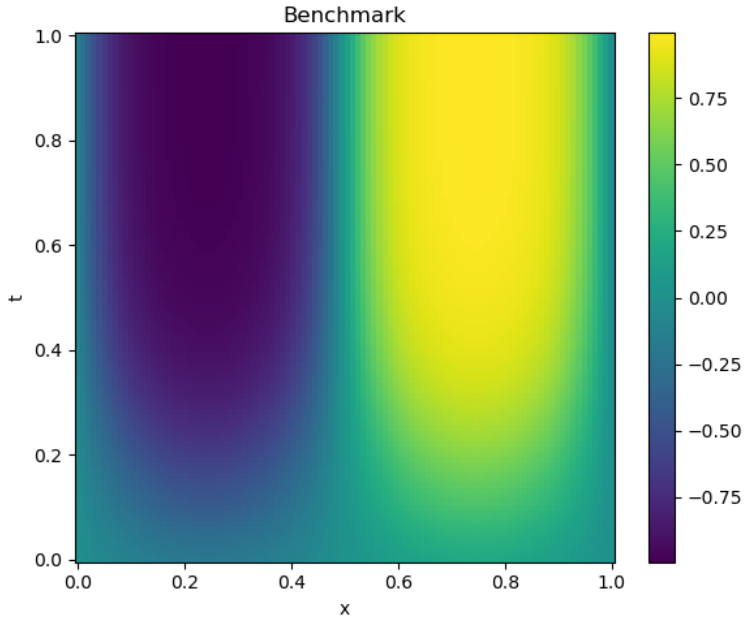


Figure 6. Benchmark solution of Allen-Cahn equation with one initial condition.

Traning and testing data The training and testing datasets are generated by numerically solving the Allen–Cahn equation. The initial condition is prescribed as $u_0(x) = 0.2\sin(x)$, with the parameter ϵ^2 randomly sampled from [0.1, 0.5]. The spatial domain is discretized using 100 uniform grid points, and the temporal interval is resolved with 1000 time steps. An energy-decreasing numerical scheme is employed to obtain the reference solutions. In total, 1000 samples are generated, of which 200 are randomly selected for training.

Neural network architectures Similar to the Burgers' equation test case in Section 3.1, both the branch and trunk networks employ seven-layer fully connected architectures with 100 hidden units per layer and hyperbolic tangent (tanh) activations. The branch network takes the discretized initial condition as input, while the trunk network uses the spatial coordinates as input.

Neural network parameters A grid search over $w_{\text{data}} \in \{1, 5, 10, 20, 50, 100\}$ and $w_{p^2} \in \{0.25, 0.5, 0.75, 1\}$ is performed by minimizing the loss function, which yields the optimal values $w_{\text{data}} = 20$ and $w_{p^2} = 0.5$. The batch size for both training and testing is set to 10, and a fixed random seed is used for model initialization to ensure reproducibility. To make a fair comparison, we choose the same weights for each loss function for all different models. The model is trained using a learning rate of 10^{-3} for 20,000 iterations.

Table 4. Comparison of the relative L^2 error $\mathcal{E}_{L^2}$ for different operator-learning models for the one-dimensional Allen–Cahn equation.

Model	DeepONet	POU-DeepONet	PI-DeepONet	PIP^2 Net
$\mathcal{E}_{L^2}$	8.10×10^{-2}	7.75×10^{-2}	2.54×10^{-2}	1.48×10^{-2}

Table 4 compares the relative L^2 error $\mathcal{E}_{L^2}$ of different operator-learning models for the one-dimensional Allen–Cahn equation. Similar to the Burgers' equation results in Table 2, the baseline DeepONet exhibits the largest error that shows the limitations of purely data-driven operator learning for nonlinear phase-field dynamics. Introducing a partition-of-unity structure in POU-DeepONet yields a modest improvement, while enforcing physics constraints through PI-DeepONet leads to a substantial reduction in error. The proposed PIP^2 Net achieves the smallest L^2 error among all models that also indicates the combined effect of physics-informed learning and partition-penalty regularization further improves the accuracy. Compared to the Burgers' equation, the relative improvement of PIP^2 Net over PI-DeepONet is more remarkable, which is due to the fact that the Allen–Cahn dynamics are more sensitive to interface resolution.

Table 5. Pointwise absolute errors at selected spatial locations x at time $t = 0.9$ for different operator-learning models on the one-dimensional Allen–Cahn equation.

Model	$x = 0$	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1$	Avg. Error
DeepONet	5.89×10^{-3}	7.39×10^{-2}	1.32×10^{-1}	1.24×10^{-1}	1.17×10^{-1}	1.06×10^{-1}	1.11×10^{-1}
POU-DeepONet	1.19×10^{-2}	5.23×10^{-2}	7.93×10^{-2}	9.04×10^{-2}	1.01×10^{-1}	1.17×10^{-1}	9.04×10^{-2}
PI-DeepONet	6.94×10^{-3}	1.07×10^{-2}	7.66×10^{-3}	1.99×10^{-2}	2.20×10^{-2}	2.56×10^{-2}	1.85×10^{-2}
PIP^2 Net	3.68×10^{-3}	8.31×10^{-3}	5.03×10^{-3}	6.35×10^{-3}	6.42×10^{-3}	6.09×10^{-3}	7.17×10^{-3}

Table 5 shows pointwise absolute errors at selected spatial locations at time $t = 0.9$, which provides a localized comparison of model performance. Consistent with the Burgers' equation results in Table 3, DeepONet still shows large pointwise errors, for most spatial locations, particularly away from the boundaries. POU-DeepONet reduces these errors but remains less accurate near regions associated with interface motion. Physics-informed models significantly improve pointwise accuracy, with PI-DeepONet achieving uniformly lower errors across the domain. The proposed PIP^2 Net attains the smallest average error, and maintains consistently low pointwise errors at all spatial locations.

As illustrated in Figure 7, all models capture the qualitative solution behavior at early time, while discrepancies become more remarkable as the phase separation dynamics evolve. Similar to the temporal results observed for the Burgers' equation in Figure 3, DeepONet exhibits strong phase lag and amplitude errors at later times. POU-DeepONet improves local accuracy, but still deviates from the benchmark solution. Physics-informed models achieve closer agreement across all time instants, with PI-DeepONet reducing temporal errors. The proposed PIP^2 Net most accurately reproduces the benchmark solution at all three time instances, consistent with the error trends reported in Tables 4 and 5.

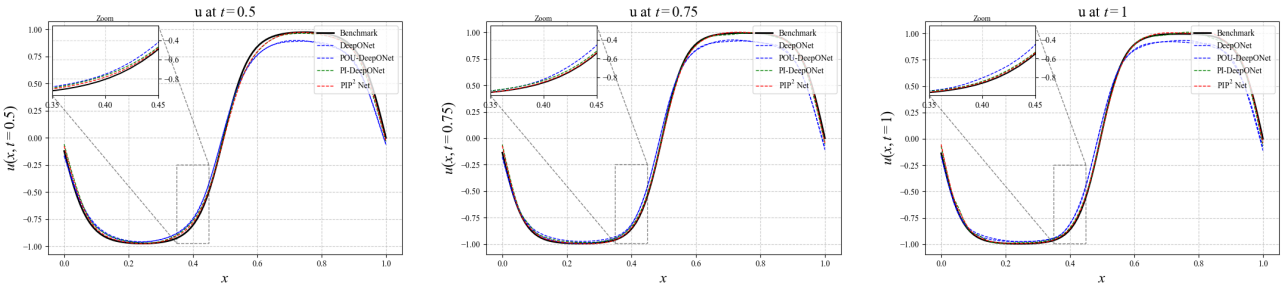


Figure 7. Comparison of the benchmark and model solutions in Table 1 for the Allen-cahn equation at three different time instants: (left) $t = 0.05$, (middle) $t = 0.75$, and (right) $t = 1.0$.

