DONIS: Importance Sampling for Training Physics-Informed DeepONet

Shudong Huang[1,2], **Rui Huang**[1,2], **Ming Hu**[1,2], **Wentao Feng**[1,2*] and **Jiancheng Lv**[1,2]

[1]College of Computer Science, Sichuan University, Chengdu 610065, China

[2]Engineering Research Center of Machine Learning and Industry Intelligence, Ministry of Education, Chengdu 610065, China

huangsd@scu.edu.cn, {huangr, huming1}@stu.scu.edu.cn, {wtfeng2021, lvjiancheng}@scu.edu.cn,

Abstract

Deep Operator Network (DeepONet) effectively learns complex operator mappings, especially for systems governed by differential equations. Physics-informed DeepONet (PI-DeepONet) extends these capabilities by integrating physical constraints, enabling robust performance with limited or no labeled data. However, combining operator learning with these constraints increases computational complexity, which makes training more difficult and convergence slower, particularly for nonlinear or high-dimensional problems. In this work, we present an enhanced PI-DeepONet framework, that applies importance sampling to both of DeepONet inputs (i.e., the functions and the collocation points) to alleviate these training challenges. By focusing on critical data regions in both input domains, our approach showcases accelerated convergence and improved accuracy across various complex applications.

1 Introduction

Solving partial differential equations (PDEs) is critical for describing and predicting natural phenomena such as fluid dynamics, heat conduction, and wave propagation. It also plays a key role in optimizing engineering designs, simulating complex systems, and advancing mathematical theories. However, for most PDE problems, obtaining analytical solutions is either infeasible or exceedingly difficult. Numerical methods, such as finite difference and finite element methods, offer efficient and approximate solutions for practical problems, providing effective approaches to addressing complex systems. Nevertheless, traditional numerical methods often rely on fine mesh discretization to enhance accuracy, resulting in exponentially growing computational costs, especially for high-dimensional problems.

In recent years, numerous methods leveraging deep learning have emerged to solve partial differential equations (PDEs) [Huang *et al.*, 2022], aiming to circumvent some of the limitations of traditional numerical approaches by exploiting the fitting capabilities of neural networks. Among these, Physics-Informed Neural Networks (PINNs) [Raissi *et al.*, 2019] integrate initial conditions, boundary conditions, and equation residuals into the loss function, thereby incorporating physical prior knowledge to supervise the neural network in approximating PDE solutions.

Another line of research focuses on training neural networks to approximate operators, referred to as *neural operators*, enabling a single model to solve a class of PDEs. Examples include DeepONet [Lu *et al.*, 2021a], Fourier Neural Operator (FNO) [Li *et al.*, 2020] and further developments [Jin *et al.*, 2022; He *et al.*, 2024; Kumar *et al.*, 2025; Wen *et al.*, 2022; Li *et al.*, 2023; Bonev *et al.*, 2023]. The operator-fitting approach inherently resembles parallel fitting of a class of functions, posing greater challenges in terms of labeled data requirements compared to PINNs. To address this issue, subsequent studies have integrated operator learning with physical constraints to mitigate the reliance on costly labeled data, as demonstrated in the framework of physics-informed DeepONets (PI-DeepONets) [Wang *et al.*, 2021] and later studies [Goswami *et al.*, 2022; Howard *et al.*, 2023; Jiao *et al.*, 2024; Li *et al.*, 2024].

Building on the integration of operator learning with physical constraints, further advancements have explored scenarios where no labeled data is available, enabling fully self-supervised operator learning. Existing research on fully self-supervised PI-DeepONet can be broadly categorized into two main approaches. The first focuses on decomposing the spatiotemporal domain of the solution, reducing the complexity of the operator by employing multi-step solving techniques [Xu *et al.*, 2023; Wang and Perdikaris, 2023]. The second approach addresses the challenge of multiple tasks in loss function. By incorporating initial and boundary condition constraints directly into the output layer or through post-processing, approaches in [Brecht *et al.*, 2023] avoids the imbalance among different terms in the loss function. While the aforementioned methods mitigate certain training challenges associated with PI-DeepONet, their reliance on specific PDE scenarios significantly restricts their generality and broader applicability.

Meanwhile, importance sampling has been adopted to accelerate the training of PINNs, owing to its ease of implementation and independence from the specific form of the

*Corresponding author.

[1]Code is available at https://github.com/ruihuang-1/donis.

PDEs. The work by [Nabian *et al.*, 2021] incorporates importance sampling into the collocation point selection process of PINNs, and further enhances efficiency through a piecewise constant approximation. Subsequent work of the DMIS framework [Yang *et al.*, 2023] introduces a dynamic triangulation strategy to optimize the accuracy of importance estimation, achieving an effective balance between speed and precision.

Motivated by these findings, we propose a more practical and versatile approach to accelerate the training of PI-DeepONet: **PI-DeepONet with Importance Sampling (DONIS)**. By harnessing the architectural flexibility of DeepONet, DONIS introduces a two-step importance sampling framework that sequentially applies importance sampling to the function and collocation point inputs of DeepONet. This strategy prioritizes mini-batch samples with greater influence on the learning objective, enabling faster convergence and enhanced predictive accuracy.

