Physica D 471 (2025) 134418

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Conformalized-DeepONet: A distribution-free framework for uncertainty quantification in deep operator networks

Christian Moya [c], Amirhossein Mollaali [d], Zecheng Zhang [a,*], Lu Lu [b], Guang Lin [c]

[a] Department of Mathematics, Florida State University, Tallahassee, FL 32304, USA
[b] Department of Statistics and Data Science, Yale University, New Haven, CT 06511, USA
[c] Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA
[d] School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA

ARTICLE INFO

Communicated by ChangQing Cheng

Keywords:
Operator learning
Uncertainty quantification
Conformal prediction
Quantile prediction
Dynamical systems
Time-dependent PDE

ABSTRACT

In this paper, we adopt conformal prediction, a distribution-free uncertainty quantification (UQ) framework, to obtain prediction intervals with coverage guarantees for Deep Operator Network (DeepONet) regression. Initially, we enhance the uncertainty quantification frameworks (B-DeepONet and Prob-DeepONet) previously proposed by the authors by using split conformal prediction. By combining conformal prediction with our Prob- and B-DeepONets, we effectively quantify uncertainty by generating rigorous prediction intervals for DeepONet prediction. Additionally, we design a novel Quantile-DeepONet that allows for a more natural use of split conformal prediction. We refer to this distribution-free effective uncertainty quantification framework as split conformal Quantile-DeepONet regression. Finally, we demonstrate the effectiveness of the proposed methods using various ordinary, partial differential equation numerical examples, and multi-fidelity learning.

1. Introduction

Recent advancements in neural network technology have solidified its standing as a reliable and efficient tool for function approximation [1,2]. This is particularly evident in scientific computing, where neural networks have demonstrated their power in approximating solutions to Partial Differential Equations (PDEs) or Ordinary Differential Equations (ODEs) [3–7]. A prevalent challenge in scientific computing involves approximating operators that map one function to another. Seminal research endeavors, such as those by Chen and Chen [8,9], have explored the utilization of neural networks to approximate such operators, commonly referred to as neural operators.

In recent years, there has been a surge of interest in neural operators, with notable contributions from various scholars [10–17]. Among these, two prominent approaches have emerged as particularly successful: Deep Operator Neural Networks (DeepONet) [10–12,18–20] and Fourier Neural Operator (FNO) [14–17,21–23].

Compared to FNO, DeepONet represents a mesh-free neural operator, implying that the output functions do not require discretization. This characteristic enhances the flexibility of DeepONet in both training and testing phases [11,22,24–29]. Furthermore, recent advancements in BelNet by [22,25,26] have extended DeepONet's capabilities, making it invariant to input function discretization. This means that input functions no longer need to conform to a shared discretization, thereby enhancing the versatility of DeepONet even further.

DeepONet and its extensions have been successfully applied to a diverse array of real-world applications. These include weather forecasting [23], sub-surface structure detection [13], electrical propagation on the left ventricle [30], geological carbon sequestration [31], disk-planet interactions in protoplanetary disks [32], hypersonic systems [33,34], power systems [35], and optimization [36].

The need for Uncertainty Quantification (UQ) in scientific machine learning [37,38] often arises from several factors. These include uncertainties in data due to measurement errors and numerical algorithm errors [21], uncertainties in model forms due to network architectures and physical models of varying fidelities [27,37,39], and uncertainties in parameters due to network training and system properties at different scales [37,39]. As a result, researchers are tasked with providing prediction intervals for predicted outputs. A good interval should meet two fundamental criteria. Firstly, it should have a significant coverage rate, encompassing precise solutions or dynamics to the greatest extent possible. Secondly, the interval should be minimized in size.

* Corresponding author.
E-mail addresses: cmoyacal@purdue.edu (C. Moya), amollaal@purdue.edu (A. Mollaali), zecheng.zhang.math@gmail.com (Z. Zhang), lu.lu@yale.edu (L. Lu), guanglin@purdue.edu (G. Lin).

https://doi.org/10.1016/j.physd.2024.134418
Received 27 April 2024; Received in revised form 29 September 2024; Accepted 22 October 2024
Available online 22 November 2024

Numerous works [21,27,36–43] have established various uncertainty quantification (UQ) frameworks for DeepONet. These frameworks aim not only for accurate prediction but also for quantifying uncertainty. For example, a recent study [21] from the authors addresses this uncertainty issue in operator learning by framing the training process within a Bayesian framework. The study uses Langevin diffusion-based sampling methods [44–48] to generate ensembles, which aids in characterizing uncertainty in DeepONet predictions. To lessen the use of ensembles and increase efficiency, we introduced a probabilistic framework for UQ in DeepONets in [27], which provides an input-dependent standard deviation as a heuristic measure of uncertainty. However, despite the promising results from these methods, constructing rigorous prediction intervals for DeepONet predictions remains an unsolved challenge, which we aim to solve using conformal prediction.

Conformal prediction [49–51] is an alternative method for constructing prediction intervals that provide nonasymptotic, distribution-free coverage guarantees. Unlike the Bayesian Framework, this approach does not require prior distribution knowledge and avoids relying on strong assumptions of large-sample asymptotic approximations, which may be difficult to justify in practical scenarios. These characteristics make conformal prediction a valuable tool for addressing scientific machine learning problems, especially those arising from engineering applications with real observed data and a lack of prior knowledge.

There are two types of conformal prediction methods: full conformal prediction [51,52] and split conformal prediction [51,53]. In this paper, we focus on split conformal prediction, which operates with finite samples and enables building adaptive and validated prediction intervals. These properties have allowed split conformal prediction to become a popular approach for machine learning and uncertainty quantification tasks, including Language Modeling [54], Graph Neural Networks [55], time series [56], and quantile regression [57]. However, despite its potential, split conformal prediction remains underutilized in the scientific machine learning community, particularly in the context of Deep Operator Networks.

The *goal* of this paper is to address the challenge of uncertainty quantification (UQ) in operator learning. It aims to showcase the implementation of split conformal prediction for constructing prediction intervals and improving existing UQ algorithms tailored for DeepONet regression, with coverage guarantees.

To achieve this goal, we make the following contributions:

1. We design a split conformal prediction algorithm (Section 3.1) that enhances the previously proposed uncertainty quantification frameworks, Bayesian DeepONet (B-DeepONet) and Probabilistic DeepONet (Prob-DeepONet). This novel algorithm allows B-DeepONet and Prob-DeepONet to generate prediction intervals with coverage guarantees.
2. We then propose a new extension of DeepONet called Quantile-DeepONet (Section 3.2), which we use in a novel Conformal Quantile-DeepONet regression algorithm to construct rigorous prediction intervals.
3. Finally, we demonstrate the effectiveness of our proposed conformalized-DeepONets by conducting multiple numerical experiments on ordinary differential equations, partial differential equations, and multi-fidelity settings.