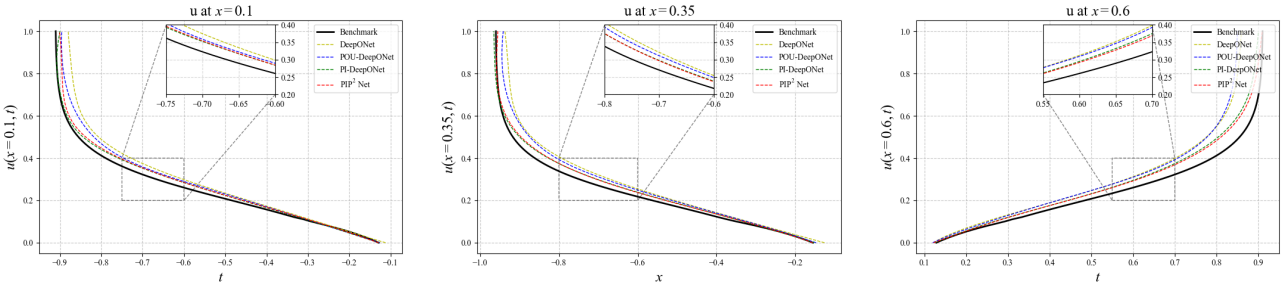
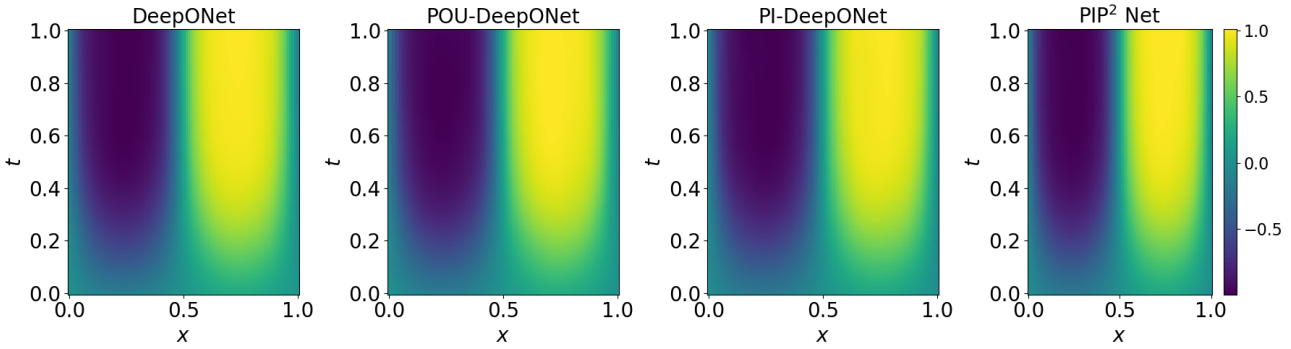


Figure 8. Comparison of the benchmark and model solutions in Table 1 for the Allen-cahn equation at three different locations: $x = 0.1$ (left), $x = 0.35$ (middle), and $x = 0.60$ (right).

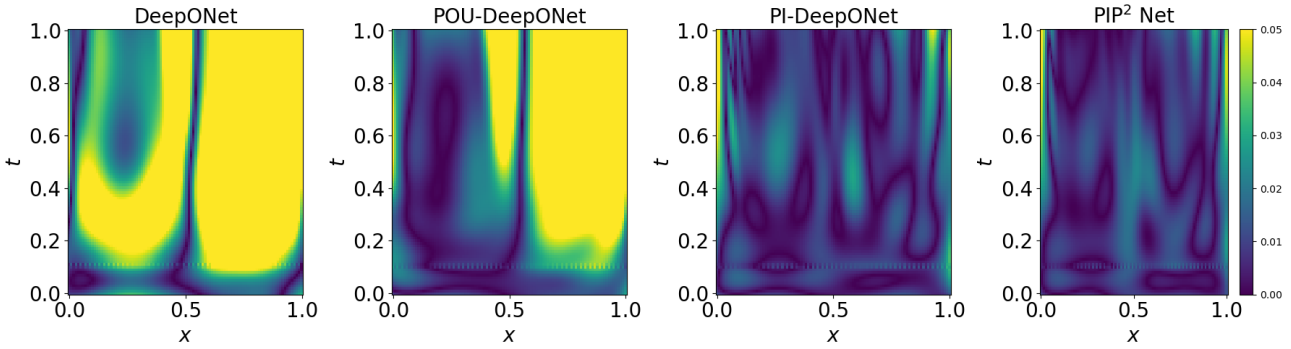
Consistent with the temporal behavior, Figure 8 shows that discrepancies among models become more evident at spatial locations influenced by interfacial dynamics. While all models perform comparably at $x = 0.1$, larger deviations are observed at $x = 0.35$ and $x = 0.60$. Similar to the Burgers' equation case, DeepONet exhibits the largest temporal errors, and POU-DeepONet provides moderate improvement. Physics-informed models maintain closer agreement with the benchmark across all locations, with PIP^2 Net most accurately matching the reference solution, consistent with the pointwise error statistics in Table 5.

Figure 9 compares the predicted spatiotemporal fields and corresponding absolute errors for the Allen–Cahn equation. Similar to the spatiotemporal results for the Burgers' equation in Figure 5, DeepONet and POU-DeepONet exhibit larger spatiotemporal errors, particularly near regions with sharp gradients. Physics-informed models substantially suppress these errors, with PI-DeepONet improving spatiotemporal coherence. Among all methods, the proposed PIP^2 Net most accurately captures the spatiotemporal evolution and yields the smallest error levels.

Remark 3.1. All experiments were performed on a CPU. For the same number of epochs, DeepONet and POU-DeepONet require about 3 min 50 s and 4 min 20 s per epoch, respectively, while PI-DeepONet takes approximately 50 min per epoch and PIP^2 Net takes approximately 52 min.



(a) Comparison of model solutions in Table 1 for the spatiotemporal field of Allen-cahn equation.



(b) Comparison of the model solutions in Table 1 for the absolute error of Allen-cahn equation.

Figure 9. Predicted spatiotemporal fields and absolute errors for Allen-cahn equation obtained from the models summarized in Table 1.