Our main contributions can be summarized as follows:

- An easily implementable importance sampling method for input functions in PI-DeepONet.
- A gradient-based importance estimation and sampling strategy for collocation points in PI-DeepONet.
- Extensive experiments across multiple widely used benchmark physical scenarios, demonstrating that our method significantly improves the convergence speed and prediction accuracy of PI-DeepONet.

2 Background

2.1 Deep Operator Network

Deep Operator Network (DeepONet) provides a neural network framework for learning operators, which are mappings between function spaces. Unlike traditional neural networks that map vectors to vectors, DeepONet learns the functional relationship between an input function $f(z)$ and an output function $u(x)$, as defined by an operator $\mathcal{G}$:

$$u(x) = \mathcal{G}(f)(x). \tag{1}$$

DeepONet achieves operator learning by decomposing the task into two sub-networks: the *branch network* and the *trunk network*. These networks work in tandem to approximate the target function.

- The branch network encodes the input function $f(z)$ by sampling it at fixed sensor locations $\{z_1, z_2, \ldots, z_m\}$. These samples are passed through a neural network $\mathrm{BranchNet}$, producing latent outputs $\mathbf{b} = [b_1, b_2, \ldots, b_p]$.
- The trunk network encodes the target evaluation location x via a neural network $\mathrm{TrunkNet}$, producing latent outputs $\mathbf{t} = [t_1, t_2, \ldots, t_p]$.

The final output of DeepONet is then computed as a linear combination of the branch and trunk outputs:

$$u(x) \approx \sum_{k=1}^{p} b_k t_k, \tag{2}$$

where p is the number of latent modes. This formulation allows DeepONet to capture the functional relationship encoded by the operator $\mathcal{G}$.

The supervised training of DeepONet minimizes the loss function:

$$\mathcal{L}_{\text{sup}} = \frac{1}{N} \sum_{i=1}^{N} \|u_i(x) - \mathcal{G}(f_i)(x)\|^2, \tag{3}$$

where $\{f_i, \mathcal{G}(f_i)\}$ are labeled input-output pairs.

2.2 Physics-Informed Deep Operator Network

Physics-Informed Deep Operator Network (PI-DeepONet) extends the DeepONet framework by embedding the governing physical constraints directly into the learning process. This is achieved by incorporating the governing partial differential equations (PDEs), boundary conditions, and initial conditions as constraints in the loss function.

Consider a PDE of the form:

$$\mathcal{F}(u; f) = 0 \quad \text{in } \Omega \times (0, T), \tag{4}$$

$$u(x, 0) = u_0(x) \quad \text{in } \Omega, \tag{5}$$

$$\mathcal{B}(u; f) = 0 \quad \text{on } \partial\Omega \times (0, T), \tag{6}$$

where f is the input function, u is the solution, $u_0(x)$ is the initial condition, $\mathcal{F}$ represents the PDE operator, and $\mathcal{B}$ represents the boundary conditions.

Our work focuses on scenarios constrained purely by physical laws, which are incorporated into the following loss function:

$$\begin{aligned}\mathcal{L} =& \frac{1}{N_r} \sum_{i=1}^{N_r} \|\mathcal{F}(u(x_r^i, t_r^i); f(x_r^i, t_r^i))\|^2 \\ &+ \frac{1}{N_b} \sum_{i=1}^{N_b} \|\mathcal{B}(u(x_b^i, t_b^i); f(x_b^i, t_b^i))\|^2 \\ &+ \frac{1}{N_0} \sum_{i=1}^{N_0} \|u(x_0^i, 0) - u_0(x_0^i)\|^2,\end{aligned} \tag{7}$$

where:

- $\{(y_r^i, t_r^i)\}_{i=1}^{N_r}$: Collocation points in the domain $\Omega \times (0, T)$,
- $\{(x_b^i, t_b^i)\}_{i=1}^{N_b}$: Boundary points on $\partial\Omega \times (0, T)$,
- $\{x_0^i\}_{i=1}^{N_0}$: Points for the initial condition.

2.3 Importance Sampling

Importance sampling is a statistical technique for estimating properties of a target distribution $p(x)$ by sampling from a more convenient proposal distribution $q(x)$. It reweights samples from $q(x)$ to approximate expectations under $p(x)$.

The expectation of a function $f(x)$ under the target distribution is:

$$\mathbf{E}_p[f(x)] = \int f(x) p(x)\, dx. \tag{8}$$

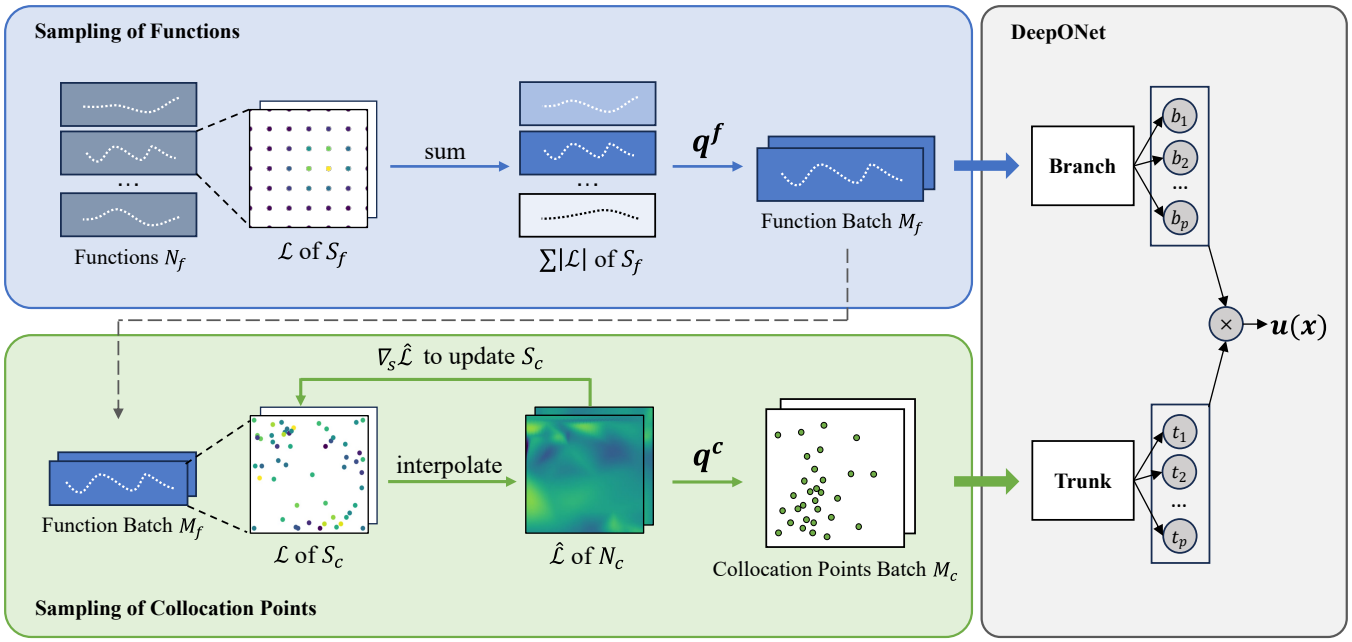

Figure 1: The Training of DONIS. Within a training iteration, importance sampling is first used to select higher-priority functions, forming the mini-batch of functions. Then, mini-batch of collocation points are sampled from regions of higher importance within the domains of these selected functions.

Instead of sampling directly from $p(x)$, importance sampling draws samples $\{x_i\}_{i=1}^N$ from $q(x)$ and approximates the expectation as:

$$\mathbf{E}_p[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N}\alpha(x_i)f(x_i), \tag{9}$$

where $\alpha(x) = \frac{p(x)}{q(x)}$ are the importance weights. A good $q(x)$ should closely approximate $p(x)$ in regions where $f(x)p(x)$ contributes the most, reducing the variance of $w(x)$.

In deep learning, importance sampling is utilized to enhance the efficiency of training by prioritizing samples that have a greater impact on the learning objective. This approach accelerates convergence and improves overall training efficiency. Specifically , the convergence rate of stochastic gradient methods is optimized when the sampling distribution aligns with the 2-norm of the gradient of the loss function [Needell *et al.*, 2014; Zhao and Zhang, 2015].

3 Our Method

In this section, we provide a detailed overview of the proposed DONIS framework, which consists of two key components: sampling of functions and sampling of collocation points. The overall framework is illustrated in Figure 1.

3.1 Method Overview

In practical DeepONet training, a single step typically involves mini-batch partitioning over both the function space and the collocation point space. Consequently, the sampling space is represented as the Cartesian product of the function space and the collocation point space, with a total size of $|N_f| \cdot |N_c|$:

$$\begin{aligned}\mathcal{L} = &\frac{1}{|M_f|\,|M_c|}\sum_{i=1}^{|M_f|}\sum_{j=1}^{|M_c|} l_r\left(\boldsymbol{f}_i, \boldsymbol{x}_j; \boldsymbol{\theta}\right)\\ &+\frac{1}{|M_f|\,|M_c^b|}\sum_{i=1}^{|M_f|}\sum_{j=1}^{|M_c^b|} l_b\left(\boldsymbol{f}_i, \boldsymbol{x}_j^b; \boldsymbol{\theta}\right)\\ &+\frac{1}{|M_f|\,|M_c^0|}\sum_{i=1}^{|M_f|}\sum_{j=1}^{|M_c^0|} l_0\left(\boldsymbol{f}_i, \boldsymbol{x}_j^0; \boldsymbol{\theta}\right),\end{aligned} \tag{10}$$

where $\boldsymbol{f}_i \in M_f$ represents a mini-batch of functions sampled from N_f. The points $\boldsymbol{x}_i \in M_c$, $\boldsymbol{x}_i^b \in M_c^b$, and $\boldsymbol{x}_i^0 \in M_c^0$ correspond to mini-batches of collocation points for the PDE residual, BC, and IC respectively. The terms l_r, l_b, and l_0 are the respective loss functions for each.