The remaining sections of this paper are organized as follows. Section 2 provides a review of the concepts of DeepONet, B-DeepONet, and Prob-DeepONet. In Section 3.1, we describe how to use split conformal prediction on B-DeepONet and Prob-DeepONet to build prediction intervals with coverage guarantees. Section 3.2 presents the design of the Quantile DeepONet, which serves as the main component of the conformal Quantile-DeepONet regression algorithm for constructing rigorous prediction intervals. We present numerical experiments that validate the performance of all conformalized DeepONets in Section 4. Finally, Section 5 discusses our results and future work, and Section 6 concludes the paper.

2. Background information

This section reviews the Deep Operator Network (DeepONet) framework [10]. It also provides a summary of two uncertainty quantification (UQ) frameworks for DeepONet: Bayesian DeepONet (B-DeepONet [21]) and probabilistic DeepONet (Prob-DeepONet [27]).

2.1. Deep operator network (DeepONet)

The Deep Operator Network (DeepONet) [10–12,30] is an operator learning framework that aims to approximate the nonlinear operator G. The operator G maps a function space U with domain K_1 to a function space V with domain K_2. DeepONet has been theoretically supported by universal approximation theorems [8,18,26,58]. These theorems enable the approximation of the operator G for a given $u \in U$ and $x \in K_2$ using the following linear trainable representation:

$$G(u)(x) \approx G_\theta(\hat{u})(x) = \sum_{k=1}^{K} b_k(\hat{u})\tau_k(x),$$

Here, θ is the vector of trainable parameters, while b_k and τ_k are K trainable coefficients and basis functions, respectively.

DeepONet consists of two sub-networks: the branch network and the trunk network (see Fig. 1). The *branch* network maps the input function $u \in U$, which is discretized using m sensors and denoted as $\hat{u}$, to a vector of K trainable coefficients $b(\hat{u}) \in \mathbb{R}^K$. It is worth noting that various variants of DeepONet have been proposed to handle input functions with different discretizations. For instance, the new enhanced structures proposed in [25,26] and the training algorithms proposed in [22].

On the other hand, the *trunk* network maps a location $x \in K_2$, which belongs to the output function domain $K_2 \subset \mathbb{R}^d$, to a vector of K trainable basis functions. This feature eliminates the need for discretization of the output function $G(u) \in V$ in DeepONet and its variants, making them mesh-free PDE approximate solvers. To train DeepONet, one could optimize,

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N_j}\left|G^{(i)} - G_\theta\left(\hat{u}^{(i)}\right)\left(x^{(i)_j}\right)\right|^2.$$

Without loss of generality and for better illustration of the algorithms, we assume $N_j = 1$, or $x^{(i)} \in K_2$ is the only evaluation point for one function $u^{(i)}$. That is, we have the dataset, using the dataset of N triplets: $\mathcal{D} = \left\{\hat{u}^{(i)}, x^{(i)}, G^{(i)}\right\}_{i=1}^{N}$, where $G^{(i)} = G\left(\hat{u}^{(i)}\right)\left(x^{(i)}\right)$ is the target operator value.

Despite the remarkable accuracy of DeepONet on multiple applications (e.g., [13,23,27,30]), it only produces pointwise predictions. However, in cases where small training datasets or noisy inputs are involved, these pointwise predictions may be unreliable. Therefore, it is necessary to have some measure of the DeepONet's uncertainty. This requirement led the authors of this paper to develop two uncertainty quantification frameworks for DeepONet: Bayesian and probabilistic DeepONets, which we review next.

2.2. Bayesian deep operator network (B-DeepONet)

This section reviews the first framework for uncertainty quantification (UQ) in DeepONet, known as Bayesian DeepONet (B-DeepONet) [21], developed by the authors of this paper. In particular, B-DeepONet is based on the Langevin theory that allows us to generate samples from an approximate posterior that converges to the Gibbs distribution proportional to $\exp(-U(\theta)/\tau)$, where U is the non-convex energy function and $\tau > 0$ the temperature.

Measuring the uncertainty linked with constrained training datasets, noisy inputs, and over-parametrization of neural networks poses a considerable challenge. This challenge is even more pronounced in deep operator regression, as it involves mapping between functional spaces. To address this challenge, we proposed the Bayesian DeepONet

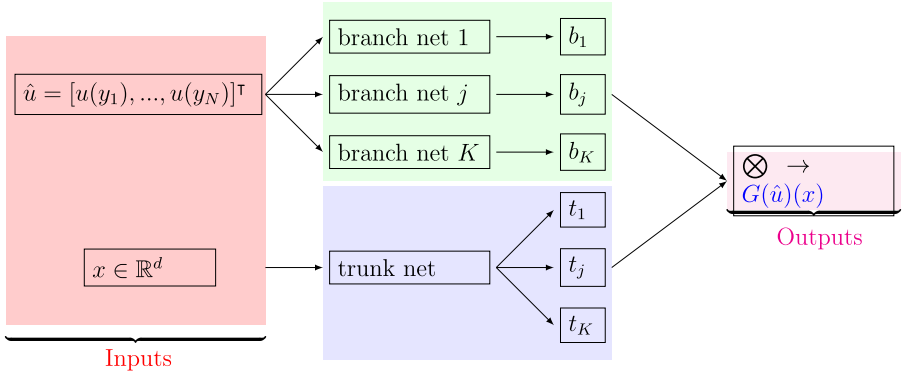


Fig. 1. Stacked version DeepONet G_θ. $\otimes$ denotes the inner product in $\mathbb{R}^K$. Specifically, $t_j \in \mathbb{R}$ is the trunk net output, and $b_j \in \mathbb{R}$ is the brunch net output..

(B-DeepONet) in [21]. B-DeepONet allows us to construct heuristic prediction intervals for the operator that maps an input function $u \in V$ to an output function $G(u)(x) \in \mathbb{R}$ evaluated at a point $x \in K_2 \subset \mathbb{R}^d$.

In B-DeepONet, our goal was to develop a method to sample from the predictive distribution $p(G|(u,x),\mathcal{D})$ of the target operator $G(u)(x)$, given a discretized input $\hat{u} \in \mathbb{R}^m$ at any $x \in K_2$ using the training dataset $\mathcal{D}$. To reach this goal, we first assumed a factorized Gaussian likelihood function for the dataset [44,59]:

$$p(G|(\hat{u},x),\theta) = \mathcal{N}(G|G_\theta(\hat{u})(x), \text{diag}(\Sigma^2)) = \prod_{i=1}^{N} \mathcal{N}(G^{(i)}|G_\theta(\hat{u}^{(i)})(x^{(i)}),\sigma),$$

where the mean $G_\theta(\hat{u})(x)$ is the output of the underlying DeepONet with a vector of trainable parameters θ and $\text{diag}(\Sigma^2)$ a diagonal covariance matrix with $\Sigma^2 = (\sigma^2, \dots, \sigma^2)$ on the diagonal [37], which we assume is given.

