Importance-Weighted Operator Learning Under Probability Measure Shifts

Lei Sun
Nara Institute of Science and Technology
sun.lei.sn5@is.naist.jp

Yusuke Tanaka
NTT Communication Science Laboratories
RIKEN AIP
ysk.tanaka@ntt.com

Xiaocheng Shang
University of Birmingham
x.shang.1@bham.ac.uk

Takaharu Yaguchi
Kyushu University
RIKEN AIP
yaguchi@imi.kyushu-u.ac.jp

Tomoharu Iwata
NTT Communication Science Laboratories
tomoharu.iwata@ntt.com

Abstract

Operator learning has attracted significant interest for its capability to approximate mappings between function spaces. A common assumption in this field is that training and test samples are drawn from the same probability measure. However, this assumption is often violated, leading to probability measure shifts. Under such shifts, standard operator learning methods often lead to degraded performance because the expected training error does not match the expected test error. In this paper, we propose importance-weighted operator learning (IWOL), a framework for learning operators under probability measure shifts in function spaces. We prove that, when the test measure is absolutely continuous with respect to the training measure, the expected test error is equivalent to an importance-weighted expected training error, with weights given by the Radon–Nikodym derivative. This result provides a theoretical basis for correcting measure shifts in operator learning, but importance weighting relies on this absolute-continuity condition and may yield unbounded weights. To overcome these limitations, we introduce relative importance weights defined with respect to a mixture of the training and test measures. The resulting weights remain well-defined even when absolute continuity fails, and they are uniformly bounded. These weights induce a relative importance-weighted training objective for neural operators under probability measure shifts. To estimate these weights, we present binary classifiers that take functions as inputs, defined through functionals modeled by kernel integral operators. This design enables discretization-invariant weight estimation in function space. Experiments on multiple PDE benchmarks and neural operator architectures demonstrate that the proposed method consistently improves performance under probability measure shifts.

1 Introduction

Many problems in science and engineering require solving partial differential equations (PDEs), with applications in physical modeling [1, 2], uncertainty quantification [3, 4], and design optimization [5,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

6]. Traditionally, these problems are addressed using numerical methods. However, numerical simulations can be computationally expensive because they often require fine-grained discretizations and many iterative solves.

Recently, operator learning has attracted substantial attention for its ability to predict mesh-free solutions directly from data, without the need for numerical simulations. Generally, operator learning aims to learn mappings between function spaces, known as "operators". In the context of PDEs, the solution operator maps input functions (e.g., initial and boundary conditions) to solution functions. Various deep neural network architectures, such as Deep Operator Networks (DeepONet) [7] and Fourier Neural Operators (FNO) [8], have been proposed to approximate solution operators from many input-output function pairs. However, most of the existing operator learning methods rely on the assumption that input functions during the training and test phases are drawn from the same probability measure. Unfortunately, this assumption is frequently violated in scientific and engineering workflows. For instance, coefficients or forcing terms may shift due to changes in material properties, environmental conditions, or data acquisition methods; sampling strategies for initial and/or boundary conditions in downstream tasks may differ from those used during training; moreover, simulation-generated training data often fail to perfectly reflect the statistical characteristics of real-world inputs. In these scenarios, while the underlying operator remains constant, the probability measure of the input functions shifts between training and test phases. Such probability measure shifts can significantly degrade generalization performance, as minimizing the expected error under the training measure does not guarantee optimality under the shifted test measure.

To address probability measure shifts, we propose an importance-weighted operator learning framework. We first show that, when the test measure is absolutely continuous with respect to the training measure, the expected test error can be rewritten as an importance-weighted training error, where the weight is given by the RadonNikodym derivative. This provides a principled way to correct the mismatch between the training and test measures. However, in infinite-dimensional function spaces, the required absolute-continuity condition can be restrictive and may fail to hold. Moreover, the importance weights can be unbounded, which may lead to a high-variance finite-sample estimate of the expected test error. To overcome these issues, we introduce relative importance weights by replacing the training measure with a mixture of the training and test measures as the reference measure. The resulting weights are always well defined and are uniformly bounded. We then use these weights to define a relative importance-weighted training objective for learning neural operators under probability measure shifts. To estimate the relative importance weights, we reduce the weight estimation problem to a binary classification task in function space. Specifically, we construct binary classifiers that take functions as inputs, implemented via functionals modeled by kernel integral operators. This design enables discretization-invariant weight estimation directly in function space.

Our contributions are summarized as follows:

- We formulate the problem of probability measure shifts in operator learning, where the training and test input functions follow different probability measures while the underlying operator remains unchanged.
- We develop an importance-weighted operator learning framework. Under an absolute-continuity assumption, we show that the expected test error can be represented as an importance-weighted training error, providing a principled way to correct for probability measure shifts in operator learning.
- We introduce relative importance weights for operator learning. By using a mixture of the training and test measures as the reference measure, the proposed weights remain well-defined even when the test measure is not absolutely continuous with respect to the training measure, and are uniformly bounded to avoid extreme reweighting.
- We propose a discretization-invariant method for estimating importance weights in function spaces, using binary classifiers parameterized by kernel integral operators.
- We demonstrate through experiments on multiple physical systems that our proposed method outperforms existing operator learning algorithms under probability measure shifts.

2 Related work

Operator learning. Many deep neural network architectures have been proposed for learning solution operators of PDEs [7, 8, 9, 10, 11, 12]. In these studies, the mapping between input functions (e.g., initial and boundary conditions) and output functions (i.e., solutions) is parameterized by deep neural networks. However, these methods rely on the assumption that input functions during the training and test phases are drawn from the same probability measure. In scenarios involving probability measure shifts, where the training and test measures differ, the expected test error differs from the expected training error. Since general operator learning methods are designed to minimize the expected training error via parameter optimization, they cannot guarantee that the expected test error will be minimized under probability measure shifts.

Distribution shift adaptation. To mitigate discrepancies between training and test distributions, prior studies have proposed various distribution shift adaptation and domain adaptation methods [13, 14, 15]. Most of these methods are developed for fixed-dimensional sample representations, which differ fundamentally from the operator learning setting considered in this work, where the goal is to learn mappings between function spaces. One representative class of methods reduces the distributional discrepancy between the training and test domains by learning distribution invariant representations [16, 17, 18]. However, such methods do not explicitly minimize the test risk; the resulting invariant features therefore do not necessarily improve performance on the test distribution [19]. Another representative class of methods is importance weighting, which adapts to covariate shift by estimating the density ratio between the training and test distributions and using the resulting weighted empirical risk over the training distribution to approximate the test risk [20].

Although importance weighting methods have been extensively studied for finite-dimensional distribution shift problems [21], their direct application to operator learning remains challenging. A straightforward approach estimates the densities of the training and test distributions separately and then takes their ratio, but density estimation is difficult, and ratio estimation can amplify errors, especially in high-dimensional settings [22]. To avoid explicit density estimation, direct density-ratio estimation methods have been proposed, including approaches based on Kullback–Leibler divergence minimization [23], least-squares estimation [24], moment matching in reproducing kernel Hilbert spaces [25], and binary classifiers [26].

However, these methods are mainly designed for finite-dimensional data. In operator learning, inputs are functions, so the density ratio must be defined between probability measures on a function space. In infinite-dimensional spaces, such ratios are generally not well defined without a suitable reference measure [27]; even when formulated as Radon–Nikodym derivatives, their existence requires a strong absolute-continuity assumption that may fail in infinite-dimensional function spaces, and the resulting ratios may be unbounded, leading to high-variance weighted empirical risks. Moreover, applying finite-dimensional estimators to discretized function values makes the estimated weights sensitive to grid resolution, sampling schemes, and specific discretizations.

Relative density-ratio methods alleviate the unboundedness issue by using a mixture of the training and test distributions as the reference distribution, yielding bounded weights [28]. Nevertheless, existing methods are primarily developed for finite-dimensional distributions and do not address function-space inputs or discretization invariance. Our method extends relative density-ratio estimation to operator learning by defining relative importance weights with respect to a mixture measure on the function space, which makes the weights well-defined and uniformly bounded. We further estimate these weights using a functional classifier parameterized by kernel integral operators, yielding a discretization-invariant estimator for operator learning.

3 Problem setting

In this paper, we consider a novel operator learning problem under probability measure shifts, where the input functions in the training and test phases follow different probability measures. However, the operator remains unchanged. Our problem setting is defined as follows.

Let $\mathcal{A}$ and $\mathcal{U}$ denote the input and output function spaces. An input function $a \in \mathcal{A}$ represents the initial or boundary conditions, spatially varying or constant coefficients, or external forcing terms, and so on. The corresponding output function $u \in \mathcal{U}$ represents the solution associated with the given

input a. The objective is to approximate a solution operator $\mathcal{S}^\dagger : \mathcal{A} \to \mathcal{U}$, which defines a nonlinear mapping between input and output function spaces. Accordingly, the input and output functions satisfy the following relationship: $u = \mathcal{S}^\dagger[a]$.

In the training phase, we assume that we obtain I input-output function pairs $\{(a_i^{\mathrm{train}}, u_i^{\mathrm{train}}) \mid i = 1, \cdots, I\}$. Here, $a_i^{\mathrm{train}} \sim \mu^{\mathrm{train}}$ are assumed to be independent and identically distributed samples generated from a training probability measure μ^{train} defined on $\mathcal{A}$, and u_i^{train} are assumed to be generated from the underlying (true) operator $\mathcal{S}^\dagger$ conditioned on a_i^{train}, possibly corrupted by noise. Additionally, we are given J input functions $\{a_j^{\mathrm{test}} \mid j = 1, \cdots, J\}$ drawn from μ^{test}, where the corresponding solutions are not provided.

In the test phase, we obtain unseen input function $a^{\mathrm{test}} \sim \mu^{\mathrm{test}}$ drawn from a test probability measure μ^{test} defined on $\mathcal{A}$. Given input function a^{test}, we would like to predict the corresponding solution u^{test}. Note that a^{test} is different from that used in the training phase. Under the probability measure shifts scenario, we assume $\mu^{\mathrm{train}} \neq \mu^{\mathrm{test}}$, while the true operator $\mathcal{S}^\dagger$ remains unchanged.