3.3. One-dimensional diffusion-reaction equation

We finally consider the one-dimensional diffusion–reaction equation, a canonical model for coupled material diffusion and chemical reaction processes [41]. Let $s(x,t)$ denote the concentration field with initial condition $s_0(x)$, evolving under diffusion with coefficient D, a quadratic reaction term with rate k, and an external source term $u(x)$. The dynamics are governed by the following PDE subject to Dirichlet boundary conditions (BCs):

$$\frac{\partial s}{\partial t} = D\,\frac{\partial^2 s}{\partial x^2} + k\,s^2 + u(x), \qquad (x,t) \in [0,1]\times[0,1], \tag{3.6a}$$

$$\text{IC:} \qquad s(x,0) = s_0(x), \qquad x \in [0,1], \tag{3.6b}$$

$$\text{BC:} \qquad s(0,t) = s(1,t) = 0, \qquad t \in [0,1]. \tag{3.6c}$$

In this example, homogeneous initial and boundary conditions are imposed, with $s_0(x) = 0$ and $s(0,t) = s(1,t) = 0$. The diffusion coefficient and reaction rate are fixed at $D = 0.01$ and $k = 0.01$, respectively. The operator learning model aims to approximate the solution operator that maps a given source term $u(x)$ to the corresponding concentration field $s(x,t)$ satisfying (3.6). Following [21], the

source terms $u(x)$ are sampled from a Gaussian random field (GRF) with a radial basis function (RBF) covariance kernel,

$$\mathcal{K}(x_1, x_2) = \sigma^2 \exp\left(-\frac{1}{2}\sum_i \left(\frac{x_{1,i} - x_{2,i}}{l_i}\right)^2\right), \tag{3.7}$$

which provides a set of smooth source functions for training and testing. The benchmark solution is shown in Figure 10.

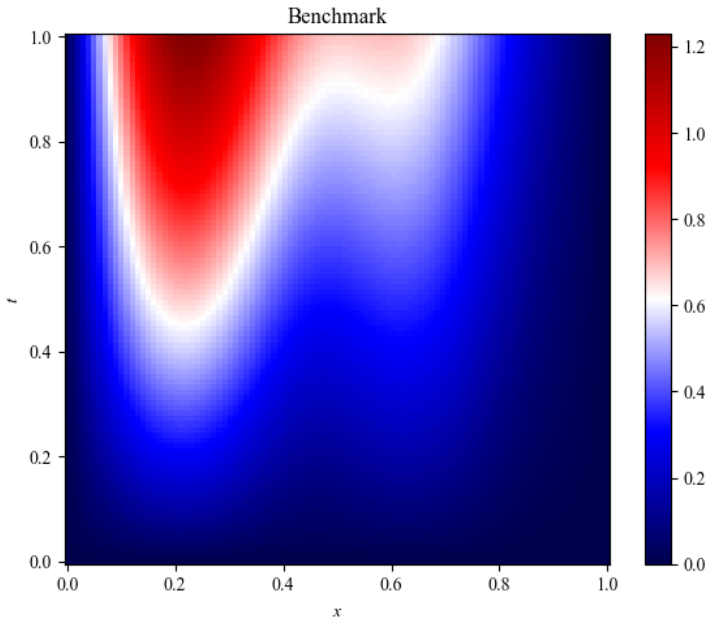


Figure 10. Benchmark solution of diffusion-reaction equation with one initial condition.

Training and tesing data Consistent with the setups in Sections 3.1 and 3.2, for each training sample $u^{(i)}$, we uniformly sample P points $\{(x_{u,j}^{(i)}, t_{u,j}^{(i)})\}_{j=1}^{P}$ from the boundary of $[0,1] \times [0,1]$, excluding the final time $t = 1$, and Q collocation points $\{(x_{r,j}^{(i)}, t_{r,j}^{(i)})\}_{j=1}^{Q}$, where $x_{r,j}^{(i)} = x_j$ and $\{t_{r,j}^{(i)}\}_{j=1}^{Q}$ are uniformly sampled in $[0,1]$. We set $P = Q = 100$, and generate 500 training input functions $u(x)$ by sampling from a Gaussian random field (GRF) with length scale $l = 0.2$ using a spatial resolution of $m = 100$ equidistant points. The test set consists of 50 independently sampled input functions from the same GRF, with reference solutions computed using a second-order implicit finite difference scheme on a 100×100 uniform grid.

Neural network architectures Consistent with the architectures used in Sections 3.1 and 3.2, the branch network takes the input function $u(x)$ and is implemented as a six-layer fully connected neural network with 50 hidden units per layer and hyperbolic tangent (`tanh`) activations. The trunk network uses the spatial coordinates as input and adopts the same six-layer fully connected architecture with identical activation functions.

Neural network parameters Following the same hyperparameter tuning strategy as in Sections 3.1 and 3.2, a grid search over $w_{p^2} \in \{0.05, 0.1, 0.15, 0.2, 0.5, 1\}$ is performed by minimizing the loss

function, yielding the optimal value $w_{p^2} = 0.1$. For both training and testing, the batch size is set to be 1000, and a fixed random seed is chosen for model initialization to ensure reproducibility. To make a fair comparison, we choose the same weights for each loss functions for all different models. In this example, we use a learning rate of 10^{-3}, and train the model for 10,000 iterations.

Table 6. Comparison of the relative L^2 error $\mathcal{E}_{L^2}$ for different operator-learning models for the one-dimensional diffusion–reaction equation.

Model	DeepONet	POU-DeepONet	PI-DeepONet	PIP^2 Net
$\mathcal{E}_{L^2}$	6.65×10^{-2}	5.10×10^{-2}	4.49×10^{-2}	1.56×10^{-4}

Table 6 shows the relative L^2 error $\mathcal{E}_{L^2}$ for different operator-learning models applied to the one-dimensional diffusion–reaction equation. Similar to the Burgers' and Allen–Cahn cases, the baseline DeepONet exhibits the largest error that indicates the limited accuracy when learning the solution operator from data alone. Incorporating a partition-of-unity structure in POU-DeepONet leads to a noticeable reduction in error. Physics-informed learning further enhances performance, with PI-DeepONet achieving additional error reduction. Notably, the proposed PIP^2 Net yields a dramatic improvement, attaining an L^2 error several orders of magnitude smaller than the other models.

Table 7. Pointwise absolute errors at selected spatial locations x at time $t = 1$ for different operator-learning models on the one-dimensional diffusion–reaction equation.

Model	$x = 0$	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1$	Avg. Error
DeepONet	1.63×10^{-2}	2.61×10^{-2}	3.59×10^{-2}	4.49×10^{-1}	5.35×10^{-1}	6.08×10^{-1}	4.75×10^{-1}
POU-DeepONet	1.39×10^{-2}	1.11×10^{-3}	6.36×10^{-3}	6.56×10^{-3}	6.71×10^{-3}	1.10×10^{-2}	9.13×10^{-3}
PI-DeepONet	1.37×10^{-2}	1.20×10^{-3}	1.23×10^{-3}	1.22×10^{-2}	1.06×10^{-2}	7.43×10^{-3}	1.36×10^{-2}
PIP^2 Net	5.82×10^{-3}	7.53×10^{-3}	5.19×10^{-3}	2.46×10^{-3}	1.78×10^{-3}	4.98×10^{-3}	5.55×10^{-3}

Table 7 presents pointwise absolute errors at selected spatial locations at the final time $t = 1$, providing a localized comparison of model accuracy. DeepONet exhibits large pointwise errors, particularly toward the right boundary, where the influence of the source term accumulates over time. POU-DeepONet significantly reduces these errors across most spatial locations, while PI-DeepONet further improves accuracy by enforcing the governing PDE residual. Consistent with the average L^2 error trends in Table 6, the proposed PIP^2 Net achieves the smallest average error and maintains uniformly low pointwise errors throughout the domain. Compared to the Burgers' and Allen–Cahn results, the improvement here is especially large, which further indicates the robustness of PIP^2 Net for source-driven diffusion processes.