It is important to note that the target operator value $G(u)(x) \in \mathbb{R}$ for a given input function $u \in U$ at any location $x \in K_2$ and the training dataset $\mathcal{D}$ is the random variable $(G|(u,x),\mathcal{D})$. To calculate the density of this random variable, the DeepONet parameters θ need to be integrated out, as shown below:

$$p(G|(u,x),\mathcal{D}) = \int p(G|(u,x),\theta)p(\theta|\mathcal{D})d\theta.$$

In the above, $p(\theta|\mathcal{D})$ is the posterior distribution of the DeepONet trainable parameters. Using Bayes' rule, we can write the posterior of DeepONet parameters as

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta),$$

where $p(\theta)$ is the *prior* distribution of the parameters, and $p(\mathcal{D}|\theta)$ is the *data likelihood*, that is, $p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(G^{(i)}|(\hat{u}^{(i)},x^{(i)}),\theta)$. We calculate this using the DeepONet forward pass and the i.i.d. training dataset $\mathcal{D}$.

Computing the posterior distribution in closed form using Bayes' rule is intractable [37] for deep operator networks. Thus, to generate samples from this distribution, we employ a Langevin diffusion-based sampling technique [47,60]. This technique has been shown to produce samples that closely match the target distribution [61,62]. Specifically, the Langevin diffusion process used to generate samples is as follows:

$$d\theta_t = -\nabla U(\theta_t)dt + \sqrt{2\tau}dW_t,$$

where θ is the vector of parameters, U is the energy function (i.e., the log data likelihood and prior), τ the temperature, and W_t is the Brownian motion. By sampling from the posterior distribution using a discretized Langevin process, we obtained an ensemble of M θ samples, denoted as $\{\theta_k\}_{k=1}^M$, as described next.

To obtain the ensemble of parameters, denoted as $\{\theta_k\}_{k=1}^M$, B-DeepONet uses a variation of the diffusion process mentioned earlier. Particularly, the authors consider the parallel tempering techniques. These methods involve two chains and facilitate particle swapping between them, aiding in escaping local traps and addressing highly non-convex problems, thereby enhancing its capability to tackle complex UQ challenges. However, employing two chains doubles the cost of energy function evaluation. Despite efforts by the authors to mitigate these costs, achieving only up to a 25% reduction in the doubled cost, this prompts the exploration of alternative UQ methods for managing uncertainty in neural operators.

To construct the prediction intervals, in our paper [21], we used the obtained M-ensemble $\{\theta_k\}_{k=1}^M$ of sampled parameters from the posterior distribution to fit a parametric predictive distribution, a common practice in the literature [37]. It is important to note that this assumption limits the applicability of these prediction intervals. Specifically, we used the Gaussian distribution $\mathcal{N}(\mu(\hat{u})(x), \sigma(\hat{u})(x))$ for an arbitrary x location. The parameters of this distribution were obtained as follows:

$$\mu(\hat{u})(x) = \frac{1}{M}\sum_{k=1}^{M} G_{\theta_k}(\hat{u})(x)$$

$$\sigma^2(\hat{u})(x) = \frac{1}{M}\sum_{k=1}^{M}\left(G_{\theta_k}(\hat{u})(x) - \mu(\hat{u})(x)\right)^2.$$

Due to the assumption of fitting the parametric predictive distribution, B-DeepONet can only provide a heuristic estimate of uncertainty. In our paper [21], we demonstrated that constructing prediction intervals with B-DeepONet requires extensive hyperparameter optimization to achieve a high prediction level (e.g., 95%) for a specific test dataset. However, there is no guarantee that these heuristic prediction intervals, developed using B-DeepONet, will be reliable for other test examples. Additionally, constructing these heuristic prediction intervals requires the forward pass of M DeepONets, making B-DeepONet a computationally expensive method.

2.3. Probabilistic deep operator network (Prob-DeepONet)

One of the main drawbacks of quantifying uncertainty using B-DeepONet is its cost. B-DeepONet requires performing a forward pass across all the DeepONets within the collected ensemble of size M. To address this drawback, the authors of this paper developed the Probabilistic DeepONet (Prob-DeepONet) [27], which we will review in this section.

Prob-DeepONet is a probabilistic model whose output approximates the parameters of a normal distribution:

$$G|X = (\hat{u},x) \sim \mathcal{N}\left(\mu_{\theta_\mu}(\hat{u})(x), \sigma_{\theta_\sigma}(\hat{u})(x)\right).$$

In the above, μ_{θ_μ} represents the mean of the probabilistic model and aims to approximate $G(u)(x) \in \mathbb{R}$, which is the target operator for a given input $u \in U$ at $x \in K_2$. This approximation is achieved using the following linear trainable representation:

$$\mu_{\theta_\mu}(\hat{u})(x) = \sum_{i=1}^{K} b_k^\mu(\hat{u}) \cdot \tau_k^\mu(x).$$

Similarly, σ_{θ_σ} represents the standard deviation of the probabilistic model. It aims to estimate the uncertainty for $G(u)(x)$ through the following linear trainable representation:

$$\log(\sigma_{\theta_\sigma}(\hat{u})(x)) = \sum_{i=1}^{K} b_k^\sigma(\hat{u}) \cdot \tau_k^\sigma(x).$$

Note that in the above, for numerical stability, we train the model to produce $\log\sigma$ and then recover the standard deviation σ using the exponential function.

Similar to the classical DeepONet, Prob-DeepONet consists of two sub-networks: the branch network and the trunk network. In the branch network, we design two components: one for the mean and the other for the standard deviation. The branch component for the mean includes shared trainable layers and a few independent layers. It maps the discretized input $\hat{u} \in \mathbb{R}^m$ to a vector of K trainable coefficients $b^\mu(\hat{u}) \in \mathbb{R}^K$ for the mean. Similarly, the branch component for the log standard deviation shares several layers with the mean component. It also has independent layers to process the map from the discretized input $\hat{u} \in \mathbb{R}^m$ to the vector of trainable coefficients $b^\sigma(\hat{u}) \in \mathbb{R}^K$.

On the other hand, the trunk network also has two components: one for the mean and another for the log standard deviation. Each component consists of a set of shared layers and a few independent layers. The trunk component for the mean maps the location $x \in K_2$ within the output function domain $K_2 \subset \mathbb{R}^d$ to the vector of K trainable basis functions $\tau^\mu(x) \in \mathbb{R}^K$. Similarly, the trunk component for the log standard deviation maps the location $x \in K_2$ to the vector of trainable basis functions $\tau^\sigma(x) \in \mathbb{R}^K$.