In this sense, a straightforward and ideal approach to importance sampling is to jointly compute the importance of all samples of size $|N_f| \cdot |N_c|$, and then samples a mini-batch of size $|M_f| \cdot |M_c|$ accordingly. However, this not only incurs significant computational costs, but also interfere with vectorization [Lu *et al.*, 2022]. Instead, we propose a two-step sampling approach that separates the sampling of functions and collocation points, striking a balance between effectiveness and computational cost. First, importance sampling is applied to the function set, selecting functions with higher importance to form the function batch for the current step. Next, batches of collocation points are sampled from regions

Algorithm 1 Importance Sampling of Functions

Input: Dataset of functions $N_f = \{\boldsymbol{f}_i\}_{i=1}^{|N_f|}$ and seed points $S_f = \{\boldsymbol{a}_n\}_{n=1}^{|S_f|}$.
Parameter: model weights $\boldsymbol{\theta}$.
Output: Batch of functions $M_f = \{\boldsymbol{f}_i^{'}\}_{i=1}^{|M_f|}$
for $i = 1$ **to** $|N_f|$ **do**
  Calculate $\{l_r(\boldsymbol{f}_i, \boldsymbol{a}_n; \boldsymbol{\theta}) \,|\boldsymbol{a}_n \in S_f\}$
  Compute q_i^f according to Eq (11)
end for
M_f sampled according to q_i^f
return M_f

with higher importance within the domains of the selected functions.

Apart from the two-step scheme described above, we further reduce the computational overhead in two aspects. Firstly, we replace gradient-based importance with the loss value to avoid gradient computation. Both theoretical analyses [Katharopoulos and Fleuret, 2017] and practical applications [Nabian *et al.*, 2021; Yang *et al.*, 2023] have shown that using the loss value as a proxy for the 2-norm of the loss gradient is a reasonable approximation. Secondly, because boundary and initial conditions usually act as penalty terms, the importance sampling methods discussed here focus solely on collocation points within the spatiotemporal domain, corresponding to the PDE loss.

3.2 Importance Sampling of Functions

First, we consider sampling on the function set (DONIS-F), where an input function and its corresponding set of collocation points are treated as a single sample. As mentioned earlier, we measure the importance of a sample based on its loss value. Therefore, to measure the importance of a function sample, we need to obtain the overall loss value of the spatiotemporal domain Ω corresponding to this function. Here, instead of computing the loss at every point of N_c, we adopt an approximate method to bypass the costly process, as demonstrated in Algorithm 1.

Here, we define the sampling probability q_i^f for a function sample $\boldsymbol{f}_i \in N_f$ as proportional to the loss (i.e., PDE residuals) l_r over a set of seed points $S_f = \{\boldsymbol{a}_n\}_{n=1}^{|S_f|}$:

$$q_i^f = \frac{\sum_{n=1}^{|S_f|} l_r(\boldsymbol{f}_i, \boldsymbol{a}_n; \boldsymbol{\theta})}{\sum_{m=1}^{|N_f|}\sum_{n=1}^{|S_f|} l_r(\boldsymbol{f}_m, \boldsymbol{a}_n; \boldsymbol{\theta})}, \quad i \in \{1, 2, \ldots, |N_f|\}, \tag{11}$$

where seed points S_f is a sparse uniform grid across Ω.

By this means, we estimates the overall loss value for each function in the function set, thereby ensuring that more important functions are selected with higher probability for training.

3.3 Importance Sampling of Collocation Points

After function sampling, the sampling space for the mini-batch is significantly reduced. However, the sampling of functions primarily reflects the overall error. To better capture localized errors, we extend this regime to collocation points (DONIS-C).

Algorithm 2 Importance Sampling of Collocation Points

Input: Batch of functions $M_f = \{\boldsymbol{f}_i^{'}\}_{i=1}^{|M_f|}$, sets of seed points $\mathcal{S} = \{S_{c,i}\}_{i=1}^{|M_f|}$ where $S_{c,i} = \{\boldsymbol{b}_{i,l}\}_{i=1,l=1}^{|M_f|,|S_c|}$.
Parameter: Collocation points $N_c = \{\boldsymbol{c}\}_{k=1}^{|N_c|}$, model weights $\boldsymbol{\theta}$.
Output: Batch of collocation points $M_c = \{M_{c,i}\}_{i=1}^{|M_f|}$, updated sets of seed points $\mathcal{S}^{'}$.
$M_c \leftarrow \{\}, \mathcal{S}^{'} \leftarrow \{\}$
for $i = 1$ **to** $|M_f|$ **do**
  Calculate $\{l_r(\boldsymbol{f}_i, \boldsymbol{b}_{i,l}; \boldsymbol{\theta}) \,|\boldsymbol{b}_{i,l} \in S_{c,i}\}$
  Interpolate to approximate $\{\hat{l}_r(\boldsymbol{f}_i, \boldsymbol{c}_k; \boldsymbol{\theta}) \,|\boldsymbol{c}_k \in N_c\}$
  Compute $q_{i,j}^c$ according to Eq (12)
  $M_c \leftarrow M_{c,i}$ sampled according to $q_{i,j}^c$
  Compute $g_{i,j}$ according to Eq (13)
  $\mathcal{S}^{'} \leftarrow S_{c,i}^{'}$sampled according to $g_{i,j}$
end for
return $M_c, \mathcal{S}^{'}$

Specifically, we need to estimate the importance of each element in the collocation point set. Contrary to the input functions in operator learning, the collocation points are distributed across a continuous spatiotemporal domain, which facilitates the use of interpolation methods to accelerate this process. Evidently, the effectiveness of interpolation heavily relies on the distribution of the source points. We refer to the source points as the seed points S_c, similar to S_f introduced in the previous section.