As shown in Figure 11, all models capture the overall solution profile at early time, while discrepancies become more evident as the diffusion–reaction dynamics evolve. DeepONet exhibits noticeable amplitude and phase errors, particularly in regions influenced by the source term, which grow over time. POU-DeepONet improves local accuracy, but still deviates from the benchmark near regions of strong curvature. Physics-informed models provide closer agreement across all time instants, with PI-DeepONet reducing both phase and amplitude errors. The proposed PIP^2 Net most

accurately reproduces the benchmark solution at all three time instances, especially in regions highlighted by the zoomed-in views, consistent with the error trends reported in Tables 6 and 7 and analogous to the improvements observed for the Burgers' and Allen–Cahn equation.

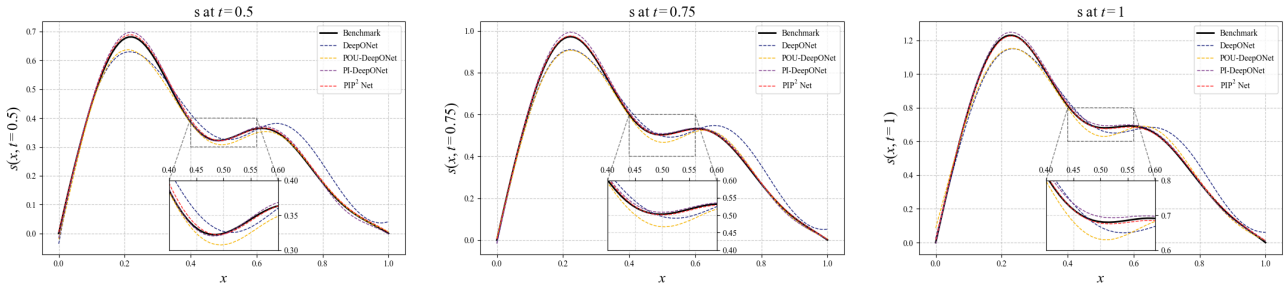


Figure 11. Comparison of the benchmark and model solutions in Table 1 for Diffusion-Reaction equation at three different time instants: (left) $t = 0.50$, (middle) $t = 0.75$, and (right) $t = 1.0$.

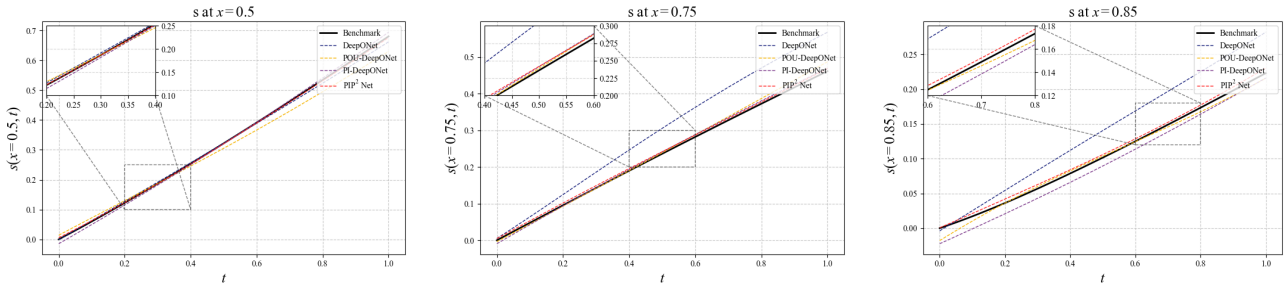
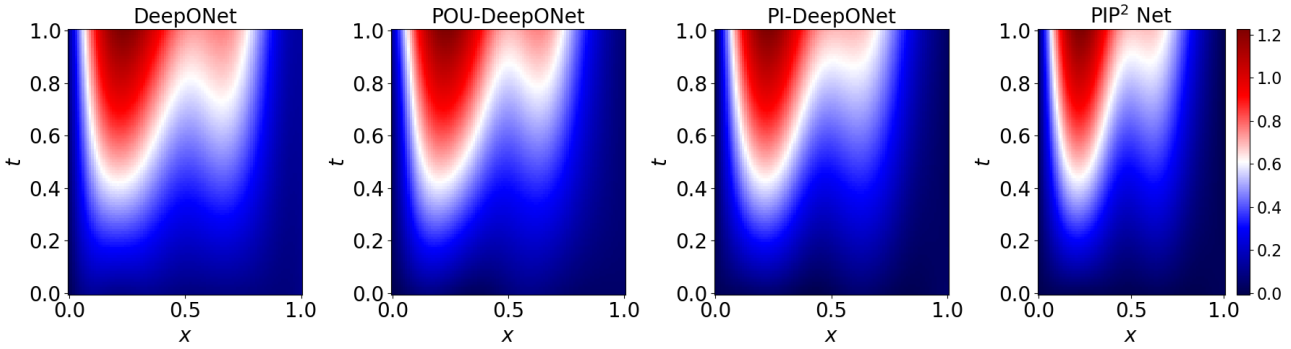


Figure 12. Comparison of the benchmark and model solutions in Table 1 for Diffusion-Reaction equation at three different locations: $x = 0.5$ (left), $x = 0.75$ (middle), and $x = 0.85$ (right).

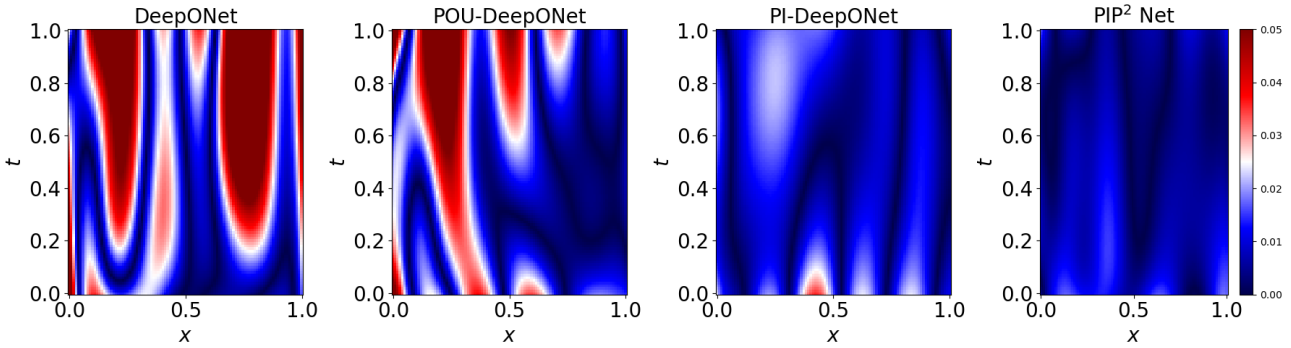
Figure 12 presents the temporal evolution of the solution at fixed spatial locations. At $x = 0.5$, all models closely follow the benchmark solution. Larger discrepancies emerge at $x = 0.75$ and $x = 0.85$, where the influence of the source term and reaction effects becomes more pronounced. DeepONet displays the largest temporal deviations, while POU-DeepONet provides moderate improvement. Physics-informed models maintain closer agreement throughout the time interval, with the proposed PIP^2 Net most accurately tracking the benchmark solution at all locations. This behavior is consistent with the pointwise error trends in Table 7, and consistent with the results for the Burgers' and Allen–Cahn equations.