To train the Prob-DeepONet parameters $\theta = \{\theta_\mu, \theta_\sigma\}$, we minimize the negative log likelihood:

$$\mathcal{L}_{\text{nll}}(\theta) = -\frac{1}{2N}\left(\sum_{i=1}^{N}\left(\frac{\left(G^{(i)} - \mu_{\theta_\mu}(\hat{u}^{(i)})(x^{(i)})\right)^2}{\sigma_{\theta_\sigma}(\hat{u}^{(i)})(x^{(i)})^2} + 2\log\sigma_{\theta_\sigma}(\hat{u}^{(i)})(x^{(i)})\right) + N\log 2\pi\right),$$

using the dataset of N triplets: $\mathcal{D} = \left\{\hat{u}^{(i)}, x^{(i)}, G^{(i)}\right\}_{i=1}^{N}$.

Similar to B-DeepONet, Prob-DeepONet only provides a heuristic estimate of uncertainty through the standard deviation. As demonstrated in our paper, constructing prediction intervals using Prob-DeepONet requires heavy hyper-parameter optimization to achieve a high confidence level (e.g., 95%) for a fixed test dataset. However, it is not guaranteed that these heuristic prediction intervals, developed using Prob-DeepONet and the estimated standard deviation, are reliable for other test examples. Therefore, the first objective of this paper is to enhance our previously proposed methods to transform these heuristic prediction intervals into rigorous prediction intervals. To accomplish this, we will employ split conformal prediction [49,53], which will be described in the following section.

3. Conformal prediction for deep operator network regression

In this section, we will first demonstrate how to use split conformal prediction [49,53] to create reliable prediction intervals using the heuristic uncertainty estimates $\sigma(\hat{u})(x)$ obtained from a trained Prob-DeepONet and the trained M-ensemble generated by B-DeepONet. Next, we will propose a novel Quantile-DeepONet extension that is better suited for developing a conformalized Quantile-DeepONet regression algorithm.

3.1. Split conformal prediction for DeepONet

Within this section, we employ split conformal prediction [49, 53], recognized as the predominant conformal prediction approach, facilitating Prob- and B-DeepONet in delivering robust prediction intervals. Formally, given any DeepONet test triplet $(\hat{u}_{\text{test}}, x_{\text{test}}, G_{\text{test}})$, our goal is to construct a marginal distribution-free prediction interval $C(\hat{u}_{\text{test}}, x_{\text{test}})$ that likely contains the operator target G_{test}. That is, given a miscoverage rate $\alpha \in (0,1)$, we aim to achieve:

$$\mathbb{P}\{G_{\text{test}} \in C(\hat{u}_{\text{test}}, x_{\text{test}})\} \geq 1 - \alpha. \tag{1}$$

To achieve this goal, split conformal prediction (see Algorithm 1) proceeds as follows.

Trained DeepONet model. First, we have access to trained B- or Prob-DeepONet models. These models predict the target operator G_{test} by using the mean output $\mu(\hat{u}_{\text{test}})(x_{\text{test}})$ and provide an estimate of the uncertainty through the standard deviation output $\sigma(\hat{u}_{\text{test}})(x_{\text{test}})$. It is important to note that, as described in our previous works [21,27], the standard deviation σ produces a larger value when there is increased uncertainty about the test input.

Score function. Then, given μ and σ, we define a score function for any DeepONet data triplet $(\hat{u}, x, G)$:

$$s(\hat{u}, x, G) = \frac{|G - \mu(\hat{u})(x)|}{\sigma(\hat{u})(x)}. \tag{2}$$

Note that this score function acts as a correction factor [49]; for our DeepONet measure of uncertainty, i.e., $s(\hat{u}, x, G)\cdot\sigma(\hat{u})(x) = |G - \mu(\hat{u})(x)|$.

Calibration quantile. Using a given calibration dataset of size n, denoted as $\{\hat{u}^{(i)}, x^{(i)}, G^{(i)}\}_{i=1}^{n}$, we compute the calibration scores $s_1 = s(\hat{u}^{(1)}, x^{(1)}, G^{(1)}), \ldots, s_n = s(\hat{u}^{(n)}, x^{(n)}, G^{(n)})$. Next, we calculate $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha)\rceil}{n}$ quantile of the calibration scores $s_1, \ldots, s_n$, where $\lceil\cdot\rceil$ denotes the ceil function. It is important to note that the calibration dataset must satisfy the exchangeability property [49], which is true in most DeepONet settings where the training, test, and calibration data are independently and identically distributed (i.i.d.). Finally, it is worth noting (see [57] for more details) that if the conformity scores s_i are almost surely distinct, then the prediction interval is nearly perfectly calibrated, that is [57],

$$\mathbb{P}\{G_{\text{test}} \in C(\hat{u}_{\text{test}}, x_{\text{test}})\} \leq 1 - \alpha + \frac{1}{1+n}. \tag{3}$$

Prediction intervals with coverage guarantees. We use the quantile $\hat{q}$ to construct rigorous prediction intervals for any new DeepONet test example $(\hat{u}_{\text{test}}, x_{\text{test}})$:

$$\begin{aligned} C(\hat{u}_{\text{test}}, y_{\text{test}}) &= \{G : s(\hat{u}_{\text{test}}, y_{\text{test}}, G) \leq \hat{q}\} \\ &\equiv \{G : |G - \mu(\hat{u}_{\text{test}})(x_{\text{test}})| \leq \hat{q}\sigma(\hat{u}_{\text{test}})(x_{\text{test}})\}. \end{aligned} \tag{4}$$

These prediction intervals hold for any distribution of the data and are rigorous in the sense that they satisfy property (1), which was demonstrated in the conformal coverage guarantee theorem 3.1 replicated next for completeness.

Theorem 3.1. ***Conformal calibration coverage guarantee theorem*** *[49,50]. Suppose $\{X_i, Y_i\}_{i=1}^{n}$ and (X_{test}, Y_{test}) are i.i.d. Then define $\hat{q}$ as*

$$\hat{q} = \inf\left\{q : \frac{|i : s(X_i, Y_i) \leq q|}{n} \geq \frac{\lceil (n+1)(1-\alpha)\rceil}{n}\right\}$$

and the resulting prediction sets as

$$C(X_{test}) = \{y : s(X_{test}, y) \leq \hat{q}\}.$$

Then,

$$\mathbb{P}\{Y_{test} \in C(X_{test})\} \geq 1 - \alpha.$$

We conclude this section by noting that the above theorem holds for any score function s.