Here, we present a dynamic interpolation algorithm to accelerate importance sampling for collocation points demonstrated in Algorithm 2, where the distribution of S_c is determined by the spatial gradient of the the loss value (i.e., PDE residuals l_r). Specifically, in each training step, we calculates each loss value of points in S_c. Then, interpolation is employed to estimate the the loss value over the spatiotemporal domain Ω, from which the mini-batch of collocation points is sampled. We define the sampling probability $q_{i,j}^c$ for a collocation point $\boldsymbol{c}_j \in N_c$ corresponding to selected function $\boldsymbol{f}_i^{'} \in |M_f|$ as proportional to the estimated PDE residuals $\hat{l}_r$:

$$q_{i,j}^c = \frac{\hat{l}_r\left(\boldsymbol{f}_i^{'}, \boldsymbol{c}_j; \boldsymbol{\theta}\right)}{\sum_{k=1}^{|N_c|}\hat{l}_r\left(\boldsymbol{f}_i^{'}, \boldsymbol{c}_k; \boldsymbol{\theta}\right)}, i \in \{1, 2, \ldots, |M_f|\}, \quad j \in \{1, 2, \ldots, |N_c|\}, \tag{12}$$

where $\hat{l}_r$ denotes the approximate of l_r by interpolation over S_c.

Lastly, we compute the spatial gradient of the loss value, and update S_c based on the magnitude of the gradient:

$$g_{i,j} \propto \left|\nabla_s \hat{l}_r\left(\boldsymbol{f}_i^{'}, \boldsymbol{c}_j; \boldsymbol{\theta}\right)\right|, \tag{13}$$

where∇_s denotes the Sobel operator, and $g_{i,j}$ is the probability of the j-th point of N_c being selected as a seed point for

the i-th function.

Compared to a fixed uniform grid, seed points updated based on the spatial gradient allows the interpolation results to better align with the true data distribution, thereby enhancing the accuracy of the importance sampling algorithm. In practice, this calculation is performed via the Sobel operator, which gives an approximation of the spatial gradient, making the algorithm more efficient with minimal hyperparameters.

3.4 Importance Reweighting

As shown in Eq (9), reweighting is necessary to ensure that samples from the proposal distribution accurately represent the target distribution. Hence, we modify the PDE loss in Eq (10) when utilizing importance sampling for functions and/or collocation points:

$$\mathcal{L}_r = \frac{1}{|M_f|\,|M_c|} \sum_{i=1}^{|M_f|} \alpha_i \sum_{j=1}^{|M_c|} \beta_{i,j} l_r\left(\boldsymbol{f}_i, \boldsymbol{x}_j; \boldsymbol{\theta}\right), \quad \boldsymbol{f}_i \in M_f, \boldsymbol{x}_j \in M_c, \tag{14}$$

where α_i is the importance weight of the i-th function, and $\beta_{i,j}$ is the importance weight of the j-th point for the i-th function:

$$\alpha_i = \frac{p_i^f}{q_i^f} = \frac{1}{|N_f| q_i^f}, \tag{15}$$

$$\beta_{i,j} = \frac{p_{i,j}^c}{q_{i,j}^c} = \frac{1}{|N_c| q_{i,j}^c}. \tag{16}$$

4 Experiments

In this section, we validate the proposed methods through a series of experiments and evaluate their performance using the specified metrics.The implementation of these methods is built upon the DeepXDE library [Lu *et al.*, 2021b], with the complete code available in the supplementary materials for reference. Table 1 presents the hyperparameters used in our experiments.

4.1 Experimental Setting

We consider three partial differential equation scenarios: the Allen-Cahn equation, viscous Burgers' equation, and a nonlinear diffusion-reaction equation. Within each PDE problems, we compare the results of vanilla PI-DeepONet with three variants: PI-DeepONet with importance sampling on collocation points (DONIS-C), importance sampling on functions (DONIS-F), and importance sampling on both collocation points and functions (DONIS).

Parameter	Value
Branch Net	[128, 128, 128, 128, 128, 128]
Trunk Net	[2, 128, 128, 128, 128, 128]
Epoch	30,000
Learning Rate	5×10^{-4}
$\lvert N_f\rvert$	1000
$\lvert N_c\rvert$	10000
$\lvert M_f\rvert$	50
$\lvert M_c\rvert$	2000

Table 1: Network and training parameters. $|N_f|$ and $|N_c|$ denotes the size of dataset of functions and collocation points, $|M_f|$ and $|M_c|$ is batch size of functions and collocation points.

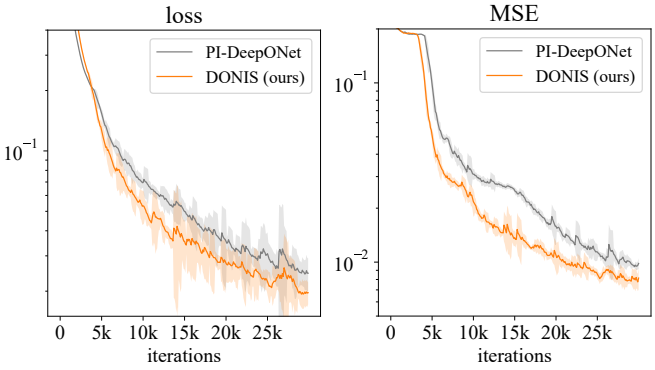

Figure 2: Convergence curves and prediction errors of Allen-Cahn equation.