Figure 13 illustrates the predicted spatiotemporal fields and corresponding absolute error distributions for the diffusion–reaction equation. While all models recover the overall solution structure, DeepONet and POU-DeepONet exhibit substantial spatiotemporal error accumulation, particularly in regions strongly influenced by the source term. Physics-informed models markedly reduces these errors. Between the two, the proposed PIP^2 Net achieves the most accurate spatiotemporal representation and the lowest error levels. This behavior is consistent with the trends

observed in Figures 11 and 12 and Tables 6 and 7, and further corroborates the advantages of PIP^2 Net observed for the Burgers' and Allen–Cahn equations.



(a) Comparison of model solutions in Table 1 for the spatiotemporal field of the diffusion-reaction equation.



(b) Comparison of the model solutions in Table 1 for the absolute error of the diffusion-reaction equation.

Figure 13. Predicted spatiotemporal fields and absolute errors for the diffusion-reaction equation obtained from the models summarized in Table 1.

Table 8. Relative error $\mathcal{E}_{L^2}$ with different partition penalty parameters λ_{P^2}.

λ_{P^2}	0.05	0.10	0.15	0.20	0.50	1.00
$\mathcal{E}_{L^2}$	1.56e-04	3.82e-04	6.13e-04	3.79e-04	2.85e-04	1.86e-04

Table 8 shows the relative L^2 errors with different partition penalty weights λ_{P^2} that ranges from 0.05 to 1 for the diffusion–reaction problem. The results show that for different λ_{P^2} values, all relative errors stay at a similar order of 10^{-4} with only small fluctuations, which indicates that the partition penalty term provides effective regularization.

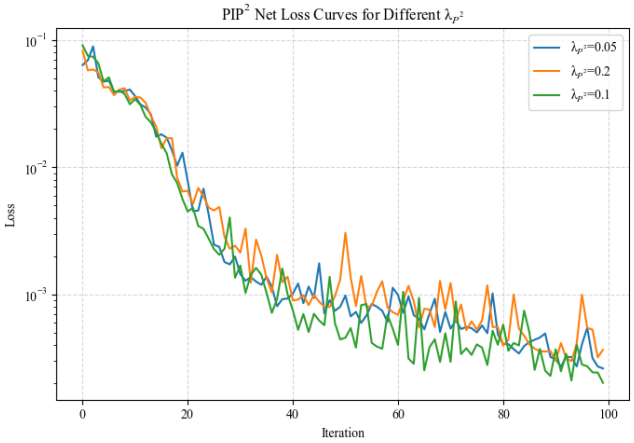


Figure 14. PIP^2 net loss curves for different λ_{P^2}.

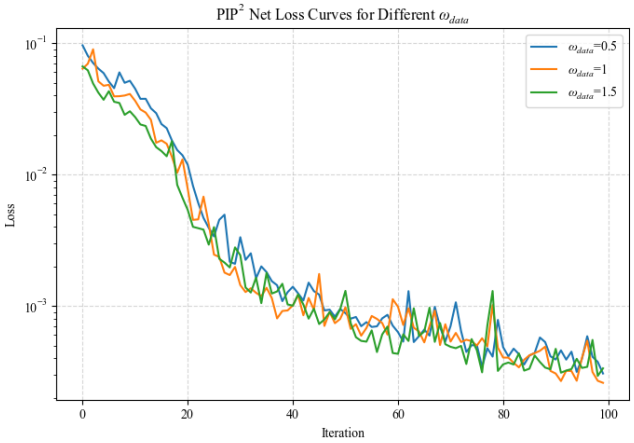


Figure 15. PIP^2 net loss curves for different ω_{data}.

Remark 3.2 (Sensitivity of different loss weights). To assess whether the error reduction achieved by PIP^2-Net in the diffusion–reaction example is sensitive to the choice of loss-weighting hyperparameters (rather than specific to a single setting), we report additional training loss curves with different loss-weight configurations, as shown in Figures 14 and 15. Figure 14 shows that the training loss evolution for different partition penalty strengths λ_{P^2}. With different λ_{P^2} in a broad range, all loss curves exhibit similar convergence behavior and comparable final loss levels. This indicates that the effectiveness of the partition penalty does not rely on a narrowly tuned value, and that the optimization process remains stable under moderate changes of λ_{P^2}. Figure 15 shows the training loss trajectories obtained under varying choices of the data-loss weight ω_{data}. Across all settings, similar to Figure 14, PIP^2 Net exhibits comparable convergence behavior and reaches to similar loss levels, indicating that its empirical performance is largely insensitive to the relative weighting between physics loss and data loss.

Table 9. Mean and standard deviation of trunk output ℓ_2 norms across different models.

Model	DeepONet	POU-DeepONet	PI-DeepONet	PIP^2 Net
Mean ℓ_2 Norm	5.0014	4.4577	8.3968	4.3893
Std. of ℓ_2 Norm	2.7086	1.7250	3.8644	1.3429

Remark 3.3. To analyze how the partition penalty affects the operator learning framework, we compare the ℓ_2 norms of each individual in trunk net for different models. As shown in Table 9, PIP^2 Net exhibits a substantially smaller standard deviation of trunk output norms compared with other models, which indicates less bias in the learned trunk net. This supports the conclusion that the partition penalty mechanism can effectively prevent dominant or vanishing modes that further reduces the mode collapse in the trunk representation.

All experiments were performed on a CPU. For the same number of epochs, DeepONet and POU-DeepONet require about 4 min 30 s and 5 min 10 s per epoch, respectively. The training time of PI-DeepONet is approximately 6 min per epoch, while PIP^2 Net takes about 6 min 15 s.

3.4. Two-dimensional Eikonal equation

We finally consider a two-dimensional Eikonal equation, which is a canonical nonlinear first-order Hamilton–Jacobi equation and is widely used to model wavefront propagation, optimal control, and distance functions [21, 28]. Let $s(x)$ denote the scalar field defined on a spatial domain $\Omega \subset \mathbb{R}^2$, where $x = (x, y)$ represents the two-dimensional spatial coordinates. The Eikonal equation is given by

$$\|\nabla s(x)\|_2 = 1, \qquad x \in \Omega, \tag{3.8a}$$

$$\text{BC:} \qquad s(x) = 0, \qquad x \in \partial\Omega, \tag{3.8b}$$

where Ω is an open domain with a piecewise smooth boundary $\partial\Omega$.