3.2. Conformal quantile DeepONet regression

In the previous section, we demonstrated how to construct finite-sample, distribution-free, and adaptive prediction intervals of the form $\mu(\hat{u})(x) \pm \hat{q}\sigma(\hat{u})(x)$ for operator targets G using the standard deviation σ as a measure of uncertainty for DeepONets. We will show in Section 4 that this approach effectively quantifies uncertainty in DeepONets. However, as stated in [49], there is no evidence to suggest that the heuristic measure of uncertainty σ for B- and Prob-DeepONet is strongly

Algorithm 1: Split Conformal Prediction to Enhance B- and Prob-DeepONet

Input: Calibration data: $\{\hat{u}^{(i)}, x^{(i)}, G^{(i)}\}_{i=1}^{n}$ where $G^{(i)} = G(\hat{u}^{(i)})(x^{(i)})$ is the exact output function values, miscoverage level: $\alpha \in (0,1)$, and mean and standard deviation models: μ and σ, obtained from the trained M-ensemble of B-DeepONet or a trained Prob-DeepONet.

Process:

Use μ and σ models to compute the calibration scores:

$$s_1 = s\left(\hat{u}^{(1)}, x^{(1)}, G^{(1)}\right), \ldots, s_n = s\left(\hat{u}^{(n)}, x^{(n)}, G^{(n)}\right),$$

where

$$s(\hat{u}, x, G) = \frac{|G - \mu(\hat{u})(x)|}{\sigma(\hat{u})(x)}.$$

Compute $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$–th quantile of the calibration scores $s_1, \ldots, s_n$.

Return: Rigorous prediction intervals (i.e., that satisfy (1)) for any new test example $(\hat{u}_{\text{test}}, x_{\text{test}})$:

$$C(\hat{u}_{\text{test}}, x_{\text{test}}) = \left[\mu(u_{\text{test}})(x_{\text{test}}) - \hat{q}\sigma(u_{\text{test}})(x_{\text{test}}), \mu(u_{\text{test}})(x_{\text{test}}) + \hat{q}\sigma(u_{\text{test}})(x_{\text{test}})\right].$$

correlated with the quantiles of the operator target distribution. Therefore, it is more natural to expect that a DeepONet method that estimates the quantiles for the target operator G distribution will enhance the use of split conformal predictions. Let us now introduce this novel Quantile-DeepONet.

Quantile-DeepONet. Our goal is to design a Quantile-DeepONet that can approximate the conditional quantiles $t_{\alpha/2}$ and $t_{1-\alpha/2}$, given (i) a miscoverage level $\alpha \in (0,1)$ (Note that the miscoverage level refers to the probability that a prediction interval does not contain the true value being estimated.), (ii) a discretized input function $\hat{u} \in \mathbb{R}^m$, and (iii) a location within the output function domain $x \in K_2$. To provide a complete understanding, let us recall the mathematical definition of a conditional quantile function. The γth conditional quantile is defined as follows:

$$t_\gamma(x) = \inf\{y \in \mathbb{R} : F(y|X = x) \geq \gamma\},$$

where

$$F(y|X = x) = \mathbb{P}\{Y \leq y|X = x\},$$

is the conditional distribution of Y given $X = x$. In other words, t_γ provides us with information about the percentile of the distribution of Y when conditioned on $X = x$.

Similar to Prob-DeepONet, Quantile-DeepONet approximates quantiles using the following linear trainable representations:

$$t_\gamma(\hat{u})(x) \approx \hat{t}_{\theta_\gamma}(\hat{u})(x) = \sum_{k=1}^{K} b_k^\gamma(\hat{u}) \cdot \tau_k^\gamma(x),$$

for $\gamma \in \{\alpha/2, 1-\alpha/2\}$. Therefore, Quantile-DeepONet consists of two sub-networks: the branch network and the trunk network.

The *branch* network comprises two components: one for the $(\alpha/2)$th quantile and another for the $(1-\alpha/2)$th quantile. Each component consists of a set of shared trainable layers and a few independent trainable layers. Additionally, each component maps the discretized input $\hat{u} \in \mathbb{R}^m$ to a vector of K trainable coefficients $b^\gamma(\hat{u}) \in \mathbb{R}^K$, where $\gamma \in \{\alpha/2, 1-\alpha/2\}$. Similarly, the *trunk* network consists of two components, one for each quantile. These two trunk components also have shared and independent trainable layers. They map the location $x \in K_2$ to a vector of K basis functions $\tau^\gamma(x) \in \mathbb{R}^m$, where $\gamma \in \{\alpha/2, 1-\alpha/2\}$.

We can train the Quantile-DeepONet parameters $\theta = \{\theta_{\alpha/2}, \theta_{1-\alpha/2}\}$ by minimizing the following loss function for $\gamma \in \{\alpha/2, 1-\alpha/2\}$:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \rho_\gamma\left(G^{(i)}, \hat{t}_{\theta_\gamma}\left(\hat{u}^{(i)}\right)\left(x^{(i)}\right)\right),$$

on the dataset of N triplets: $D = \left\{\hat{u}^{(i)}, x^{(i)}, G^{(i)}\right\}_{i=1}^{N}$. In the above expression, the loss function ρ_γ refers to the pinball loss [57,63,64]:

Algorithm 2: Split Conformal Quantile DeepONet Regression

Input: Calibration data: $\{\hat{u}^{(i)}, x^{(i)}, G^{(i)}\}_{i=1}^{n}$, miscoverage level: $\alpha \in (0,1)$, and trained quantile-DeepONet outputs $\hat{t}_{\alpha/2}$ and $\hat{t}_{1-\alpha/2}$.

Process:

Use the trained quantile-DeepONet to compute the calibration scores:

$$s_1 = s\left(\hat{u}^{(1)}, x^{(1)}, G^{(1)}\right), \ldots, s_n = s\left(\hat{u}^{(n)}, x^{(n)}, G^{(n)}\right),$$

where

$$s(u, x, G) = \max\left\{\hat{t}_{\theta^*_{\alpha/2}}(u)(x) - G, G - \hat{t}_{\theta^*_{1-\alpha/2}}(u)(x)\right\}.$$

Compute $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the calibration scores $s_1, \ldots, s_n$.

Return: Rigorous prediction intervals for any new test example $(\hat{u}_{\text{test}}, x_{\text{test}})$:

$$C(\hat{u}_{\text{test}}, x_{\text{test}}) = \left[\hat{t}_{\theta^*_{\alpha/2}}(\hat{u}_{\text{test}})(x_{\text{test}}) - \hat{q}, \hat{t}_{\theta^*_{1-\alpha/2}}(\hat{u}_{\text{test}})(x_{\text{test}}) + \hat{q}\right].$$

$$\rho_\gamma(y, \hat{y}) = \begin{cases} \gamma(y - \hat{y}) & \text{if } y - \hat{y} > 0, \\ (1-\gamma)(\hat{y} - y) & \text{otherwise.} \end{cases}$$

Once trained, a Quantile-DeepONet can provide an estimate of the prediction interval for any new test sample:

$$\hat{C}(\hat{u}_{\text{test}}, x_{\text{test}}) = \left[\hat{t}_{\theta^*_{\alpha/2}}\left(\hat{u}_{\text{test}}\right)\left(x_{\text{test}}\right), \hat{t}_{\theta^*_{1-\alpha/2}}\left(\hat{u}_{\text{test}}\right)\left(x_{\text{test}}\right)\right],$$

where $\theta^*_{\alpha/2}$ and $\theta^*_{1-\alpha/2}$ denote the optimizers of the corresponding losses. However, the estimate provided above fails to meet the coverage probability requirement stated in (1). To satisfy this requirement, we need to incorporate conformal prediction. It is worth noting that the combination of quantile and conformal prediction is referred to as Conformal Quantile Regression (CQR) in the literature [57], and we will now describe its application to Quantile-DeepONets.