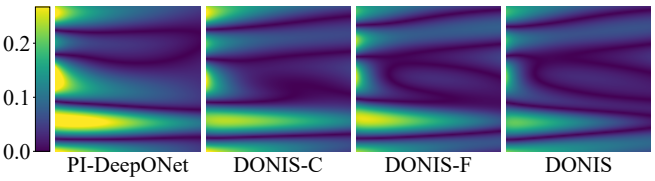

Figure 3: Prediction errors of Allen-Cahn equation over Ω.

	MSE	L^2 RE
PI-DeepONet	8.35e-03	1.07e-01
DONIS-C (ours)	8.24e-03	1.06e-01
DONIS-F (ours)	8.20e-03	1.06e-01
DONIS (ours)	**6.81e-03**	**9.62e-02**

Table 2: Metrics of Allen-Cahn equation

Metrics. We first train the model without any labeled data within the spatiotemporal domain Ω and then evaluate its performance using results obtained from traditional numerical solvers. The evaluation is conducted based on the Mean Square Error (MSE) and the L^2 relative error as metrics.

Hyperparameters. General hyperparameters are shown in Tabel 1.

4.2 Allen-Cahn Equation

We first consider the one-dimensional Allen-Cahn equation in the form of:

$$\frac{\partial u}{\partial t} = \epsilon^2 \left(\frac{\partial^2 u}{\partial x^2}\right)^2 - \left(u^3 - u\right), x \in [0,1], t \in [0,1], \tag{17}$$

with periodic boundary conditions. ϵ, which controls interface thickness, is set to 0.01.

We aim to train an operator mapping the initial condition $u_0(x)$ to the solution, i.e., $G : u_0(x) \mapsto u(x,t), \quad t \in [0,1]$,

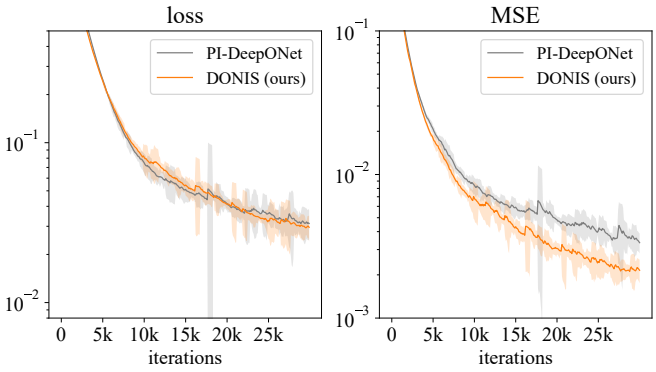

Figure 4: Convergence curves and prediction errors of Burger's equation.

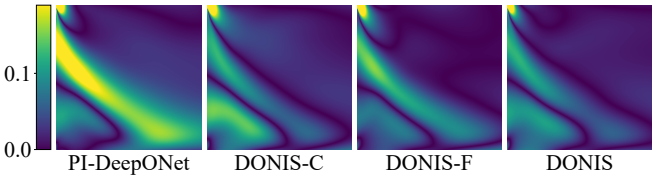

Figure 5: Prediction errors of Burger's equation over Ω.

	MSE	L^2 RE
PI-DeepONet	2.31e-03	9.14e-02
DONIS-C (ours)	2.08e-03	8.67e-02
DONIS-F (ours)	1.61e-03	7.64e-02
DONIS (ours)	**1.42e-03**	**7.16e-02**

Table 3: Metrics of Burger's equation

where the initial conditions are generated based on the Exponential Sine Squared kernel:

$$k(x, x') = \exp\left(-\frac{2\sin^2\left(\frac{\pi|x-x'|}{T}\right)}{\ell^2}\right), \tag{18}$$

where $l = 2, T = 1$.

We set the size of seed points as 50 for S_f and 200 for S_c for this problem.

Figure 2 illustrates that DONIS accelerates loss convergence and achieves lower prediction error over the course of the training procedure. Figure 3 and the metrics in Table 2 demonstrate a consistent reduction in prediction error when applying DONIS-C, DONIS-F, or both in combination. Overall, DONIS reduces MSE by 18% and brings L^2 relative error down to below 10%. By leveraging importance sampling, DONIS prioritizes functions and collocation points that exert a greater influence on the learning objective, thereby facilitating faster convergence and improved predictive accuracy.