A solution to (3.8) is the signed distance function to the boundary $\partial\Omega$, which quantifies the shortest distance from a point x to the boundary. Specifically, the solution can be expressed as

$$s(x) = \begin{cases} d(x, \partial\Omega), & x \in \Omega, \\ -d(x, \partial\Omega), & x \in \Omega^c, \end{cases} \tag{3.9}$$

where Ω^c denotes the complement of Ω.

The distance function $d(x, \partial\Omega)$ is defined as

$$d(x, \partial\Omega) := \inf_{y \in \partial\Omega} \|x - y\|_2. \tag{3.10}$$

From an operator-learning perspective, the Eikonal problem can be viewed as learning a nonlinear operator that maps the domain geometry (or an implicit boundary representation) to the corresponding signed distance function $s(x)$. Unlike time-dependent evolution equations such as Burgers' equation, the Eikonal equation represents a steady-state nonlinear PDE with strong geometric structure that poses different challenges compared to other test cases for neural operator models.

Training and testing data For each input curve $\Gamma^{(i)}$, training points are sampled as follows. The curve $\Gamma^{(i)}$ is represented by a set of fixed sensor locations $\{(x_j^{(i)}, y_j^{(i)})\}_{j=1}^m$ obtained from a parametric description of the boundary. At these boundary sensor points, the Dirichlet boundary condition $s(x_j^{(i)}, y_j^{(i)}) = 0$ is enforced. In addition, a set of interior collocation points $\{(x_{r,j}^{(i)}, y_{r,j}^{(i)})\}_{j=1}^Q$ is uniformly sampled within the computational domain $D = [-2, 2] \times [-2, 2]$ to evaluate the PDE residual of the Eikonal equation. The residual points are used to enforce the physics constraint $\|\nabla s(x, y)\|_2 = 1$. A total of $N = 2000$ input curves are randomly generated. In the case of parametric circles, each curve corresponds to a circle with radius $r^{(i)}$ sampled from a uniform distribution. From the generated dataset, 2000 input curves are selected for training, while the remaining 100 curves are reserved for testing.

Neural network architectures Following the network architectures adopted in Sections 3.1 and 3.2, the branch network is designed to take the input function $u(x)$ as its input and consists of a six-layer fully connected neural network, with 50 hidden units per layer and hyperbolic tangent (`tanh`) activation functions. The trunk network takes the spatial coordinates as input and is implemented as a seven-layer fully connected neural network, also with 50 hidden units per layer and the same activation functions.

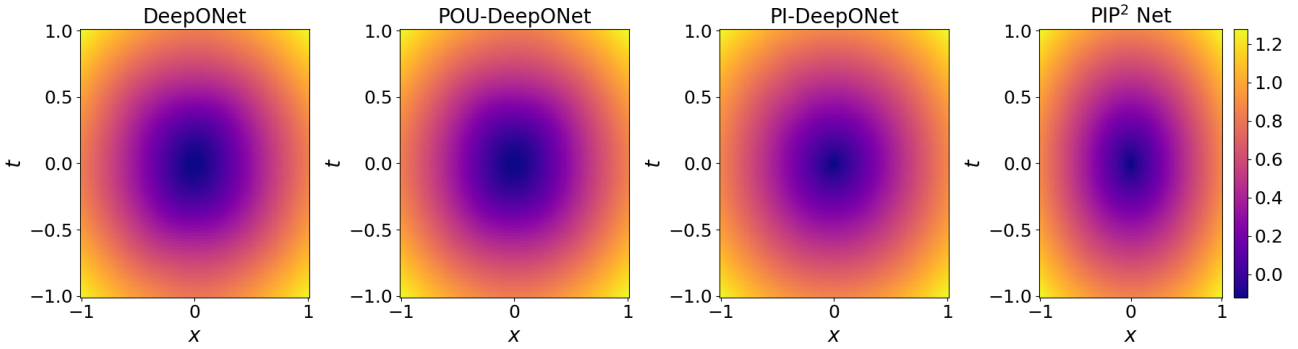
Neural network parameters Following the same hyperparameter tuning strategy as in Sections 3.1 and 3.2, a grid search over $w_{p^2} \in \{0.05, 0.1, 0.15, 0.2, 0.5, 1\}$ is performed by minimizing the loss function, yielding the optimal value $w_{p^2} = 0.5$. The batch size for both training and testing is set to 10,000, and a fixed random seed is used for data generation and model initialization to ensure reproducibility. The same loss weights and coefficients are used across all test cases for consistency. In this example, we use a learning rate of 10^{-3}, and train the model for 15,000 iterations.

Table 10 shows the relative L^2 error $\mathcal{E}_{L^2}$ for different operator-learning models applied to the two-dimensional Eikonal equation. Similar to the Burgers' equation, the Allen–Cahn equation, and the diffusion-reaction equation cases in previous sections, the baseline DeepONet exhibits the largest error with limited accuracy when learning the solution operator from data alone. Incorporating a partition-of-unity structure in POU-DeepONet leads to a slight reduction in error. Physics-informed learning, on the other hand, improves significantly performance, with PI-DeepONet achieving additional error reduction. Notably, the proposed PIP^2 Net is the best, reducing the L^2 error by more than 50% relative to DeepONet. Figure 16 further confirms this result.

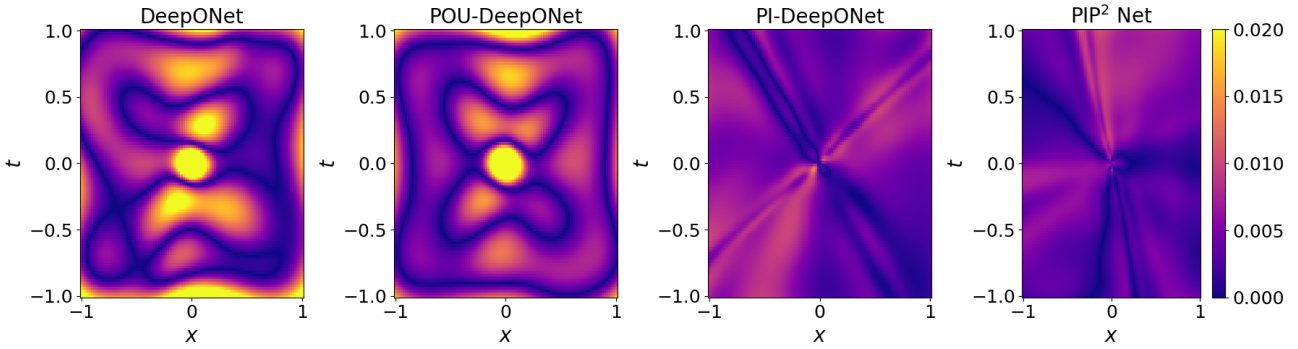
Table 10. Comparison of the relative L^2 error $\mathcal{E}_{L^2}$ for different operator-learning models for the two-dimensional Eikonal equation.

Model	DeepONet	POU-DeepONet	PI-DeepONet	PIP^2 Net
$\mathcal{E}_{L^2}$	2.41×10^{-2}	2.20×10^{-2}	1.50×10^{-2}	1.01×10^{-2}

For this case, all experiments were conducted on a GPU. DeepONet and POU-DeepONet require approximately 40 s and 1 min 15 s per epoch, respectively. PI-DeepONet takes about 3 min 40 s per epoch, and the PIP^2 Net increases the training time to approximately 5 min 15 s per epoch.