Conformal Quantile-DeepONet Regression. The details of the split conformal Quantile-DeepONet regression procedure are provided in Algorithm 2. This procedure begins by defining a suitable score function.

Score function. Given the trained Quantile-DeepONet, we compute the score for any DeepONet data triplet $(\hat{u}, x, G)$ as follows:

$$s(u, x, G) = \max\left\{\hat{t}_{\theta^*_{\alpha/2}}(u)(x) - G, G - \hat{t}_{\theta^*_{1-\alpha/2}}(u)(x)\right\}.$$

Table 1
Summary of the number of training (N) and calibration (n) data samples for each experiment. Each sample is a triplet consisting of $\left(\hat{u}^{(i)}, x^{(i)}, G^{(i)}\right)$.

Experiment	# N training	# n calibration
The nonlinear pendulum equation	5000	500
The diffusion-reaction system	10 000	1000
The viscous Burgers' equation	30 000	3000

Table 2
Summary of the width and depth of the trunk and branch feed-forward neural networks used in each experiment. For the depth of Prob- and Quantile-DeepONets, we use the notation **shared(independent)** layers.

Experiment	Trunk depth	Trunk width	Branch depth	Branch width
The nonlinear pendulum	3(1)	100	3(1)	100
The diffusion-reaction	4(1)	100	4(1)	100
The viscous Burgers'	5(1)	128	5(1)	128

Note that if the operator target G satisfies $G < \hat{t}_{\theta^*_{\alpha/2}}(u)(x)$ or $G > \hat{t}_{\theta^*_{1-\alpha/2}}(u)(x)$, then the score represents the amount of error incurred by Quantile-DeepONet.

Calibration quantile. As before, using a given calibration i.i.d dataset of size n, denoted as $\{\hat{u}^{(i)}, x^{(i)}, G^{(i)}\}_{i=1}^{N}$, where $G^{(i)} = G(\hat{u}^{(i)})(x^{(i)})$, we compute the calibration scores $s_1 = s(\hat{u}^{(1)}, x^{(1)}, G^{(1)}), \ldots, s_n = s(\hat{u}^{(n)}, x^{(n)}, G^{(n)})$. Then, we calculate $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the calibration scores $s_1, \ldots, s_n$.

Prediction intervals with coverage guarantees. We use the quantile $\hat{q}$ to construct rigorous prediction intervals for any new DeepONet test example $(\hat{u}_{\text{test}}, x_{\text{test}})$:

$$C(\hat{u}_{\text{test}}, x_{\text{test}}) = \left[\hat{t}_{\theta^*_{\alpha/2}}(\hat{u}_{\text{test}})(x_{\text{test}}) - \hat{q}, \hat{t}_{\theta^*_{1-\alpha/2}}(\hat{u}_{\text{test}})(x_{\text{test}}) + \hat{q}\right]. \tag{5}$$

These prediction intervals are adaptive, distribution-free, and satisfy the coverage probability property (1). The validity of this property was demonstrated in [57] by extending theorem 3.1 to the case of CQR.

4. Numerical experiments

In this section, we assess the ability of conformalized-DeepONets (conformal prediction combined with B-DeepONet, Prob-DeepONet, or Quantile-DeepONet) to generate prediction intervals with coverage guarantees. We illustrate this through three experiments: (i) the nonlinear pendulum (Section 4.1), (ii) the diffusion-reaction system 4.2, and (iii) the viscous Burgers' equation 4.3.

Datasets. To train our DeepONet models, we used a dataset consisting of N samples, each in the form of a triplet $\left(\hat{u}^{(i)}, x^{(i)}, G^{(i)}\right)$. The specific size of the training dataset, N, for each experiment is provided in Table 1. For the calibration dataset, we used $n = \frac{N}{10}$ samples, with details outlined in Table 1. Finally, for testing purposes, we evaluated all our DeepONet models using 100 trajectories per experiment. Each trajectory included the target operator values $G(u)(X_{\text{test}})$ evaluated on a mesh $X_{\text{test}} \subset K_2$ composed of 100 uniformly distributed location points within the output function domain K_2. It should be noted that the testing mesh size is much more dense than the training mesh, i.e., our experiments also demonstrate the extrapolation of the proposed methods.

Neural Networks. We used the feed-forward neural network (FNN) architecture to build the branch and trunk sub-networks. The corresponding width and depth for each experiment, including the shared and independent layers for Prob- and Quantile-DeepONet, are listed in Table 2. We used the ReLU function as the activation function for each input and hidden layer. Finally, for B-DeepONet, we used an ensemble of size $M = 200$.

Note that the architectures of both quantile and probabilistic DeepONets are the same, closely following the vanilla DeepONet structure with one key difference in the final layer. In this layer, the trunk and branch are divided into two independent single layers, allowing for the computation of two separate dot products—one for each quantile in the quantile DeepONet, or one for the mean and one for the log of the standard deviation in the probabilistic DeepONet.

Optimizer and Sampler. For Prob- and Quantile-DeepONet, we used the Adam optimizer [65] with default hyperparameters and a learning rate $\eta = 10^{-3}$. We adjusted the learning rate using the reduced on plateau scheduler. For B-DeepONet, we used the stochastic gradient replica-exchange sampling algorithm [21,66,67] with standard hyperparameters.

Baselines. We compared the proposed Conformalized-DeepONets (which combine conformal prediction with B-DeepONet, Prob-DeepONet, or Quantile-DeepONet) against the following baselines: B-DeepONet [21], Prob-DeepONet [27], and Quantile-DeepONet (Section 3.2). Specifically, for each conformalized and baseline model, we calculated the coverage over the test dataset. This coverage is determined by

$$C_j = \frac{1}{n_{\text{eval}}} \sum_{i=1}^{n_{\text{eval}}} \mathbb{1}\left\{G^{(i,j)} \in C_j\left(\hat{u}^{(j)}, x^{(i,j)}\right)\right\}, \text{ for } j = 1, \ldots, n_{\text{traj}}.$$

In other words, we calculate the coverage of n_{traj} test trajectories and each C_j is the ratio of location points whose reference solutions are in the established prediction interval. These are evaluated at $n_{\text{eval}} = 100$ uniformly spaced points within the output function domain.