4.3 Burgers' Equation

Next, we consider the one-dimensional Burgers equation:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, x \in [0,1], t \in [0,1], \tag{19}$$

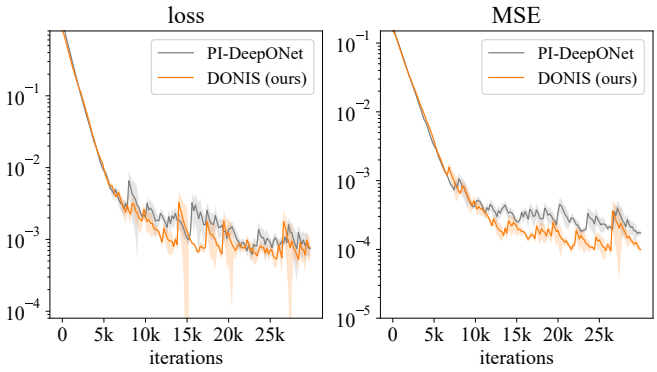

Figure 6: Convergence curves and prediction errors of nonlinear diffusion equation.

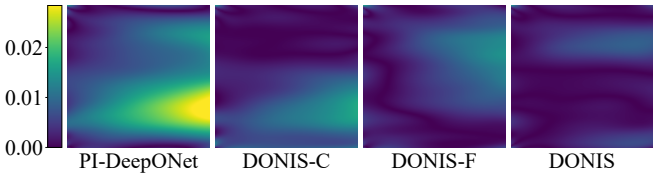

Figure 7: Predition errors of nonlinear diffusion reaction equation.

	MSE	L^2 RE
PI-DeepONet	3.27e-05	1.26e-02
DONIS-C (ours)	3.16e-05	1.24e-02
DONIS-F (ours)	2.93e-05	1.19e-02
DONIS (ours)	**2.32e-05**	**1.06e-02**

Table 4: Metrics of nonlinear diffusion-reaction equation

with zero Dirichlet boundary conditions. The viscosity ν is set to 0.01. Similarly, the goal is to train an operator mapping the initial condition $u_0(x)$ to the solution, i.e., $G : u_0(x) \mapsto u(x,t), \quad t \in [0,1]$, where the initial conditions are generated based on the Radial Basis Function (RBF) kernel:

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right), \tag{20}$$

where $l = 0.5$.

We set the size of seed points as 100 for S_f and 500 for S_c for this problem.

As shown in Figure 4, for the Burgers equation, DONIS achieves a significant improvement in prediction accuracy, even though its acceleration of convergence is less pronounced. Table 3 reports a 39% reduction in MSE achieved by DONIS. Notably, DONIS-F demonstrates superior performance compared to DONIS-C in this experiment, which can be attributed to the Burgers equation's higher sensitivity to variations in input functions (i.e., initial conditions).This sensitivity arises from its nonlinear convection term, particularly under conditions of weak diffusion.

$\lvert S_f \rvert$	MSE	L^2 RE
50	1.32e-03	6.90e-02
100	1.21e-03	6.62e-02
200	**1.20e-03**	**6.58e-02**
500	1.22e-03	6.63e-02

Table 5: Experiments on $|S_f|$ in DONIS-F for the Burgers' Equation.

$\lvert S_c \rvert$	MSE	L^2 RE
100	2.26e-03	9.04e-02
200	2.28e-03	9.08e-02
500	**2.08e-03**	**8.67e-02**
1000	2.12e-03	8.75e-02

Table 6: Experiments on $|S_c|$ in DONIS-C for the Burgers' Equation.

4.4 Nonlinear Diffusion-Reaction Equation

Lastly, we consider a nonlinear diffusion-reaction equation in the form of:

$$\frac{\partial u}{\partial t} = 0.01\frac{\partial^2 u}{\partial x^2} - 0.01u^2 + f(x), x \in [0,1], t \in [0,1], \quad (21)$$

subject to zero Dirichlet boundary conditions and an initial condition given by $u(x,0) = 0$. Here, we train an operator mapping the source term $f(x)$ to the solution, i.e., $G : f(x) \mapsto u(x,t), \quad t \in [0,1]$, where $f(x)$ are generated based on the RBF kernel shown in Eq (20).

We set the size of seed points as 50 for S_f and 200 for S_c for this problem.

Figure 6 and 7, along with Table 4, illustrate the experimental results for the nonlinear diffusion-reaction equation. By fixing the boundary and initial conditions, the convergence difficulty of the corresponding loss terms was significantly reduced, which greatly simplified the convergence process. Notably, despite the changes in problem formulation and the reduction in training complexity, DONIS achieved faster convergence and lower prediction error. Specifically, DONIS-F, DONIS-C, and DONIS all demonstrated further improvements with DONIS demonstrating a 29% reduction in MSE as shown in Table 4.

4.5 Hyperparameter Analysis

In this section, we explore the effect of the seed points of DONIS-F and DONIS-C to further optimize the performance of DONIS.

Table 5 and 6 reveal that the accuracy of DONIS generally improves as the sizes of $|S_c|$ or $|S_f|$ increase. This is because a larger number of seed points enhances the accuracy of importance estimation, thus improving the overall performance of the method. However, once the number reaches a certain threshold, the improvement becomes less significant, and the computational cost increases considerably. Hence, specific sizes of S_c and S_f $|S_c|$ were chosen in our experiments to strike a balance between accuracy and computational cost.

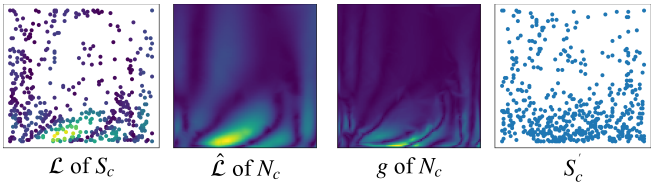

Figure 8: Dynamic interpolation algorithm during the training process for Burger's equation.