(a) Comparison of model solutions in Table 1 for the spatiotemporal field of Eikonal equation.



(b) Comparison of the model solutions in Table 1 for the absolute error of Eikonal equation.

Figure 16. Predicted spatiotemporal fields and absolute errors for Burgers' equation obtained from the models summarized in Table 1.

4. Conclusions and future works

In this work, we propose a physics-informed operator-learning framework, termed PIP^2 Net, which incorporates a partition penalty (P^2) mechanism into the physics-informed DeepONet architecture. The proposed framework tends to improve the structural instability of trunk-network basis functions that can occur in existing operator-learning models, and may result in mode imbalance or collapse [42]. The partition penalty refers to a partition-of-unity (PoU)–inspired penalty regularization [32, 37], which encourages well-conditioned and coordinated trunk-network outputs, that further enhances the stability of operator approximation. We evaluate PIP^2 Net on three representative nonlinear partial differential equations, i.e., the Burgers equation, the Allen–Cahn equation, and the diffusion–reaction equation [1, 2]. Comparisons with DeepONet, PI-DeepONet, and POU-DeepONet [31, 39] show that PIP^2 Net consistently achieves lower L^2 errors for all test cases.

In the future, we plan to investigate several promising research directions. First, we will extend the proposed PIP^2-Net to substantially more complex physical systems, such as the Navier–Stokes equations (NSEs) [43, 44] and the Boussinesq equations [45–47], where multiscale interactions and chaotic dynamics pose significant challenges for existing operator-learning frameworks. Another important direction is the integration of the P^2 mechanism into Fourier neural operator (FNO)

architectures [15] to develop PIP^2-based FNO models with improved spectral stability and improved ability to capture high-frequency dynamics. Additional directions include geometry-aware extensions for problems defined on irregular or parameterized domains, coupling PIP^2 Net with data assimilation techniques [48–50] to enable real-time digital twins in geophysical modeling [51, 52] and in disease modeling [53, 54], developing probabilistic learning framework [55, 56] for uncertainty quantification in sparse observation settings, and deriving theoretical analysis to better understand how the P^2 regularization improves conditioning, generalization, and operator approximation.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Data and code availability

The implementation of the proposed method is available at https://github.com/Hongjin-Mi/PIP2-Net. Data supporting the findings of this work can be obtained from the corresponding author upon reasonable request.

Acknowledgments

Yeyu Zhang is grateful to acknowledge the support of the National Natural Science Foundation of China (NSFC) under Grants No. 12401562, No. 12571459, and No. 12241103.

Conflict of interest

Changhong Mou is the guest editor for [Electronic Research Archive] and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

References

1. L. C. Evans, *Partial Differential Equations*, American Mathematical Society, 2022.
2. S. J. Farlow, *Partial Differential Equations for Scientists and Engineers*, Courier Corporation, 1993.
3. R. Courant, K. Friedrichs, H. Lewy, On the partial difference equations of mathematical physics, *IBM J. Res. Dev.*, **11** (1967), 215–234. https://doi.org/10.1147/rd.112.0215
4. L. C. Berselli, T. Iliescu, W. J. Layton, *Mathematics of Large Eddy Simulation of Turbulent Flows*, Springer, 2006. https://doi.org/10.1007/b137408
5. J. S. Hesthaven, G. Rozza, B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, Springer, 2016. https://doi.org/10.1007/978-3-319-22470-1
6. W. Layton, *Introduction to the Numerical Analysis of Incompressible Viscous Flows*, SIAM, 2008. https://doi.org/10.1137/1.9780898718904

7. T. J. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Courier Corporation, 2003.
8. R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, SIAM, 2007. https://doi.org/10.1137/1.9780898717839
9. S. Yin, Z. Xiang, Adaptive collision avoidance strategy for usvs in perception-limited environments using dynamic priority guidance, *Adv. Eng. Inf.*, **63** (2025), 103355. https://doi.org/10.1016/j.aei.2025.103355
10. S. Yin, Z. Xiang, Multi-objective collaborative path planning for heterogeneous autonomous underwater vehicles in cluttered environments, *Swarm Evol. Comput.*, **100** (2026), 102251. https://doi.org/10.1016/j.swevo.2025.102251
11. S. Yin, Z. Xiang, A hyper-heuristic algorithm via proximal policy optimization for multi-objective truss problems, *Expert Syst. Appl.*, **256** (2024), 124929. https://doi.org/10.1016/j.eswa.2024.124929
12. C. Mou, S. N. Stechmann, N. Chen, Simulation and data assimilation in an idealized coupled atmosphere–ocean–sea ice floe model with cloud effects, *Nonlinear Processes Geophys.*, **32** (2025), 329–351. https://doi.org/10.5194/npg-32-329-2025
13. C. O. de Burgh-Day, T. Leeuwenburg, Machine learning for numerical weather and climate modelling: A review, *Geosci. Model Dev.*, **16** (2023), 6433–6477. https://doi.org/10.5194/gmd-16-6433-2023
14. K. Kashinath, M. Mustafa, A. Albert, J. Wu, C. Jiang, S. Esmaeilzadeh, et al., Physics-informed machine learning: Case studies for weather and climate modelling, *Phil. Trans. R. Soc. A*, **379** (2021), 20200093. https://doi.org/10.1098/rsta.2020.0093
15. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, et al., Fourier neural operator for parametric partial differential equations, preprint, arXiv:2010.08895.
16. 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** (2021), 218–229. https://doi.org/10.1038/s42256-021-00302-5
17. N. B. Kovachki, S. Lanthaler, A. M. Stuart, Operator learning: Algorithms and analysis, in *Handbook of Numerical Analysis*, Elsevier, **25** (2024), 419–467. https://doi.org/10.1016/bs.hna.2024.05.009
18. N. Boullé, A. Townsend, A mathematical guide to operator learning, in *Handbook of Numerical Analysis*, Elsevier, **25** (2024), 83–125. https://doi.org/10.1016/bs.hna.2024.05.003
19. 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 Networks*, **6** (1995), 911–917. https://doi.org/10.1109/72.392253
20. 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** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