Expected error mismatch under probability measure shifts. In the training phase, we aim to build an approximation of solution operator $u = \mathcal{S}_\theta[a]$ using deep neural networks (DNNs), where θ represents their parameters. The parameters can be estimated by minimizing the following expected training error,

$$L^{\mathrm{train}}(\theta) = \int_{\mathcal{A}} \left\| \mathcal{S}^\dagger[a] - \mathcal{S}_\theta[a] \right\|_{\mathcal{U}}^2 \mathrm{d}\mu^{\mathrm{train}}[a], \tag{1}$$

where $\|\cdot\|_{\mathcal{U}}^2$ is the norm on output function space $\mathcal{U}$ (typically L^2-norm). While minimizing $L^{\mathrm{train}}(\theta)$ optimizes the model performance with respect to training measure μ^{train}, our primary goal is to minimize the prediction error with respect to test measure μ^{test}. The expected test error is given by

$$L^{\mathrm{test}}(\theta) = \int_{\mathcal{A}} \left\| \mathcal{S}^\dagger[a] - \mathcal{S}_\theta[a] \right\|_{\mathcal{U}}^2 \mathrm{d}\mu^{\mathrm{test}}[a]. \tag{2}$$

In the situation where $\mu^{\mathrm{train}} = \mu^{\mathrm{test}}$, the expected training error $L^{\mathrm{train}}(\theta)$ matches the expected test error $L^{\mathrm{test}}(\theta)$. Therefore, optimizing parameters θ by minimizing $L^{\mathrm{train}}(\theta)$ ensures a corresponding reduction in $L^{\mathrm{test}}(\theta)$. However, when $\mu^{\mathrm{train}} \neq \mu^{\mathrm{test}}$, the expected training error and the expected test error do not match: $L^{\mathrm{train}}(\theta) \neq L^{\mathrm{test}}(\theta)$. Since standard operator learning methods focus on minimizing $L^{\mathrm{train}}(\theta)$, they fail to guarantee the minimization of $L^{\mathrm{test}}(\theta)$ under probability measure shifts.

4 Proposed method

4.1 Importance-weighted operator learning (IWOL)

To address the probability measure shifts between training and test phases, we propose the IWOL framework. Our goal is to approximate the expected test error $L^{\mathrm{test}}(\theta)$ by leveraging input-output function pairs $\{(a_i^{\mathrm{train}}, u_i^{\mathrm{train}})\}_{i=1}^I$ from the training measure μ^{train} and unlabeled input functions $\{a_j^{\mathrm{test}}\}_{j=1}^J$ drawn from the test measure μ^{test}. In this subsection, we show that the expected test error can be reformulated as an importance-weighted training error.

Assume that the test measure μ^{test} is absolutely continuous with respect to the training measure μ^{train}, denoted by $\mu^{\mathrm{test}} \ll \mu^{\mathrm{train}}$. Then the Radon–Nikodym derivative

$$w[a] := \frac{\mathrm{d}\mu^{\mathrm{test}}}{\mathrm{d}\mu^{\mathrm{train}}}[a] \tag{3}$$

is well defined. Under this assumption, the expected test error can be rewritten as follows:

$$\begin{aligned} L^{\mathrm{test}}(\theta) &= \int_{\mathcal{A}} \left\| \mathcal{S}^\dagger[a] - \mathcal{S}_\theta[a] \right\|_{\mathcal{U}}^2 \mathrm{d}\mu^{\mathrm{test}}[a] \\ &= \int_{\mathcal{A}} \frac{\mathrm{d}\mu^{\mathrm{test}}}{\mathrm{d}\mu^{\mathrm{train}}}[a] \left\| \mathcal{S}^\dagger[a] - \mathcal{S}_\theta[a] \right\|_{\mathcal{U}}^2 \mathrm{d}\mu^{\mathrm{train}}[a] \\ &= \int_{\mathcal{A}} w[a] \left\| \mathcal{S}^\dagger[a] - \mathcal{S}_\theta[a] \right\|_{\mathcal{U}}^2 \mathrm{d}\mu^{\mathrm{train}}[a] \\ &=: L^{\mathrm{IWOL}}(\theta). \end{aligned} \tag{4}$$

This means that $L^{\mathrm{IWOL}}(\theta)$ is an equivalent representation of $L^{\mathrm{test}}(\theta)$, so minimizing $L^{\mathrm{IWOL}}(\theta)$ yields the same optimal parameters as minimizing $L^{\mathrm{test}}(\theta)$. Although importance-weighted risk correction is well established in finite-dimensional covariate-shift settings [29], Eq. (4) formulates this correction at the level of probability measures on function spaces, thereby providing a theoretical basis for correcting probability measure shifts in operator learning. This equivalence, however, relies on the assumption of absolute continuity, which guarantees the existence of the Radon–Nikodym derivative [30]. In finite-dimensional parametrized function cases, the absolute-continuity condition can be satisfied when the parameter distributions differ but have overlapping support. In general operator-learning settings, however, the inputs are often defined on infinite-dimensional function spaces, where absolute continuity is a strong assumption that may fail to hold. When this assumption fails, the importance-weighted training error is no longer well defined. This motivates the use of a modified reference measure that guarantees the existence of importance weights in function spaces.

4.2 Relative importance weights

To address the limitation that the importance-weighted training error may be ill defined in infinite-dimensional function spaces, we construct an alternative reference measure that dominates the test measure while incorporating the training measure. Specifically, we introduce the following mixture measure:

$$\mu^{\alpha} := (1-\alpha)\mu^{\mathrm{train}} + \alpha\mu^{\mathrm{test}}, \tag{5}$$

where $\alpha \in (0,1)$ is the mixing parameter. The mixture measure μ^{α} is a convex combination of the training and test measures. Since it contains the component $\alpha\mu^{\mathrm{test}}$, it dominates the test measure for any $\alpha > 0$, while its $(1-\alpha)\mu^{\mathrm{train}}$ component keeps the reference measure tied to the training measure. Therefore, for any $\alpha > 0$, we have $\mu^{\mathrm{test}} \ll \mu^{\alpha}$ (i.e., μ^{test} is absolutely continuous with respect to μ^{α}).

We then define the relative importance weight as:

$$w_{\alpha}[a] := \frac{\mathrm{d}\mu^{\mathrm{test}}}{\mathrm{d}\mu^{\alpha}}[a]. \tag{6}$$

The relative weight has two key advantages. First, for any $\alpha > 0$, we have $\mu^{\mathrm{test}} \ll \mu^{\alpha}$, and hence $w_{\alpha}[a]$ is always well defined. Second, $w_{\alpha}[a]$ is uniformly bounded. The importance weight defined in Eq. (3) can be unbounded, allowing a small number of samples with extremely large weights to dominate the weighted training error. Prior studies on finite-dimensional covariate shift have shown that such unbounded or extreme weights can substantially increase the variance of finite-sample estimates of the expected test error, thereby compromising the learning stability and robustness [31, 32, 33]. This issue can also arise in operator learning over function spaces, providing an additional motivation for introducing relative importance weights. Indeed, since $\mu^{\alpha} = (1-\alpha)\mu^{\mathrm{train}} + \alpha\mu^{\mathrm{test}} \geq \alpha\mu^{\mathrm{test}}$, we have

$$\alpha w_{\alpha}[a] = \frac{d(\alpha\mu^{\mathrm{test}})}{d\mu^{\alpha}}[a] \leq 1. \tag{7}$$

Therefore,

$$w_{\alpha}[a] \leq \frac{1}{\alpha}, \quad \alpha > 0. \tag{8}$$

When the importance weight $w[a]$ exists, the relative weight $w_{\alpha}[a]$ can be regarded as a bounded regularization of $w[a]$, which approaches $w[a]$ as $\alpha \to 0$. Based on $w_{\alpha}[a]$, we define the relative importance-weighted training error as follows:

$$L^{\mathrm{RIWOL}}(\theta) := \int_A w_{\alpha}[a] \left\| S^{\dagger}[a] - S_{\theta}[a] \right\|_U^2 \, \mathrm{d}\mu^{\mathrm{train}}[a]. \tag{9}$$

Note that $L^{\mathrm{RIWOL}}(\theta)$ is generally not an unbiased estimator of the expected test error, since w_{α} is defined with respect to the mixture measure μ^{α} rather than the training measure μ^{train}. Nevertheless, for any $\alpha > 0$, this objective is well defined due to $\mu^{\mathrm{test}} \ll \mu^{\alpha}$, and the relative importance weight is uniformly bounded by $1/\alpha$. This boundedness mitigates the high-variance issue that can arise in $L^{\mathrm{IWOL}}(\theta)$ when estimating the expected test error from finite samples, thereby improving the training stability and robustness. Hence, $L^{\mathrm{RIWOL}}(\theta)$ can be viewed as a well-defined alternative to $L^{\mathrm{IWOL}}(\theta)$ with bounded weights.

4.3 Relative importance weight estimation

In Eq. (9), the relative importance weight $w_\alpha[a]$ is unknown. Since the underlying measures μ^α and μ^{test} typically lack explicit and evaluable forms, computing the weight via Eq. (6) in function spaces is difficult. To address this, we formulate the estimation of relative importance weights as a binary classification problem in function spaces. Specifically, we first introduce the binary classification problem defined on function spaces and show the closed-form relationship between the optimal classifier and the relative importance weight. Subsequently, we propose to construct the classifier using kernel integral operators.

Classification problem for weight estimation. Let $D : \mathcal{A} \to (0, 1)$ be a binary classifier defined on a function space, i.e., a functional that takes functions as inputs. We use D to distinguish between input functions drawn from a mixture measure μ^α (labeled as 1) and those drawn from a test measure μ^{test} (labeled as 0). This binary classification task is formulated as the minimization of binary cross-entropy error $R(D)$:

$$R(D) = -\pi_\alpha \int_{\mathcal{A}} \log D[a] \, \mathrm{d}\mu^\alpha[a] - \pi_{\mathrm{test}} \int_{\mathcal{A}} \log\bigl(1 - D[a]\bigr) \, \mathrm{d}\mu^{\mathrm{test}}[a], \tag{10}$$

where π_α and π_{test} represent the prior probabilities of sampling from the mixture and test measures, respectively. By substituting $\mathrm{d}\mu^{\mathrm{test}}[a] = w_\alpha[a]\mathrm{d}\mu^\alpha[a]$ into Eq. (10), we can evaluate the risk $R(D)$ using an integral with respect to the mixture measure:

$$R(D) = -\int_{\mathcal{A}} \Bigl[\pi_\alpha \log D[a] + \pi_{\mathrm{test}} w_\alpha[a] \log\bigl(1 - D[a]\bigr)\Bigr] \, \mathrm{d}\mu^\alpha[a]. \tag{11}$$

The optimal functional $D^\dagger$ is obtained by minimizing the integrand independently for each $a \in \mathcal{A}$. Taking the derivative with respect to D and setting it to zero yields the first-order optimality condition:

$$-\frac{\pi_\alpha}{D^\dagger[a]} + \frac{\pi_{\mathrm{test}} w_\alpha[a]}{1 - D^\dagger[a]} = 0. \tag{12}$$

Solving for $D^\dagger[a]$, we identify the optimal functional as

$$D^\dagger[a] = \frac{\pi_\alpha}{\pi_\alpha + \pi_{\mathrm{test}} w_\alpha[a]}. \tag{13}$$

Since the second variation is strictly positive, this solution corresponds to a unique global minimum. Rearranging the above expression yields the closed-form relationship:

$$w_\alpha[a] = \frac{\pi_\alpha}{\pi_{\mathrm{test}}} \left(\frac{1}{D^\dagger[a]} - 1\right). \tag{14}$$

Learning the classifier with finite samples. In Eq. (14), the optimal functional $D^\dagger$ is unknown; therefore, we consider approximating it using neural networks.