	MSE	L^2 RE
DON	2.31e-03	9.14e-02
DONIS-C-fixed(ours)	2.21e-03	8.94e-02
DONIS-C(ours)	2.08e-03	8.67e-02
DONIS-fixed(ours)	1.67e-03	7.76e-02
DONIS(ours)	**1.42e-03**	**7.16e-02**

Table 7: Ablation study of dynamic interpolation algorithm in DONIS-C for the Burgers' Equation.

4.6 Ablation Study

In the following section, we present ablation experiments on the dynamic interpolation algorithm used in DONIS-C. The algorithm updates the seed points S_c by leveraging the spatial gradient of the loss value, with the goal of enhancing the accuracy of DONIS-C by ensuring that the interpolation results better reflect the true data distribution. Figure 8 provides a visualization of the updating process of S_c.

The results are summarized in Table 7, where methods that do not employ the dynamic interpolation algorithm are labeled with "fixed", indicating that the seed points S_c remain unchanged after initialization. As evidenced by the results, incorporating dynamic interpolation improves the performance of DONIS-C, both when used independently and in conjunction with DONIS-F.

5 Conclusions

In this paper, we propose a novel approach to accelerate the training of physics-informed DeepONet: PI-DeepONet with Importance Sampling (DONIS). Leveraging the architectural flexibility of DeepONet, DONIS introduces a two-step importance sampling framework that sequentially applies importance sampling to the function inputs and collocation points of DeepONet. In function sampling, the importance of function samples is approximated using the loss function evaluated over a uniform grid. For collocation points, their importance is estimated using a dynamic interpolation algorithm. Evaluated on three widely tested PDEs and across different operator mapping configurations, DONIS consistently enhances the convergence speed of self-supervised operator training while reducing prediction errors.These results demonstrate the practical versatility of DONIS, making it a robust and efficient framework applicable to a wide range of operator learning tasks.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant 62306199; the National Key Research and Development Program of China (2022YFC3801304).

References

[Bonev *et al.*, 2023] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere. In *International conference on machine learning*, pages 2806–2823. PMLR, 2023.

[Brecht *et al.*, 2023] Rüdiger Brecht, Dmytro R Popovych, Alex Bihlo, and Roman O Popovych. Improving physics-informed deeponets with hard constraints. *arXiv preprint arXiv:2309.07899*, 2023.

[Goswami *et al.*, 2022] Somdatta Goswami, Minglang Yin, Yue Yu, and George Em Karniadakis. A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391:114587, 2022.

[He *et al.*, 2024] Junyan He, Seid Koric, Diab Abueidda, Ali Najafi, and Iwona Jasiuk. Geom-deeponet: A point-cloud-based deep operator network for field predictions on 3d parameterized geometries. *Computer Methods in Applied Mechanics and Engineering*, 429:117130, 2024.

[Howard *et al.*, 2023] Amanda A Howard, Mauro Perego, George Em Karniadakis, and Panos Stinis. Multifidelity deep operator networks for data-driven and physics-informed problems. *Journal of Computational Physics*, 493:112462, 2023.

[Huang *et al.*, 2022] Shudong Huang, Wentao Feng, Chenwei Tang, and Jiancheng Lv. Partial differential equations meet deep neural networks: A survey. *arXiv preprint arXiv:2211.05567*, 2022.

[Jiao *et al.*, 2024] Anran Jiao, Qile Yan, Jhn Harlim, and Lu Lu. Solving forward and inverse pde problems on unknown manifolds via physics-informed neural operators. *arXiv preprint arXiv:2407.05477*, 2024.

[Jin *et al.*, 2022] Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

[Katharopoulos and Fleuret, 2017] Angelos Katharopoulos and François Fleuret. Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*, 2017.

[Kumar *et al.*, 2025] Varun Kumar, Somdatta Goswami, Katiana Kontolati, Michael D Shields, and George Em Karniadakis. Synergistic learning with multi-task deeponet for efficient pde problem solving. *Neural Networks*, page 107113, 2025.

[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.*, 2023] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

[Li *et al.*, 2024] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

[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.

[Nabian *et al.*, 2021] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 2021.

[Needell *et al.*, 2014] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *Advances in neural information processing systems*, 27, 2014.

[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.

[Wang and Perdikaris, 2023] Sifan Wang and Paris Perdikaris. Long-time integration of parametric evolution equations with physics-informed deeponets. *Journal of Computational Physics*, 475:111855, 2023.

[Wang *et al.*, 2021] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[Wen *et al.*, 2022] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[Xu *et al.*, 2023] Wuzhe Xu, Yulong Lu, and Li Wang. Transfer learning enhanced deeponet for long-time prediction of evolution equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10629–10636, 2023.

[Yang *et al.*, 2023] Zijiang Yang, Zhongwei Qiu, and Dongmei Fu. Dmis: Dynamic mesh-based importance sampling for training physics-informed neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5375–5383, 2023.

[Zhao and Zhang, 2015] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9. PMLR, 2015.