21. S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed deeponets, *Sci. Adv.*, **7** (2021), eabi8605. https://doi.org/10.1126/sciadv.abi8605
22. S. Wang, S. Sankaran, P. Perdikaris, Respecting causality is all you need for training physics-informed neural networks, preprint, arXiv:2203.07404.
23. B. Lu, C. Moya, G. Lin, NSGA-PINN: A multi-objective optimization method for physics-informed neural network training, *Algorithms*, **16** (2023), 194. https://doi.org/10.3390/a16040194
24. B. Lu, C. Mou, G. Lin, Morephy-Net: An evolutionary multi-objective optimization for replica-exchange-based physics-informed operator learning network, preprint, arXiv:2509.00663.
25. L. Liu, W. Cai, Multiscale deeponet for nonlinear operators in oscillatory function spaces for building seismic wave responses, preprint, arXiv:2111.04860.
26. J. Chen, H. Yu, B. Li, H. Zhang, X. Jin, S. Meng, et al., DeepONet-embedded physics-informed neural network for production prediction of multiscale shale matrix–fracture system, *Phys. Fluids*, **37**, (2025), 016608. https://doi.org/10.1063/5.0245212
27. B. Yang, X. Li, J. Zhao, Y. Jiang, DD-DeepONet: Domain decomposition and deeponet for solving partial differential equations in three application scenarios, preprint, arXiv:2508.02717.
28. C. Mou, Y. Zhang, X. Zhu, Q. Zhuang, PAS-Net: Physics-informed adaptive scale deep operator network, preprint, arXiv:2511.14925.
29. J. M. Melenk, I. Babuška, The partition of unity finite element method: Basic theory and applications, *Comput. Methods Appl. Mech. Eng.*, **139** (1996), 289–314. https://doi.org/10.1016/S0045-7825(96)01087-0
30. E. Larsson, V. Shcherbakov, A. Heryudono, A least squares radial basis function partition of unity method for solving PDEs, *SIAM J. Sci. Comput.*, **39** (2017), A2538–A2563. https://doi.org/10.1137/17M1118087
31. V. Shcherbakov, E. Larsson, Radial basis function partition of unity methods for pricing vanilla basket options, *Comput. Math. Appl.*, **71** (2016), 185–200. https://doi.org/10.1016/j.camwa.2015.11.007
32. A. Heryudono, E. Larsson, A. Ramage, L. von Sydow, Preconditioning for radial basis function partition of unity methods, *J. Sci. Comput.*, **67** (2016), 1089–1109. https://doi.org/10.1007/s10915-015-0120-6
33. A. Safdari-Vaighani, A. Heryudono, E. Larsson, A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications, *J. Sci. Comput.*, **64** (2015), 341–367. https://doi.org/10.1007/s10915-014-9935-9
34. V. Shankar, G. B. Wright, Mesh-free semi-Lagrangian methods for transport on a sphere using radial basis functions, *J. Comput. Phys.*, **366** (2018), 170–190. https://doi.org/10.1016/j.jcp.2018.04.007
35. T. Fan, N. Trask, M. D'Elia, E. Darve, Probabilistic partition of unity networks for high-dimensional regression problems, *Int. J. Numer. Meth. Eng.*, **124** (2023), 2215–2236. https://doi.org/10.1002/nme.7207

36. S. Goswami, A. D. Jagtap, H. Babaee, B. T. Susi, G. E. Karniadakis, Learning stiff chemical kinetics using extended deep neural operators, *Comput. Methods Appl. Mech. Eng.*, **419** (2024), 116674. https://doi.org/10.1016/j.cma.2023.116674

37. L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, *SIAM Rev.*, **63** (2021), 208–228. https://doi.org/10.1137/19M1274067

38. G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nat. Rev. Phys.*, **3** (2021), 422–440. https://doi.org/10.1038/s42254-021-00314-5

39. B. Lu, C. Mou, G. Lin, MoPINNEnKF: Iterative model inference using generic-pinn-based ensemble kalman filter, preprint, arXiv:2506.00731v1.

40. T. A. Driscoll, N. Hale, L. N. Trefethen, Chebfun Guide, 2014. Available from: https://www.chebfun.org/docs/guide/chebfun_guide.pdf.

41. R. D. Reitz, A study of numerical methods for reaction-diffusion equations, *SIAM J. Sci. Stat. Comput.*, **2** (1981), 95–106. https://doi.org/10.1137/0902008

42. N. B. Kovachki, S. Lanthaler, H. Mhaskar, Data complexity estimates for operator learning, preprint, arXiv:2405.15992.

43. C. Mou, B. Koc, O. San, L. G. Rebholz, T. Iliescu, Data-driven variational multiscale reduced order models, *Comput. Methods Appl. Mech. Eng.*, **373** (2021), 113470. https://doi.org/10.1016/j.cma.2020.113470

44. C. Mou, E. Merzari, O. San, T. Iliescu, An energy-based lengthscale for reduced order models of turbulent flows, *Nucl. Eng. Des.*, **412** (2023), 112454. https://doi.org/10.1016/j.nucengdes.2023.112454

45. Y. Zhang, L. M. Smith, S. N. Stechmann, Convergence to precipitating quasi-geostrophic equations with phase changes: Asymptotics and numerical assessment, *Phil. Trans. R. Soc. A*, **380** (2022), 20210030. https://doi.org/10.1098/rsta.2021.0030

46. Y. Zhang, L. M. Smith, S. N. Stechmann, Fast-wave averaging with phase changes: Asymptotics and application to moist atmospheric dynamics, *J. Nonlinear Sci.*, **31** (2021), 38. https://doi.org/10.1007/s00332-021-09697-2

47. Y. Zhang, L. M. Smith, S. N. Stechmann, Effects of clouds and phase changes on fast-wave averaging: A numerical assessment, *J. Fluid Mech.*, **920** (2021), A49. https://doi.org/10.1017/jfm.2021.427

48. C. Mou, L. M. Smith, N. Chen, Combining stochastic parameterized reduced-order models with machine learning for data assimilation and uncertainty quantification with partial observations, *J. Adv. Model. Earth Syst.*, **15** (2023), e2022MS003597. https://doi.org/10.1029/2022MS003597

49. N. Chen, A. J. Majda, Efficient nonlinear optimal smoothing and sampling algorithms for complex turbulent nonlinear dynamical systems with partial observations, *J. Comput. Phys.*, **410** (2020), 109381. https://doi.org/10.1016/j.jcp.2020.109381

50. C. Mou, H. Liu, D. R. Wells, T. Iliescu, Data-driven correction reduced order models for the quasi-geostrophic equations: A numerical investigation, *Int. J. Comput. Fluid Dyn.*, **34** (2020), 147–159. https://doi.org/10.1080/10618562.2020.1723556

51. A. Rasheed, O. San, T. Kvamsdal, Digital twin: Values, challenges and enablers from a modeling perspective, *IEEE Access*, **8** (2020), 21980–22012. https://doi.org/10.1109/ACCESS.2020.2970143

52. Q. Chen, Z. Xu, J. Zhang, D. Xiu, Targeted digital twin via flow map learning and its application to fluid dynamics, preprint, arXiv:2510.07549.

53. H. Pan, Y. Zhao, H. Wang, X. Li, E. Leung, F. Chen, et al., Influencing factors of barthel index scores among the community-dwelling elderly in hong kong: A random intercept model, *BMC Geriatr.*, **21** (2021), 484. https://doi.org/10.1186/s12877-021-02422-4

54. S. S. Wu, H. Pan, R. C. Sheldrick, J. Shao, X. M. Liu, S. S. Zheng, et al., Development and validation of the parent-reported indicator of developmental evaluation for chinese children (pride) tool, *World J. Pediatr.*, **21** (2025), 183–191. https://doi.org/10.1007/s12519-025-00878-7

55. N. Chen, C. Mou, L. M. Smith, Y. Zhang, A stochastic precipitating quasi-geostrophic model, *Phys. Fluids*, **36** (2024), 116618. https://doi.org/10.1063/5.0231366

56. Y. Chen, D. Xiu, Learning stochastic dynamical system via flow map operator, *J. Comput. Phys.*, **508** (2024), 112984. https://doi.org/10.1016/j.jcp.2024.112984

AIMS Press