Let D_ϕ be a neural network parameterized by ϕ. Given training samples $\{a_i^{\mathrm{train}}\}_{i=1}^I \sim \mu^{\mathrm{train}}$ and test samples $\{a_j^{\mathrm{test}}\}_{j=1}^J \sim \mu^{\mathrm{test}}$, we construct empirical mixture samples $\{a_m^\alpha\}_{m=1}^M$ by drawing each sample from the training samples with probability $1 - \alpha$ and from the test samples with probability α, respectively. We then learn ϕ by minimizing the empirical binary cross-entropy loss:

$$\hat{\phi} = \arg\min_\phi \hat{R}(D_\phi), \tag{15}$$

where the empirical error is given by

$$\hat{R}(D_\phi) = -\frac{1}{M + J} \left[\sum_{m=1}^M \log D_\phi[a_m^\alpha] + \sum_{j=1}^J \log\Bigl(1 - D_\phi[a_j^{\mathrm{test}}]\Bigr)\right]. \tag{16}$$

Given the optimized parameters $\hat{\phi}$, we construct the relative importance weight estimator based on Eq. (14):

$$\hat{w}_\alpha[a] := \frac{M}{J} \left(\frac{1}{D_{\hat{\phi}}[a]} - 1\right). \tag{17}$$

These weights are incorporated into the training objective $L^{\mathrm{RIWOL}}(\theta)$ to mitigate the probability measure shifts between training and test phases.

Classifier design in function spaces. Since the binary classifier D takes functions as inputs, a straightforward approach is to discretize the function on a finite grid and then apply a standard neural network to it. However, this approach has limitations, for example, the learned neural network model typically may not generalize well across different discretizations. To overcome this limitation, we formulate the classifier as a functional acting on an infinite-dimensional function space. The framework is based on a kernel integral operator [8, 34, 35], allowing operations on inputs to be defined directly in the function spaces, ensuring discretization invariance.

We assume that input functions $a \in \mathcal{A}$ are $\mathbb{R}^{d_a}$-valued and defined on a bounded domain $\Omega \subset \mathbb{R}^d$. The proposed architecture $D_\phi : \mathcal{A} \to (0, 1)$ is composed of a lifting layer, iterative kernel integral layers, and a projection layer that maps the function to a scalar. The formulation is given by

$$D_\phi := \sigma_{\text{out}} \circ \mathcal{Q} \circ \mathcal{G} \circ \mathcal{K}_{T-1} \circ \cdots \circ \mathcal{K}_0 \circ \mathcal{P}. \tag{18}$$

The components are defined as follows:

- $\mathcal{P}$: A local lifting operator that maps the input $a(\boldsymbol{x}) \in \mathbb{R}^{d_a}$ to a hidden representation $v_0(\boldsymbol{x}) \in \mathbb{R}^{d_0}$ for any spatial coordinate $\boldsymbol{x} \in \Omega$.
- $\mathcal{K}_t$: A sequence of T iterative kernel integral layers. For $t = 0, \cdots, T-1$, the update $v_{t+1} = \mathcal{K}_t[v_t]$ is defined as

$$v_{t+1}(\boldsymbol{x}) = \sigma_t \left(W_t v_t(\boldsymbol{x}) + \int_\Omega \kappa_t(\boldsymbol{x}, \boldsymbol{y}) v_t(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} + b_t(\boldsymbol{x}) \right),$$

  where $\boldsymbol{y} \in \Omega$ is the integration variable, $W_t \in \mathbb{R}^{d_{t+1} \times d_t}$ is a linear transformation, $\kappa_t : \{v_t \in \mathbb{R}^{d_t}\} \to \{v_{t+1} \in \mathbb{R}^{d_{t+1}}\}$ is a learnable kernel function parameterized by a neural network, $b_t(\boldsymbol{x}) \in \mathbb{R}^{d_{t+1}}$ is a bias function, and σ_t is a non-linear activation function.
- $\mathcal{G}$: A global average pooling operator that aggregates spatial information, reducing the function v_T to a finite-dimensional vector: $\mathcal{G}[v_T] = \frac{1}{|\Omega|} \int_\Omega v_T(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$.
- $\mathcal{Q}$: A linear layer that maps the finite-dimensional global feature vector $\mathcal{G}[v_T] \in \mathbb{R}^{d_T}$ to a scalar.
- σ_{out}: An activation function, which maps a scalar to $(0, 1)$.

4.4 Training procedure

We estimate parameters of the neural operator θ and parameters of the binary classifiers ϕ in two steps:

Step 1: Binary classifier training. We optimize parameters ϕ of binary classifiers D_ϕ by minimizing the error defined by Eq. (15) using the mixture samples $\{a_m^\alpha\}_{m=1}^M$ and the test samples $\{a_j^{\text{test}}\}_{j=1}^J$.

Step 2: Relative importance-weighted neural operator training. We estimate the neural operator parameters θ by minimizing the following relative importance-weighted empirical loss:

$$\hat{L}^{\text{RIWOL}}(\theta) = \frac{1}{I} \sum_{i=1}^{I} \left(\hat{w}_\alpha[a_i^{\text{train}}] \left\| u_i^{\text{train}} - \mathcal{S}_\theta[a_i^{\text{train}}] \right\|_{\mathcal{U}}^2 \right). \tag{19}$$

This objective is optimized using the training data $\{(a_i^{\text{train}}, u_i^{\text{train}})\}_{i=1}^I$. The sample weight $\hat{w}_\alpha[a_i^{\text{train}}]$ in Eq. (19) is computed via Eq. (17) using the classifier learned in Step 1. The overall training procedure is provided in Appendix A.

In our experiments, for Step 2, we employed FNO [8] and DeepONet [7] to instantiate the neural operator $\mathcal{S}_\theta$. Note that $\mathcal{S}_\theta$ is architecture-agnostic and can be implemented by various architectures, such as Graph Neural Operators [36, 34] and Convolutional Neural Operators [37]. For the binary classifier D_ϕ in Step 1, we parameterized the kernel integral layers $\mathcal{K}_t$ using spectral convolution layers via the fast Fourier transform (FFT), adhering to the FNO architecture [8]. However, the formulation of D_ϕ is general: $\mathcal{K}_t$ can be flexibly instantiated by other integral approximation mechanisms, such as graph message passing [36, 34] or linear attention blocks [38].

5 Experiments

In this section, we demonstrate the effectiveness of our proposed IWOL, using simulation data of PDEs. We focus on tasks that predict solution functions when we are given initial conditions as input functions.

5.1 Datasets

We evaluated our proposed method on commonly used PDEs, including the advection equation, Burgers equation, Darcy flow equation, shallow-water equation, incompressible Navier–Stokes (NS) equation, Kortewegde Vries equation, and CahnHilliard equation. The first five datasets were generated by the numerical simulations used in [39] and are publicly available [40]. The remaining two datasets were generated by the authors through numerical simulations. Detailed dataset information, including the PDE systems, data resolutions, and sample sizes, is provided in Appendices B.1 and D.2.

Construction of probability measure shifts. To simulate probability measure shifts, we partitioned each dataset into two clusters using the fast Fourier transform (FFT) [41] for feature extraction and the k-means algorithm [42] for clustering. Specifically, after extracting spectral features via FFT, we divided the samples into two clusters denoted as A and B using the k-means algorithm. The data split was constructed as follows: **Training dataset:** Included all samples from cluster A and a random 10% subset of cluster B (to ensure overlap between the training and test probability measures). **Test dataset:** Consisted of 45% of the samples from cluster B that were not used for training. **Unlabeled dataset:** Comprised of the remaining 45% of samples from cluster B. Specifically, only the input functions of these samples were used for importance estimation. For the KdV and Cahn–Hilliard datasets generated in this study, probability measure shifts were constructed by changing the data-generating mechanism of the initial states between the training and test phases, as detailed in Appendix D. The physical motivation for constructing probability measure shifts is further discussed in Appendix E.1. The effect of overlap between training and test probability measures is further investigated in Appendix E.2.

5.2 Experimental setup

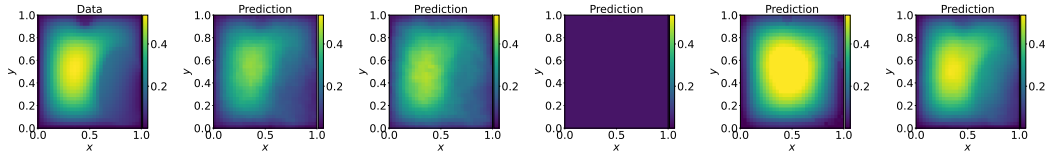
We employed FNO [8] as the architecture for the neural operator $\mathcal{S}_\theta[a]$. The model was implemented in PyTorch [43] and trained using the Adam optimizer [44]. Following the standard FNO configuration [8], we set the channel width to 32, retained 12 Fourier modes along each dimension, and used GELU activations [45]. The model was trained for 600 epochs with an initial learning rate of 10^{-3}, which was reduced by a factor of two every 100 epochs. We also evaluated DeepONet [7] as an alternative implementation of $\mathcal{S}_\theta[a]$; its architectural details and training configuration are provided in Appendix B.2. For importance weight estimation, we trained a binary classifier D_ϕ as defined in Eq. (18). The classifier received samples from both the training set and the unlabeled set as inputs, and was optimized using the empirical binary cross-entropy objective $\hat{R}(D_\phi)$ in Eq. (16). Detailed architectural specifications of D_ϕ are provided in Appendix B.2.