4.1. Experiment: The nonlinear pendulum

In this experiment, we consider the nonlinear pendulum subject to external forcing, represented by:

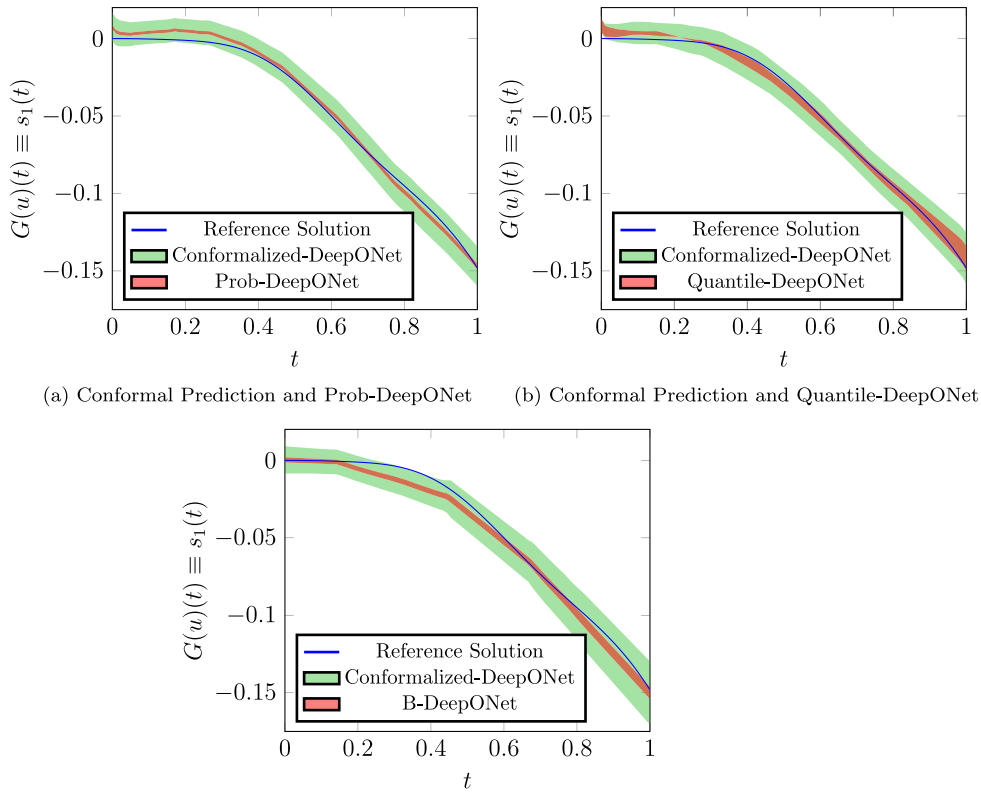
$$\frac{ds_1}{dt} = s_2,$$
$$\frac{ds_2}{dt} = -k \sin s_1 + u(t),$$

with the initial condition $(s_1(0), s_2(0))^\top = (0, 0)^\top$ and time domain $[0, 1.0]$ (s). In the above, $(s_1(t), s_2(t))^\top$ is the state, $k = 1.0$ is a constant determined by the acceleration due to gravity and the length of the pendulum, and $u(t)$ is the time-dependent external forcing with time domain $[0, 1]$ (s).

Our aim is to approximate the solution operator $G : u(t) \mapsto s_1(t)$, which maps the external forcing input to the state s_1. We draw samples of the external forcing inputs for training, calibration, and testing from the mean-zero Gaussian Random Field (GRF) $u \sim \mathcal{G}(0, k_\ell(t_1, t_2))$. This GRF has a radial-basis function covariance kernel $k_\ell(t_1, t_2)$ with a length-scale of $\ell = 0.2$. Finally, we discretize each sampled input u using $m = 100$ sensors.

Checking Coverage. We evaluated the ability of all conformalized-DeepONet models and baseline models to generate prediction intervals (with misscoverage rate $\alpha = 0.05$) with coverage guarantees for trajectories in the test dataset. Fig. 2 illustrates the prediction intervals for a test trajectory randomly selected from the test dataset. Furthermore, Table 3 presents the average coverage achieved by both conformalized-DeepONets and baseline models for all trajectories in the test dataset.

Fig. 2 and Table 3 offer significant insights. Firstly, the conformalized DeepONets (Bayesian, Probabilistic and Quantile) deliver the



(a) Conformal Prediction and Prob-DeepONet

(b) Conformal Prediction and Quantile-DeepONet

(c) Conformal Prediction and B-DeepONet

Fig. 2. Prediction intervals for a random test trajectory of the nonlinear pendulum experiment given a miscoverage rate $\alpha = 0.05$. **(a)** Prediction intervals for the Conformalized Prob-DeepONet and the baseline Prob-DeepONet. **(b)** Prediction intervals for the Conformalized Quantile-DeepONet and the baseline Quantile-DeepONet. **(c)** Prediction intervals for the Conformalized B-DeepONet and the baseline B-DeepONet. Please refer to Table 3 for the summary of the average coverage rate for all testing trajectories with different approaches.

Table 3
The average coverage percentage for the prediction intervals (with a miscoverage rate of $\alpha = 0.05$) calculated based on the 100 test trajectories in the nonlinear pendulum experiment.

DeepONet model	Average coverage % ($\alpha = 0.05$)
Conformalized Prob-DeepONet	**94.69%**
Conformalized Quantile-DeepONet	**95.86%**
Conformalized B-DeepONet	**96.41%**
Prob-DeepONet	3.80%
Quantile-DeepONet	55.31%
B-DeepONet	50.59%

desired coverage on average, with a miscoverage rate of $\alpha = 0.05$. Moreover, the Quantile-DeepONet significantly outperforms the Prob-DeepONet and B-DeepONet. Lastly, the Probabilistic DeepONet displays the poorest average coverage. These findings support the conclusions from our previous paper, which highlighted the necessity of extensive hyperparameter optimization to improve the quality of prediction intervals for Prob-DeepONet and B-DeepONet. However, the improved results from conformalized-DeepONet come with the added computational cost compared to the original DeepONets. This extra cost mainly arises from the n forward passes required to compute the scores s.

Finally, Fig. 3 shows the distribution of coverages (with a miscoverage rate of $\alpha = 0.05$) for all prediction intervals obtained by applying the conformalized Quantile-DeepONet to all 100 test trajectories of the nonlinear pendulum example. As anticipated, the coverages concentrate around $1 - \alpha$, that is, with 95% coverage. Next, we present two ablation studies showing the adaptivity and effect of the calibration dataset size.