5.3 Baselines

We compared the proposed method with standard supervised operator learning baselines, including FNO [8] and DeepONet [7], which were trained by minimizing Eq. (1). In addition to these supervised baselines, we compared with three baselines that used unlabeled test inputs: (i) SSOL, which was first trained on labeled training pairs, generated pseudo-labels for unlabeled test inputs, and then retrained on the combined dataset; (ii) KDE, which estimated importance weights via finite-dimensional density estimation on discretized inputs; and (iii) FNN, which used a feedforward neural network as classifier for weight estimation. KDE, FNN, and our proposed method all estimate $\hat{w}_\alpha[a]$ using a mixing parameter of $\alpha = 0.5$, and are trained by minimizing the relative importance-weighted training error $\hat{L}^{\mathrm{RIWOL}}(\theta)$, as defined in Eq. (19). Detailed implementations are provided in Appendix B.3. Analyses of the mixing parameter α, the amount of unlabeled data, and the distribution of the estimated weights are provided in Appendix E.3, E.4, and E.5. Further discussions of classifier architectures are provided in Appendix E.6.

Table 1: Performance comparison under probability measure shifts using the FNO architecture. We report the average RMSE and standard deviation over five runs. All values are multiplied by 10^3 in order to improve the readability. The best results are highlighted in bold.

Method	Advection	Burgers'	Darcy flow	Shallow-water	NS ($\nu = 10^{-3}$)	NS ($\nu = 10^{-4}$)
FNO	6.4 ± 0.7	5.4 ± 1.2	30.5 ± 0.9	8.2 ± 0.7	61.7 ± 3.1	207.3 ± 11.5
FNO-SSOL	6.8 ± 1.8	7.7 ± 2.4	28.5 ± 2.7	19.4 ± 4.2	73.2 ± 11.3	159.1 ± 19.5
FNO-KDE	5.7 ± 1.1	4.7 ± 1.9	181.0 ± 14.9	10.1 ± 2.2	741.0 ± 23.7	1041.4 ± 43.2
FNO-FNN	3.8 ± 0.7	10.0 ± 1.9	51.6 ± 3.1	7.9 ± 0.8	30.8 ± 5.5	179.0 ± 9.4
FNO-Ours	$\mathbf{2.6 \pm 0.5}$	$\mathbf{1.6 \pm 1.1}$	$\mathbf{20.1 \pm 0.4}$	$\mathbf{5.6 \pm 0.9}$	$\mathbf{15.8 \pm 4.4}$	$\mathbf{114.3 \pm 9.9}$



(a) Ground truth (b) FNO (c) FNO-SSOL (d) FNO-KDE (e) FNO-FNN (f) FNO-Ours

Figure 1: Visualization of Darcy flow solutions predicted by baselines, and our method on the test set (32×32 resolution). All methods are trained using the same FNO architecture.

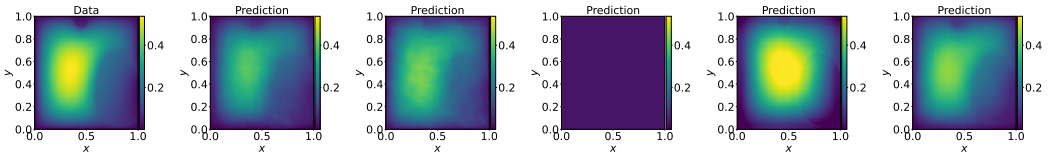
5.4 Results

Table 1 reports the root mean square error (RMSE) between the true and predicted solutions for the proposed method (Ours) and the baseline methods on the test dataset. For all methods, including SSOL, FNN, KDE, and Ours, the neural operator is implemented using FNO. We report the average RMSE and standard deviation over five trials. Bold values indicate the best performance. The corresponding results obtained using DeepONet are provided in the Appendix C. In all cases, the proposed method outperforms the baselines. This indicates that under the setting of probability measure shifts, the proposed method enables the model to leverage unlabeled data (input functions) from the test measure, allowing the model to focus more on training samples that closely align with the test samples during the training process, resulting in better performance. The performance of SSOL was worse than that of standard operator learning methods in most cases. This is because SSOL was pre-trained on the training measure. Due to the probability measure shifts, the predictions of the pre-trained model on the test measure typically contain errors. Using these erroneous predictions as ground truth to retrain the model leads to error accumulation, resulting in performance that is inferior to that of standard operator learning methods. The KDE and FNN variants achieved better performance than standard operator learning methods, but they were still not as good as the proposed approach. This suggests that importance weighting is effective for operator learning. However, naively representing functions as finite-dimensional vectors to estimate importance weights makes the weights dependent on a fixed discretization grid, which can lead to inaccurate estimates. In contrast, our method relies on a binary classifier that takes functions as inputs. The classifier is defined via functionals modeled by kernel integral operators, so it does not depend on a fixed grid and can estimate importance weights for functions more accurately. Furthermore, we evaluated the zero-shot super-resolution setting, where the model is trained on lower-resolution data (32×32) but evaluated on higher-resolution data (64×64). Table 2 presents RMSE on Darcy flow equation over the test set. Figure 1 visualizes the solutions predicted by FNO, SSOL, KDE, FNN, and Ours. The super-resolution results are visualized in Figure 2. Collectively, these figures demonstrate that our method captures the physical behavior more accurately than the baselines, even under probability measure shifts.

Table 2: Super-resolution performance on the Darcy flow equation. All values are multiplied by 10^3 in order to improve the readability.

Method	RMSE
FNO	49.5 ± 0.5
FNO-SSOL	44.2 ± 1.7
FNO-KDE	176.9 ± 37.1
FNO-FNN	58.0 ± 1.1
FNO-Ours	$\mathbf{38.3 \pm 0.3}$

To evaluate the discretization invariance of the weight estimator, we trained the classifier D_ϕ using training inputs and unlabeled test inputs discretized at a resolution of 32×32. After training, we fixed all classifier parameters. We then discretized the same training input functions used for neural operator training at resolutions of 16×16, 32×32, and 64×64, and used the same classifier to estimate the corresponding importance weights, $\hat{w}_{16}$, $\hat{w}_{32}$, and $\hat{w}_{64}$, according to Eq. 17. For



(a) Ground truth (b) FNO (c) FNO-SSOL (d) FNO-KDE (e) FNO-FNN (f) FNO-Ours

Figure 2: Visualization of Darcy flow solutions in the super-resolution setting: models are trained on low-resolution data (32×32) and evaluated on high-resolution data (64×64). Solutions are predicted by baselines and our method. All methods are trained using the same FNO architecture.

Table 3: Discretization invariance of importance weight estimation on the Darcy flow equation. The classifier is trained at a resolution of 32×32, and importance weights are estimated at resolutions of 16×16, 32×32, and 64×64. Neural operator training and evaluation are performed at 32×32. Here, $\hat{w}_i$ denotes the importance weights estimated at resolution $i \times i$, and $\rho_{i,j}$ denotes the Pearson correlation between $\hat{w}_i$ and $\hat{w}_j$.

Method	Pearson correlation ↑			RMSE ($\times 10^3$) ↓		
	$\rho_{64,32}$	$\rho_{64,16}$	$\rho_{32,16}$	$\hat{w}_{64}$	$\hat{w}_{32}$	$\hat{w}_{16}$
FNO-KDE	0.07 ± 0.13	-0.01 ± 0.22	0.03 ± 0.06	179.7 ± 13.3	181.0 ± 14.9	177.2 ± 12.8
FNO-FNN	0.30 ± 0.04	0.19 ± 0.06	0.49 ± 0.12	97.3 ± 3.8	51.6 ± 3.1	57.1 ± 2.9
FNO-Ours	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.96 \pm 0.03}$	$\mathbf{19.6 \pm 0.3}$	$\mathbf{20.1 \pm 0.4}$	$\mathbf{20.4 \pm 0.7}$

all three sets of weights, the neural operator training and test data were fixed at a resolution of 32×32, and all other training settings were kept unchanged. Thus, the only factor varied in this experiment was the input resolution used for weight estimation. We report the Pearson correlations between the weights estimated at different resolutions, together with the test RMSE of the neural operators trained using the corresponding weights. All results are reported as the mean and standard deviation over five runs. Our method achieved cross-resolution weight correlations ranging from 0.96 to 0.99, while the corresponding test RMSE remains stable between 19.6 and 20.4. These results show that, although the classifier was trained only at a resolution of 32×32, it produced highly consistent importance weights and maintains stable downstream predictive performance when evaluated at the resolutions of 16×16 and 64×64. In contrast, the finite-dimensional FNN- and KDE-based estimators exhibit substantially lower correlations between weights estimated at different resolutions, and their downstream predictive performance is also considerably worse than that of the proposed method. Additional experimental results, including an analysis of computational cost, a comparison between standard importance weighting (IW) and relative importance weighting (RIW) under different levels of overlap, and comparisons with test-time adaptation methods, are provided in Appendices E.7, E.8, and E.9, respectively.

6 Conclusions

We have proposed an importance-weighted operator learning method to improve predictive performance under probability measure shifts between the training and test phases. By introducing relative importance weights defined with respect to a mixture measure, our method ensures the existence and boundedness of the weights. To estimate these weights, we employ binary classifiers parameterized by kernel integral operators, enabling discretization-invariant weight estimation in function space. Experiments across multiple PDE systems and neural operator architectures demonstrate the effectiveness of the proposed method under probability measure shifts. Although our method performs well, its effectiveness may be limited when very few unlabeled test samples (e.g., two or three) are available, making reliable importance weight estimation challenging. This limitation is common to distribution adaptation methods relying on unlabeled test data. Combining our framework with meta-learning may help address this issue [46]. Another limitation is its dependence on the overlap between training and test measures. Although relative importance weights are guaranteed to exist and remain bounded, the weighted training objective is still evaluated under the training measure. Consequently, insufficient overlap may limit performance improvements, as demonstrated in Appendix E.2. The broader impacts of our work are discussed in Appendix F.

Acknowledgments

This work was supported by JST SPRING, Grant Number JPMJSP2140, JST ASPIRE Grant Number JPMJAP2329 and JST CREST Grant Number JPMJCR24Q5, Japan.

References

[1] Chia-Ch'iao Lin and Lee A Segel. *Mathematics Applied to Deterministic Problems in the Natural Sciences*. SIAM, 1988.

[2] Peter Lynch. The origins of computer weather prediction and climate modeling. *Journal of Computational Physics*, 227(7):3431–3444, 2008.

[3] Audrey Olivier, Michael D Shields, and Lori Graham-Brady. Bayesian neural networks for uncertainty quantification in data-driven materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 386:114079, 2021.

[4] Oksana A. Chkrebtii, David A. Campbell, Ben Calderhead, and Mark A. Girolami. Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, 11(4):1239–1267, 2016.

[5] Michael Athanasopoulos, Hassan Ugail, and Gabriela González Castro. Parametric design of aircraft geometry using partial differential equations. *Advances in Engineering Software*, 40(7):479–486, 2009.

[6] Alan R. Parkinson, Richard J. Balling, and John D. Hedengren. *Optimization Methods for Engineering Design*. Brigham Young University, 2013.

[7] 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.

[8] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[9] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric PDEs. *The SMAI Journal of Computational Mathematics*, 7:121–157, 2021.

[10] Ravi G Patel, Nathaniel A Trask, Mitchell A Wood, and Eric C Cyr. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373:113500, 2021.

[11] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[12] 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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2806–2823. PMLR, 2023.

[13] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[14] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[15] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A Brief Review of Domain Adaptation. In *Advances in Data Science and Information Engineering*, pages 877–894, Cham, 2021. Springer International Publishing.

[16] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189. PMLR, 2015.

[17] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, 2017.

[18] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[19] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7523–7532. PMLR, 2019.

[20] Nan Lu, Tianyi Zhang, Tongtong Fang, Takeshi Teshima, and Masashi Sugiyama. Rethinking importance weighting for transfer learning. In *Federated and Transfer Learning*, pages 185–231. Springer, 2022.

[21] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

[22] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul Von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[23] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems*, 20:1433–1440, 2007.

[24] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10:1391–1445, 2009.

[25] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems*, 19:601–608, 2006.

[26] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(75):2137–2155, 2009.

[27] Jakiw Pidstrigach, Youssef Marzouk, Sebastian Reich, and Sven Wang. Infinite-dimensional diffusion models. *Journal of Machine Learning Research*, 25(414):1–52, 2024.

[28] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Neural Computation*, 25(5):1324–1370, 2013.

[29] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[30] Olav Kallenberg. *Foundations of Modern Probability*. Springer, 3 edition, 2021.

[31] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. *Advances in Neural Information Processing Systems*, 23:442–450, 2010.

[32] Fengpei Li, Henry Lam, and Siddharth Prusty. Robust importance weighting for covariate shift. In *International Conference on Artificial Intelligence and Statistics*, pages 352–362. PMLR, 2020.

[33] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate Shift by Kernel Mean Matching. In *Dataset Shift in Machine Learning*, pages 131–160. The MIT Press, 2008.

[34] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

[35] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[36] Masanobu Horie and Naoto Mitsume. Physics-embedded neural networks: Graph neural PDE solvers with mixed boundary conditions. *Advances in Neural Information Processing Systems*, 35:23218–23229, 2022.

[37] Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of PDEs. *Advances in Neural Information Processing Systems*, 36:77187–77200, 2023.

[38] Shuhao Cao. Choose a transformer: Fourier or Galerkin. *Advances in Neural Information Processing Systems*, 34:24924–24940, 2021.

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

[40] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench Datasets. DaRUS, 2022. DOI: 10.18419/darus-2986.

[41] Pierre Duhamel and Martin Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal Processing*, 19(4):259–299, 1990.

[42] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8024–8035, 2019.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[45] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[46] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2022.

[47] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[48] Bernard W Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 2018.

[49] Mohamed Mossad, Irina Strelnikova, Robin Wing, Gerd Baumgarten, and Michael Gerding. Spectral variability of gravity-wave kinetic and potential energy at 69° N: a seven-year lidar study. *Atmospheric Chemistry and Physics*, 25(21):14839–14864, 2025.

[50] Augustine Kong, Jun S Liu, and Wing Hung Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

[51] Kazuki Adachi, Shin'ya Yamaguchi, Atsutoshi Kumagai, and Tomoki Hamagami. Test-time adaptation for regression by subspace alignment. In *International Conference on Learning Representations*, 2025.

A Training Algorithm

Algorithm 1 Training procedure.

Input: Training data $\{(a_i^{\mathrm{train}}, u_i^{\mathrm{train}})\}_{i=1}^I$, unlabeled test data $\{a_j^{\mathrm{test}}\}_{j=1}^J$, mini-batch size B_{clf} for training classifier, B_{ol} for training neural operator, mixing parameter α.
Output: Trained neural operator parameters θ.

Initialize neural operator parameters θ and binary classifier parameters ϕ.
/* Relative importance weight estimation */
Generate mixture samples $\{a_m^\alpha\}_{m=1}^M$ from the training and unlabeled test samples using α.
Construct the classifier training dataset $\{(a_m^\alpha, 1)\}_{m=1}^M \cup \{(a_j^{\mathrm{test}}, 0)\}_{j=1}^J$.
repeat
  Randomly sample a mini-batch of indices $\mathcal{B}_{\mathrm{clf}} \subset \{1, \dots, M+J\}$ with size B_{clf}.
  Predict binary probabilities for sampled inputs using the classifier D_ϕ.
  Compute the empirical binary cross-entropy loss $\hat{R}(D_\phi)$ via Eq. (16).
  Update parameters ϕ using the gradient of $\hat{R}(D_\phi)$ via a stochastic gradient method.
until End condition is satisfied.
Compute relative importance weights $\hat{w}_i = \hat{w}_\alpha[a_i^{\mathrm{train}}]$ for all $i \in \{1, \dots, I\}$ via Eq. (17) using the optimized ϕ.
/* Weighted operator learning */
repeat
  Randomly sample a mini-batch of indices $\mathcal{B}_{\mathrm{ol}} \subset \{1, \dots, I\}$ with size B_{ol}.
  Predict solutions $\hat{u}^{\mathrm{train}}$ for the sampled inputs using neural operator $\mathcal{S}_\theta$.
  Compute the relative importance-weighted loss $\hat{L}^{\mathrm{RIWOL}}$ via Eq. (19) using weights $\{\hat{w}_i\}_{i \in \mathcal{B}_{\mathrm{ol}}}$.
  Update parameters θ using the gradient of $\hat{L}^{\mathrm{RIWOL}}$ via a stochastic gradient method.
until End condition is satisfied.

B Details of experiments

This section provides a summary of the detailed experimental settings for all experiments, including those described in Section 5.

B.1 Datasets

We evaluated our proposed method on commonly used PDEs: the advection equation, Burgers equation, Darcy flow equation, shallow-water equation, and incompressible Navier–Stokes (NS) equation. All datasets were generated by numerical simulations used in [39] and are publicly available [40]. A summary of the datasets is provided in Table 4.

Advection equation. The linear advection equation for function $q(x,t) \in L^2((0,1) \times (0,2]; \mathbb{R})$ is given by

$$\partial_t q(x,t) + \beta \partial_x (q(x,t)/2) = 0, \quad q(x,0) = q_0(x),$$

where q_0 is the initial condition and $\beta = 2$ is an advection speed.

Burgers equation. The non-linear Burgers equation for the function $q(x,t) \in L^2((0,1) \times (0,2]; \mathbb{R})$ is given by

$$\partial_t q(x,t) + \partial_x (q^2(x,t)/2) = \nu \partial_{xx} q(x,t), \quad q(x,0) = q_0(x),$$

where q_0 is the initial condition and $\nu = 0.4$ is the diffusion coefficient.

Darcy flow equation. The 2D Darcy flow equation for function $q(x)$ is given by

$$-\nabla \cdot (k(x) \nabla q(x)) = f(x), \quad x \in (0,1)^2,$$

where $k(x)$ is the diffusion coefficient and $f(x) = 1$ is the forcing function.

Table 4: Overview of datasets with number of spatial dimensions M, spatial resolution N_s, and number of samples.

PDE	M	N_s	Number of samples
Advection	1	128	10,000
Burgers	1	128	10,000
Darcy flow	2	32×32	10,000
Shallow-water	2	32×32	1,000
Navier–Stokes ($\nu = 10^{-3}$)	2	32×32	5,000
Navier–Stokes ($\nu = 10^{-4}$)	2	32×32	10,000

Shallow-water equation. The 2D shallow-water equations are a special case of the Navier–Stokes equations. The system in conservative form is given by

$$\begin{aligned} \partial_t h + \partial_x(hq) + \partial_y(hv) &= 0, \\ \partial_t(hq) + \partial_x\left(hq^2 + \frac{1}{2}g_r h^2\right) + \partial_y(hqv) &= -g_r h\, \partial_x b, \\ \partial_t(hv) + \partial_y\left(hv^2 + \frac{1}{2}g_r h^2\right) + \partial_x(hqv) &= -g_r h\, \partial_y b. \end{aligned}$$

where $(x, y) \in [-2.5, 2.5]^2$, $t \in [0, 1]$. The variable $h(x, y, t)$ represents the fluid height, (q, v) are the horizontal velocity components, and b describes a spatially varying bathymetry. hq, hv can be interpreted as the directional momentum components, and g_r is the gravitational acceleration.

Navier–Stokes equation. The 2D incompressible Navier–Stokes equation is considered on the unit torus in vorticity form:

$$\partial_t \omega(x,t) + p(x,t) \cdot \nabla \omega(x,t) = \nu \nabla^2 \omega(x,t) + f(x), \nabla \cdot p(x,t) = 0, \quad \omega(x,0) = \omega_0(x).$$

Here, $\omega(x, t)$ denotes the vorticity field at spatial location x and time t, and $\omega_0(x)$ is the initial vorticity. The vector field $p(x, t)$ represents the velocity field, which satisfies the incompressibility constraint $\nabla \cdot p(x, t) = 0$. The parameter $\nu \in \mathbb{R}_+$ is the kinematic viscosity, and $f(x)$ denotes an external forcing function defined by $f(x) = 0.1\,(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$. The viscosity is varied across different experimental settings, with $\nu \in \{10^{-3}, 10^{-4}\}$.

B.2 Detailed experimental setup

We provide here the detailed architectural and training configurations used in our experiments. For the FNO implementation of $\mathcal{S}_\theta[a]$, we followed the standard configuration described in the main text, using a channel width of 32, 12 Fourier modes along each dimension, and GELU activations. The model was trained for 600 epochs, with the learning rate being reduced by a factor of two every 100 epochs.

For the DeepONet implementation of $\mathcal{S}_\theta[a]$, we employed a standard unstacked architecture in which both the branch and trunk networks were parameterized by multi-layer perceptrons (MLPs). Each network consisted of three layers with 200 hidden units and $\tanh$ activations. The DeepONet model was trained for 10,000 epochs using the same optimizer and initial learning rate as the FNO model.

The binary classifier D_ϕ used for importance weight estimation was constructed as follows. The lifting operator $\mathcal{P}$ was implemented as a linear layer with 16 hidden units. The kernel integral component $\mathcal{K}_t$ consisted of two Fourier integral operator layers with a channel width of 10, five Fourier modes in each dimension, and GELU activations. For feature aggregation, $\mathcal{G}$ was implemented as mean pooling over spatial features. Finally, the projection layer $\mathcal{Q}$ was parameterized by an MLP with 16 hidden units followed by a sigmoid activation σ_{out}.

To train D_ϕ, we constructed a binary classification dataset from samples in the labeled training set and the unlabeled set. The combined dataset was randomly split into 90% training and 10% validation subsets. The classifier was trained for at most 300 epochs, with early stopping based on validation performance.

B.3 Detailed implementation of the baselines

We provide implementation details for the additional baselines used in the experiments.

Table 5: Performance comparison under probability measure shifts using the DeepONet architecture. We report the average RMSE and standard deviation on the test set over five runs. All values are multiplied by 10^3 in order to improve the readability. The best results are highlighted in bold.

Method	Advection	Burgers'	Darcy flow	Shallow-water	NS ($\nu = 10^{-3}$)	NS ($\nu = 10^{-4}$)
DeepONet	7.2 ± 0.9	4.9 ± 0.8	27.4 ± 0.9	6.7 ± 0.5	58.3 ± 3.9	233.7 ± 9.8
DeepONet-SSOL	6.8 ± 1.8	7.7 ± 2.4	29.9 ± 2.7	9.9 ± 3.1	61.0 ± 7.6	193.9 ± 21.1
DeepONet-KDE	4.6 ± 1.4	4.3 ± 1.2	153.6 ± 19.7	7.3 ± 1.0	674.3 ± 36.3	1125.1 ± 39.8
DeepONet-FNN	5.4 ± 0.9	6.8 ± 1.9	26.0 ± 0.6	5.6 ± 0.8	49.1 ± 1.7	163.6 ± 16.5
DeepONet-Ours	$\mathbf{2.7 \pm 0.9}$	$\mathbf{1.9 \pm 0.6}$	$\mathbf{19.8 \pm 0.5}$	$\mathbf{5.1 \pm 0.3}$	$\mathbf{27.8 \pm 5.1}$	$\mathbf{129.3 \pm 14.2}$

For the SSOL baseline, we adopted a self-training strategy. The neural operator was first trained on the labeled training dataset by minimizing the supervised objective in Eq. (1). The trained model was then used to generate pseudo-labels for the unlabeled inputs. Finally, the model was retrained on the combined labeled and pseudo-labeled dataset.

For the KDE baseline, each discretized input function was treated as a finite-dimensional vector. We first applied principal component analysis [47] for feature extraction. Kernel density estimation [48] was then used to estimate the training density and the test density separately, and their ratio was used to construct the importance weights.

For the FNN baseline, the classifier D_ϕ was implemented as a feedforward neural network and trained using the empirical binary cross-entropy objective in Eq. (16). The network consisted of four fully connected layers with 40 units per layer and ReLU activations. The trained classifier was then used to estimate the importance weights.

For all importance-weighted methods, including KDE, FNN, and the proposed method, we used the same mixing parameter $\alpha = 0.5$ to estimate $\hat{w}_\alpha[a]$. These methods were trained by minimizing the empirical relative importance-weighted objective $\hat{L}^{\mathrm{RIWOL}}(\theta)$ in Eq. (19).

C Evaluation results for DeepONet

Table 5 reports the RMSE between the true and predicted solutions for the proposed method (Ours) and the baseline methods on the test dataset. For all the methods, including SSOL, FNN, KDE, and Ours, the neural operator is implemented using DeepONet. We report the average RMSE and standard deviation over five trials. Bold values indicate the best performance.

D Additional experiments with probability measure shifts induced by data-generating mechanisms

In Section 5, probability measure shifts were induced by clustering the input functions. In this appendix, we provide an additional set of experiments in which the probability measure shifts are induced directly by changing the data-generating mechanism of the input functions.

D.1 Construction of probability measure shifts

We generate each initial condition from a random sine basis expansion:

$$a(x) = \sum_{j=1}^{K} b_j \sin\left(\frac{\pi k_j x}{X}\right),$$

where $K = 4$, and the amplitude coefficients b_j and frequency parameters k_j are independently sampled for each input function.

For the training set, we fix the sampling distributions as

$$b_j \sim \mathrm{Unif}(-0.5, 0.5), \quad k_j \sim \mathrm{Unif}(0.1, 1.0).$$

We then construct three test settings by changing the sampling distributions of b_j and/or k_j. These settings are summarized in Table 6. This design allows us to systematically evaluate whether the proposed method remains effective when the shifts arise from changes in the input-function generation mechanism.

Table 6: Parameter distributions used for generating the test datasets under different probability measure shifts.

Setting	b_j distribution	k_j distribution
Setting 1	$\mathrm{Unif}(-0.5, 0.5)$	$\mathrm{Unif}(0.9, 1.9)$
Setting 2	$\mathrm{Unif}(0, 1.0)$	$\mathrm{Unif}(0.9, 1.9)$
Setting 3	$\mathrm{Unif}(0, 1.0)$	$\mathrm{Unif}(0.1, 1.0)$

D.2 Datasets

We evaluated our proposed method on two one-dimensional nonlinear PDEs: the Korteweg–de Vries (KdV) equation and the Cahn–Hilliard (CH) equation. For both PDEs, the input function is the initial condition generated according to the procedure described in Appendix D.1, and the corresponding output functions were obtained by numerically solving the PDE. A summary of the simulation settings is provided below.

Korteweg–de Vries equation. The one-dimensional Korteweg–de Vries (KdV) equation for the function $q(x,t) \in L^2([0,10] \times [0,0.5]; \mathbb{R})$ is considered in the Hamiltonian form

$$\partial_t q(x,t) = \partial_x \left(3q^2(x,t) + \partial_{xx} q(x,t)\right), \quad q(x,0) = q_0(x),$$

where q_0 is the initial condition.

Cahn–Hilliard equation. The one-dimensional Cahn–Hilliard (CH) equation for the function $q(x,t) \in L^2([0,1] \times [0,0.05]; \mathbb{R})$ is considered as follows:

$$\partial_t q(x,t) = \partial_{xx} \left(-q(x,t) + q^3(x,t) - \gamma \partial_{xx} q(x,t)\right), \quad q(x,0) = q_0(x), \tag{20}$$

where $\gamma = 5 \times 10^{-4}$ and q_0 denotes the initial condition.

For each sampled initial condition, the spatial domain was uniformly discretized using $N_s = 64$ grid points. The resulting discretized PDE system was then solved using the Dormand–Prince method with adaptive time-stepping, as implemented in SciPy. Unless otherwise specified, we generated 2000 labeled training samples, 2000 test samples, and 2000 unlabeled samples. The unlabeled samples were used solely for estimating importance weights, and their corresponding solutions were not used during operator training.

D.3 Experimental setup and baselines

We employed DeepONet [7] as the architecture for the neural operator $\mathcal{S}_\theta[a]$. The architectural details, optimization settings, and baseline implementations were kept consistent with those described in Sections 5.2 and 5.3. The only difference is that the probability measure shifts in this appendix were constructed by changing the data-generating mechanism of the input functions, as described in Appendix D.1. For the importance-weighted methods, including KDE, FNN, and the proposed method, we used the same mixing parameter $\alpha = 0.5$.

D.4 Results

Table 7 reports the RMSE of different methods on the test datasets for the KdV and Cahn–Hilliard (CH) equations under probability measure shifts induced by changes in the input-function generation mechanism. For all implemented methods, the neural operator is implemented using DeepONet. We report the average RMSE and standard deviation over five trials, with bold values indicating the best performance. Across all reported settings, the proposed method outperforms the baselines. This indicates that the proposed functional weight estimator remains effective when the probability measure shifts are induced by explicit changes in the generative parameters of the input functions. These results complement the clustering-based experiments in Section 5.4. They show that the proposed method is not limited to shifts induced by clustering the input functions, but also remains effective under probability measure shifts induced by changes in the input-function generation mechanism.

Table 7: Performance comparison under probability measure shifts on the KdV and CH equations using the DeepONet architecture. We report the test RMSE averaged over five runs, with standard deviations. All values are multiplied by 10^3 in order to improve the readability. The training parameter distributions are fixed as $(b_j, k_j) \sim (\mathrm{Unif}(-0.5, 0.5), \mathrm{Unif}(0.1, 1.0))$, and the test parameter distributions for Settings 1–3 are summarized in Table 6. The best result in each column is highlighted in bold.

(a) KdV equation

Method	Setting 1	Setting 2	Setting 3
DeepONet	44.0 ± 1.0	879.8 ± 3.7	109.8 ± 6.2
DeepONet-SSOL	41.4 ± 3.3	1074.7 ± 2.8	144.7 ± 4.8
DeepONet-KDE	37.4 ± 0.4	992.1 ± 3.6	320.6 ± 4.1
DeepONet-FNN	36.7 ± 0.8	1162.7 ± 8.2	117.0 ± 3.5
DeepONet-Ours	$\mathbf{34.6 \pm 1.1}$	$\mathbf{842.4 \pm 3.8}$	$\mathbf{103.0 \pm 1.5}$

(b) CH equation

Method	Setting 1	Setting 2	Setting 3
DeepONet	50.9 ± 2.0	231.4 ± 7.9	32.7 ± 1.2
DeepONet-SSOL	35.2 ± 5.3	231.6 ± 13.6	55.3 ± 1.9
DeepONet-KDE	62.3 ± 2.6	250.4 ± 3.6	28.6 ± 0.8
DeepONet-FNN	43.1 ± 1.9	696.0 ± 9.8	27.7 ± 1.6
DeepONet-Ours	$\mathbf{26.8 \pm 2.9}$	$\mathbf{128.6 \pm 7.0}$	$\mathbf{25.6 \pm 1.4}$

E Additional experimental results

E.1 Spectral analysis of probability measure shifts

The physical motivation for the FFT + k-means experimental setting is that input functions sampled under different physical regimes (e.g., varying temperature or energy scales) often exhibit different spectral signatures in Fourier space. For example, in weather forecasting, atmospheric states in summer and winter can exhibit substantially different spectral characteristics, reflecting seasonal differences in temperature and wind field structure. Indeed, the analysis in [49] shows that wintertime atmospheric states contain a higher proportion of low-frequency components, whereas summertime atmospheric states contain a higher proportion of high-frequency components. Moreover, since the spectral composition of the target distribution is typically unknown at the time training data is generated via simulation, some degree of mismatch between the training and test spectral distributions is a natural consequence of this lack of prior knowledge, rather than an artifact specific to our setting. Accordingly, we construct training and test sets from the two clusters obtained via FFT-based clustering, using one for training and the other for testing, in order to emulate this form of distribution shift in a controlled manner.

To examine whether FFT + k-means constructs this type of measure shift arising from differences in spectral composition, we further performed a spectral analysis of the training and test data for the Darcy flow equation. Specifically, we applied a Fourier transform to each two-dimensional input field and then averaged the resulting spectrum over radial frequencies. The normalized radial frequency is defined as $\tilde{k} = \frac{\sqrt{k_x^2 + k_y^2}}{k_{\max}} \in [0, 1]$, where k_x and k_y denote the frequencies along the two spatial dimensions, respectively, and $k_{\max}$ denotes the maximum radial frequency considered. Smaller values of $\tilde{k}$ correspond to low-frequency components, whereas larger values correspond to high-frequency components.

As shown in Table 8, the training data are more concentrated in the low-frequency region, whereas the test data contain a higher proportion of high-frequency components, indicating a clear shift in spectral composition. These results confirm that FFT + k-means constructs a probability measure shift induced by differences in the spectral statistics of the input functions. It therefore provides a controlled and physically motivated abstraction of practical scenarios in which simulation data and downstream inputs exhibit different frequency distributions.

Table 8: Spectral analysis of the training and test data on the Darcy flow equation. We report the proportions of lower- and higher-frequency components in the training and test datasets.

Frequency group	Training data (%)	Test data (%)
Lower-frequency components ($\tilde{k} \leq 0.147$)	79.5	65.2
Higher-frequency components ($\tilde{k} > 0.147$)	18.6	33.9

Table 9: Effect of overlap between the training and test measures on the Darcy flow equation. We report the average test RMSE and standard deviation over five runs. All values are multiplied by 10^3.

Method	0%	1%	10%	30%
FNO	38.9 ± 0.4	36.4 ± 0.6	30.5 ± 0.9	24.2 ± 0.7
FNO-Ours	39.5 ± 0.8	31.9 ± 0.1	20.1 ± 0.4	15.3 ± 0.2

E.2 Effect of overlap between training and test measures

Although defining relative importance weights with respect to a mixture measure guarantees their existence and boundedness, the weighted training objective is still evaluated under the training measure. Therefore, the proposed method still relies on overlap between the training and test measures for effective adaptation.

In the clustering-based experiments described in Section 5.1, the training set includes a randomly selected 10% subset of cluster B to ensure overlap between the training and test measures. To investigate the effect of overlap, we varied this proportion from 0% to 1%, 10%, and 30% and evaluated the methods on the Darcy flow equation over five runs.

As shown in Table 9, with 0% overlap, FNO-Ours does not improve upon the unweighted FNO, and the difference is not statistically significant according to a paired t-test at the 5% significance level. However, with only 1% overlap, FNO-Ours already significantly outperforms FNO, and the improvement becomes more pronounced as the overlap increases to 10% and 30%.

These results clarify that defining relative importance weights with respect to a mixture measure resolves their existence and boundedness issues but does not eliminate the need for overlap between the training and test measures. Nevertheless, the proposed method achieves a significant improvement with only 1% overlap, demonstrating its effectiveness even when the overlap is very limited.

To further examine the performance of the proposed method on test samples outside the training support, we conducted an additional experiment based on Setting 1 in Table 6. The distribution of the input-generation parameters for the training data was kept unchanged, with $k_j \in [0.1, 1.0]$, while the unlabeled test inputs followed the same parameter distribution as in Setting 1, with $k_j \in [0.9, 1.9]$. Thus, the training and unlabeled test distributions overlap only over $[0.9, 1.0]$ along the k_j dimension.

To evaluate performance at different locations outside the training support, we kept the other input-generation parameters unchanged and evaluated the models on test samples with $k_j = 1.2$, 1.5, and 1.9, respectively. At the level of the input-generation parameters, all three test settings lie outside the training support, $k_j \in [0.1, 1.0]$. Moreover, as k_j increases, the test samples move progressively farther from the boundary of the training support.

As shown in Table 10, DeepONet-Ours achieves lower RMSE than the unweighted DeepONet across all three settings. The improvement is most pronounced when the test samples are relatively close to the boundary of the training support and gradually diminishes as they move farther away.

These results indicate that test samples lying outside the training support do not necessarily render importance weighting ineffective. When the test inputs remain close to the boundary of the training support, the training data may still contain substantial information relevant to these inputs. The proposed method can therefore improve predictive performance by assigning greater importance to more relevant training samples. However, as the test inputs move farther from the training support, the amount of relevant information available in the training data decreases, and the benefit of importance weighting correspondingly diminishes.

Table 10: Performance on the KdV equation with test inputs generated outside the training parameter support. Here, k_j denotes the frequency parameter of the j-th sine component used to generate the initial conditions. The training data use $k_j \in [0.1, 1.0]$, while test inputs are generated with $k_j = 1.2$, 1.5, and 1.9. We report the average test RMSE and standard deviation over five runs. All values are multiplied by 10^3.

Method	$k_j = 1.2$	$k_j = 1.5$	$k_j = 1.9$
DeepONet	29.0 ± 0.9	53.8 ± 0.6	75.1 ± 1.4
DeepONet-Ours	14.0 ± 0.2	43.9 ± 0.6	71.9 ± 1.7

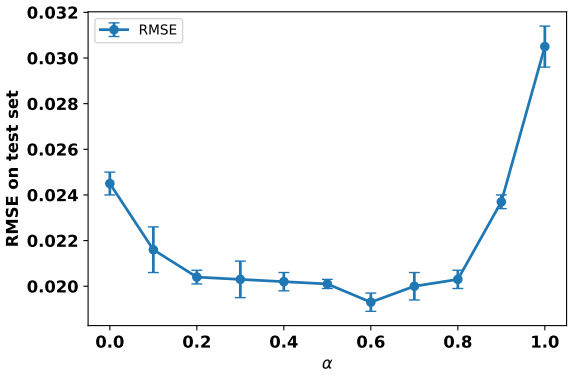


Figure 3: Average test RMSE with standard errors of the proposed method on the Darcy flow equation when varying the mixing parameter α.

E.3 Effect of the mixing parameter α

Figure 3 shows the average test RMSEs, together with standard errors, of the proposed method for the Darcy flow equation as the mixing parameter α in the relative importance weight defined in Eq. (6) is varied. The figure reports the RMSE for α ranging from 0 to 1. When $\alpha = 0$, the relative importance weight in Eq. (6) reduces to the standard importance weight in Eq. (3), and the model parameters are trained by minimizing the importance-weighted training error in Eq. (4). When $\alpha = 1$, all samples are assigned equal weights, and the model parameters are trained by minimizing the ordinary training error in Eq. (1). For $\alpha \in (0, 1)$, the proposed method estimates the relative importance weights in Eq. (6), and the model parameters are trained by minimizing the relative importance-weighted error in Eq. (9).

The results show that both standard and relative importance weighting outperform the unweighted baseline. This demonstrates that, under probability measure shifts, incorporating unlabeled data (input functions) from the test measure enables the model to emphasize training samples that are more consistent with the test distribution, thereby improving the predictive performance. Moreover, relative importance weighting achieves better performance than standard importance weighting. This suggests that relative weights mitigate the effect of unbounded or extreme weights and reduce the variance of the finite-sample estimator of the expected test error, resulting in further performance gains. This finding is consistent with observations in the finite-dimensional setting [28].

To further investigate the relationship between α and the degree of overlap between the training and test measures, we evaluated $\alpha \in \{0.2, 0.5, 0.8\}$ under different overlap conditions on the Darcy flow equation. Specifically, the overlap ratio refers to the proportion of samples from cluster B randomly included in the training set, as described in Section 5.1. As shown in Table 11, $\alpha = 0.5$ achieves stable and competitive performance across all settings, whereas $\alpha = 0.8$ performs better when the overlap ratio is only 1%. This is consistent with the tighter upper bound on relative importance weights induced by a larger α, which can further improve training stability when the overlap is extremely limited. Although these results suggest that the optimal choice of α may depend on the degree of overlap, the actual overlap between the training and test measures is generally unknown in practical applications. Therefore, adaptively selecting α based on the overlap level is challenging.

Table 11: Sensitivity to the mixing parameter α under different overlap conditions on the Darcy flow equation. We report the average test RMSE and standard deviation over five runs. All values are multiplied by 10^3.

Overlap	$\alpha = 0.2$	$\alpha = 0.5$	$\alpha = 0.8$
1%	31.7 ± 0.2	31.9 ± 0.1	30.6 ± 0.4
10%	20.7 ± 1.0	20.1 ± 0.4	20.4 ± 0.7
30%	15.8 ± 0.3	15.3 ± 0.2	16.0 ± 0.6

Table 12: Distribution statistics of importance weights over $I = 2495$ training samples for the Darcy flow equation. KDE and FNN denote two baseline weight estimation methods, while Ours (IW) and Ours (RIW) denote our methods based on standard importance weighting and relative importance weighting, respectively. We report the maximum, minimum, mean, variance, and effective sample size (ESS) of the resulting weights for each method.

Method	I	max	min	mean	var	ESS
KDE	2495	2.5×10^{-11}	0.0	5.5×10^{-13}	1.8×10^{-23}	41.4
FNN	2495	5.3×10^{-4}	0.0	2.1×10^{-6}	3.1×10^{-10}	34.8
Ours (IW)	2495	31.1	3.6×10^{-7}	9.9×10^{-1}	9.1	245.2
Ours (RIW)	2495	1.9	6.6×10^{-7}	2.6×10^{-1}	3.2×10^{-1}	437.0

While $\alpha = 0.5$ is not always optimal, it provides a reasonable default when no principled criterion for selecting α is available.

E.4 Weight distribution statistics

Table 12 reports the distribution statistics of the estimated importance weights on the Darcy flow equation. We also report the effective sample size (ESS), a standard diagnostic that quantifies how many training samples are effectively used under the estimated importance weights [50]. It is computed as

$$\mathrm{ESS} = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{\sum_{i=1}^{n} w_i^2},$$

where w_i denotes the estimated importance weight of the i-th training sample. The results show that the KDE and FNN baselines assign most of the total weight to a small subset of training samples, leaving most samples with nearly zero relative weights and leading to very small ESS values. This indicates that these finite-dimensional weight estimation methods effectively use only a limited fraction of the training data during weighted training. In comparison, Ours (IW) produces more informative importance weights and achieves a substantially larger ESS, suggesting that the proposed weight estimation method is more suitable for operator learning under probability measure shifts. However, since Ours (IW) uses importance weighting, it can still generate large weights, thereby increasing the variance of the weighted empirical objective. Ours (RIW) mitigates this issue by producing bounded weights with lower variance, as reflected by its smaller maximum weight and variance, as well as the largest ESS. These observations are consistent with the theoretical motivation of relative importance weighting: it is designed to reduce the instability caused by extreme weights while still placing greater emphasis on training samples that are more relevant to the test measure.

E.5 Results with different numbers of unlabeled samples in the test measure

Figure 4 shows the average test RMSE with standard errors as the number of unlabeled samples in the test measure varies. As the number of unlabeled samples increased, the proposed method achieved lower RMSE, indicating improved performance. Overall, the proposed method consistently outperformed the baselines across different numbers of unlabeled samples in the test measure.

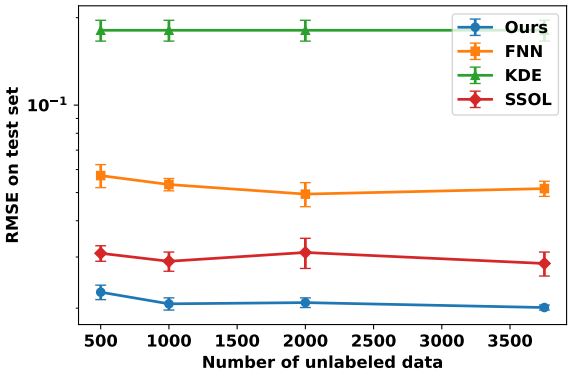


Figure 4: Average RMSE with standard errors of the proposed method and baselines on the Darcy flow equation when varying the number of unlabeled test samples.

Table 13: Effect of classifier architecture on the Burgers', Darcy flow, and Navier–Stokes (NS) equations. GNO-type and FNO-type denote the classifier architectures used for importance weight estimation, while the downstream neural operator is FNO in all cases. We report the average test RMSE and standard deviation over five runs. All values are multiplied by 10^3.

Method	Burgers'	Darcy flow	NS($\nu = 10^{-3}$)
FNO	5.4 ± 1.2	30.5 ± 0.9	61.7 ± 3.1
FNO-Ours (GNO-type)	2.3 ± 1.6	20.6 ± 0.7	15.0 ± 2.3
FNO-Ours (FNO-type)	1.6 ± 1.1	20.1 ± 0.4	15.8 ± 4.4

E.6 Effect of classifier architecture

To investigate whether the performance of the proposed weight estimator depends on architecture-specific bias, we replaced the original FNO-type classifier with a GNO-type classifier. For a controlled comparison, only the classifier architecture was changed, while the data split, weight construction, and downstream FNO training protocol were kept unchanged.

The GNO-type classifier consisted of two kernel integral layers with hidden dimension 10 and kernel rank 4. For each input field, we sampled 256 nodes and constructed a radius graph with a normalized distance threshold of 0.15. The dynamic kernel generated edge weights from relative node positions, distances, and input field values at both endpoints. Node features were updated through mean aggregation of neighboring messages, residual connections, LayerNorm, and feedforward layers. Global mean pooling followed by a binary classification head was used for importance weight estimation. The graph construction was adapted to each PDE: a one-dimensional periodic graph for Burgers', a two-dimensional non-periodic graph for Darcy flow, and a two-dimensional periodic graph for Navier–Stokes.

As shown in Table 13, both classifier architectures substantially outperform the unweighted FNO baseline across all three PDEs. The GNO-type classifier achieves RMSEs of 2.3, 20.6, and 15.0 on the Burgers', Darcy flow, and Navier–Stokes equations, respectively, comparable to those obtained using the FNO-type classifier (1.6, 20.1, and 15.8). Moreover, the Pearson correlations between the importance weights estimated by the two classifiers are 0.88, 0.93, and 0.97 for the three PDEs, respectively, indicating strong consistency in their relative importance assignments.

These results demonstrate that structurally different classifier architectures yield highly correlated importance weights and comparable downstream predictive performance, supporting the robustness of the proposed framework to classifier architecture.

E.7 Computational cost

Table 14 reports the training time of the proposed method and the baselines on the Darcy flow equation. All training times were measured on an NVIDIA TITAN RTX GPU. The training time of

Table 14: Training time in seconds for the Darcy flow equation.

FNO	FNO-KDE	FNO-FNN	FNO-Ours
4423.7	4433.0	4498.7	4526.3

Table 15: Comparison of standard importance weighting (IW) and relative importance weighting (RIW) under different overlap conditions on the Darcy flow equation. We report the average test RMSE and standard deviation over five runs. All values are multiplied by 10^3.

Method	1%	10%
FNO	36.4 ± 0.6	30.5 ± 0.9
FNO-Ours (IW)	38.7 ± 1.2	24.5 ± 0.5
FNO-Ours (RIW)	31.9 ± 0.1	20.1 ± 0.4

our method is comparable to, or slightly longer than, that of the other methods. This result indicates that the proposed method is practical in terms of the computational cost.

E.8 Comparison of standard and relative importance weighting

To investigate when standard importance weighting (IW) may degrade predictive performance and whether relative importance weighting (RIW) mitigates this issue, we conducted additional experiments on the Darcy flow equation. Specifically, we compared standard importance weighting, FNO-Ours (IW), with relative importance weighting, FNO-Ours (RIW), under training–test overlap ratios of 1% and 10%.

As shown in Table 15, at 1% overlap, FNO-Ours (IW) yields a higher RMSE than the unweighted FNO, suggesting that standard importance weighting may assign excessively large weights to a small number of training samples and destabilize finite-sample training. In contrast, FNO-Ours (RIW) reduces the RMSE to 31.9, outperforming both baselines.

At 10% overlap, both weighting methods outperform FNO, while FNO-Ours (RIW) further reduces the RMSE from 24.5 for FNO-Ours (IW) to 20.1. Thus, RIW consistently achieves lower test error than standard IW, with a more pronounced advantage under limited overlap.

To further investigate the practical differences between standard IW and RIW, we extended the experiments in Section D to probability measure shifts satisfying the absolute-continuity condition. By allowing the training and test distributions of the parameters b_j and k_j to overlap, we construct probability measures on the function space that satisfy the absolute-continuity condition while exhibiting a probability measure shift.

For the training measure, we retain $b_j \sim \mathrm{Unif}(-0.5, 0.5)$ and $k_j \sim \mathrm{Unif}(0.1, 1.0)$. For the test measure, we keep the distribution of b_j unchanged and set $k_j \sim \mathrm{Unif}(c, 1.0)$, where $c \in \{0.5, 0.7, 0.9\}$. Since each test interval $[c, 1.0]$ is contained within the training interval $[0.1, 1.0]$, the test parameter measure is absolutely continuous with respect to the training parameter measure. Consequently, the induced test measure on the function space is also absolutely continuous with respect to the training measure.

Under this extension of the Section D setting, we compared IW and RIW on the KdV equation across the three test distributions. As shown in Table 16, RIW achieves lower RMSE than standard IW across all settings. In particular, when $k_j \in [0.9, 1.0]$, IW performs worse than the unweighted FNO, whereas RIW achieves the lowest RMSE.

When the test measure is concentrated within a narrow region of the training support, only a small fraction of the training samples is highly relevant to the test measure. Standard IW may therefore assign excessively large weights to these samples, reducing the stability of finite-sample training. In contrast, RIW uses a mixture reference measure and bounds the population relative importance weights by $1/\alpha$, thereby suppressing extreme weights and improving predictive performance.

Table 16: Comparison of standard importance weighting (IW) and relative importance weighting (RIW) on the KdV equation under probability measure shifts satisfying absolute continuity. We report the average test RMSE and standard deviation. All values are multiplied by 10^3.

Method	$k_j \in [0.5, 1.0]$	$k_j \in [0.7, 1.0]$	$k_j \in [0.9, 1.0]$
FNO	10.7 ± 0.6	12.6 ± 1.1	15.2 ± 1.3
FNO-Ours (IW)	8.7 ± 0.5	9.9 ± 0.9	21.1 ± 1.8
FNO-Ours (RIW)	8.1 ± 0.3	8.5 ± 0.4	13.4 ± 0.8

Table 17: Comparison with the test-time adaptation baseline SSA on the Burgers', Darcy flow, and Navier–Stokes (NS) equations. We report the average test RMSE and standard deviation over five runs. All values are multiplied by 10^3.

Method	Burgers'	Darcy flow	NS ($\nu = 10^{-3}$)
FNO	5.4 ± 1.2	30.5 ± 0.9	61.7 ± 3.1
FNO-SSA	22.7 ± 1.8	53.7 ± 1.1	75.4 ± 5.9
FNO-Ours	1.6 ± 1.1	20.1 ± 0.4	15.8 ± 4.4

E.9 Comparison with test-time adaptation

To further evaluate the proposed method, we consider a test-time adaptation (TTA) baseline. To the best of our knowledge, there are currently no TTA methods specifically designed for operator learning. We therefore adopt subspace alignment for regression (SSA) [51], a recently proposed TTA method for regression tasks, as a comparison baseline.

As shown in Table 17, FNO-SSA achieves RMSEs of 22.7, 53.7, and 75.4 on the Burgers', Darcy flow, and Navier–Stokes equations, respectively, performing worse than the unweighted FNO across all three PDEs. In contrast, FNO-Ours achieves substantially lower RMSEs of 1.6, 20.1, and 15.8, respectively. These results demonstrate the effectiveness of the proposed method compared with SSA under probability measure shifts. The limited performance of SSA may be attributed to its lack of specific design for operator learning.

F Broader impact statement

The proposed method computes weights for each training sample using unlabeled datasets drawn from the test measure. Consequently, there is a potential risk that users might use biased datasets to calculate these weights without careful consideration, potentially leading to biased predictions. We encourage research into developing methods for automatically detecting biased datasets.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The Abstract and Introduction outline the scope and contributions of this work. A summary of the main contributions is provided at the last of the Introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations and future work in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The theoretical components state their assumptions explicitly, including absolute continuity for the Radon–Nikodym derivative and the use of a mixture measure for relative importance weights.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides substantial information needed to reproduce the main experiments, including the training algorithm in Appendix A, dataset construction and split rules in Section 5.1, model and optimizer details in Section 5.2, baseline implementations in Section 5.3, and additional dataset details in Appendix D.2.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: While the code has not yet been released as it is still undergoing organization and documentation, we have provided detailed descriptions of the experimental setup in Section 5.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the main experimental settings, including PDE datasets, probability measure shift construction, training/test/unlabeled splits, FNO and DeepONet configurations, optimizer, learning rate schedule, number of epochs, classifier architecture, and baseline implementations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports mean RMSE and standard deviation over five runs for the main FNO and DeepONet results, and reports standard errors in additional analyses such as the mixing-parameter and unlabeled-data studies.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E.7 reports computation cost and states that training times were measured on an NVIDIA TITAN RTX GPU.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses simulated PDE data and publicly available scientific benchmark datasets, and does not involve human subjects, private data, surveillance, or high-risk deployment.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: A discussion of the broader impact of this work is presented in Appendix F.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The original papers of the datasets and models used in this work are cited in the paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.