Ablation Study I: Adaptivity of Prediction Intervals. We tested the proposed conformalized Prob-DeepONet for its ability to provide adaptive prediction intervals — that is, prediction intervals of varying lengths for each $x \in K_2$, while still ensuring coverage guarantees. The adaptive coverage will reduce the over-confidence issue of many algorithms. To accomplish this, we applied the conformalized Prob-DeepONet to all 100 test trajectories of the pendulum experiment, with the goal of analyzing the distribution of the prediction intervals' lengths. Fig. 4 illustrates the distribution of the lengths of prediction intervals. This distribution demonstrates that the prediction intervals are adaptive, meaning their length varies for each $x \in K_2$. If this was not the situation, the distribution would be focused around a specific length. Clearly, our experiment indicates that this is not the scenario.

Ablation Study II: Testing the Effect of the Calibration Dataset Size n. We tested the effect of varying the calibration dataset size n on the distribution of coverages for the nonlinear pendulum example and a miscoverage rate of $\alpha = 0.05$.

To this end, we varied n over the set $\{500, 1000, 5000, 10000\}$. For each n, we computed the distribution of coverages. This was done for R rounds ($R = 200$), where we generated a calibration dataset of size n and a validation dataset of size n_{val}. Each dataset contains DeepONet triplets of the form $(\hat{u}, x, G)$. We then calculated the coverages as follows.

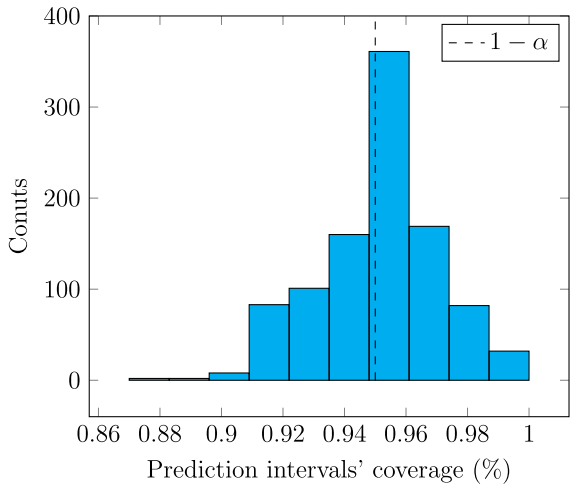


Fig. 3. The distribution of prediction intervals' coverages (in percentages), obtained by using the conformalized Quantile-DeepONet on the 100 test trajectories of the nonlinear pendulum experiment, with a specified miscoverage rate of $\alpha = 0.05$.

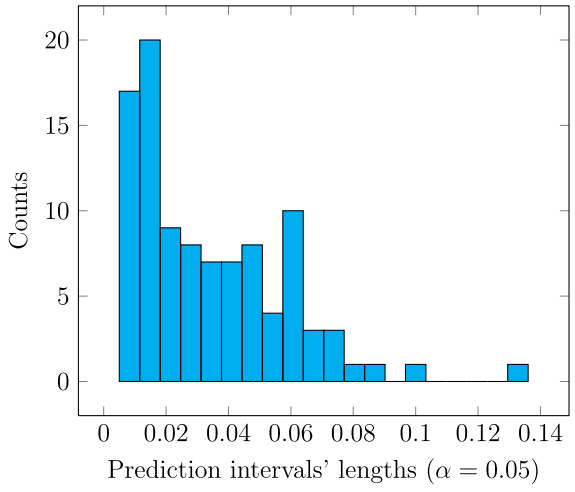


Fig. 4. The distribution of the lengths of the prediction intervals (given a miscoverage rate $\alpha = 0.05$) predicted using the proposed conformalized Prob-DeepONet for the 100 test trajectories of the nonlinear pendulum experiment.

$$C_k = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \mathbb{1}\left\{G^{(i,k)} \in C_k\left(\hat{u}^{(i,k)}, x^{(i,k)}\right)\right\}, \quad \text{for } k = 1, \ldots, R.$$

Finally, we obtained the distribution of the obtained coverages $\{C_k\}_{k=1}^{R}$, which we plot for each n.

Note that to prevent the generation of R distinct datasets of size $n + n_{\text{val}}$, we used the approach proposed in [49]. This method creates just one dataset of size $n + n_{\text{val}}$ for all R rounds. Then, in each round, we shuffled this dataset and used the initial n samples for calibration and the remaining n_{val} samples for validation.

Fig. 5 illustrates the obtained distribution of coverages for $n \in \{500, 1000, 5000, 10000\}$. Observe that for $n > 500$, the distribution, as expected, centers around 95% ($\alpha = 0.05$). Notably, for $n = 1000$, we achieve a satisfactory distribution, and the gain from increasing the calibration dataset size is minimal.

Finally, for $n = 500$, the distribution peaks at a value larger than 95%. This could be due to the small number of calibration samples, which may not allow for a proper characterization of the coverage. However, even in this extreme case, conformalized-DeepONet still produces satisfactory prediction intervals around $1 - \alpha$.

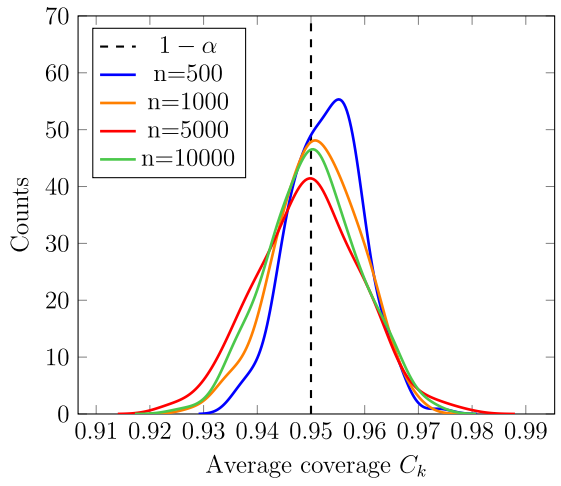


Fig. 5. The empirical distribution of coverages C_k for different values of the calibration dataset size $n \in \{500, 1000, 5000, 10000\}$. We observe that for $n > 500$, the distribution, as expected, centers around 95% ($\alpha = 0.05$). Notably, for $n = 1000$ and this nonlinear pendulum dataset, we achieve a satisfactory distribution, and the gain from increasing n is minimal.

Ablation Study III: Verifying the Effect of the Miscoverage Rate α. To conclude this example, we examined the effect of varying the miscoverage rate α over the set $0.05, 0.1, 0.2$. As shown in Fig. 6, the width of the resulting prediction intervals decreases as α increases, while maintaining approximate coverage levels of 95%, 90%, and 80%, respectively, as expected.

4.2. Experiment: The diffusion-reaction system

In the second experiment, we examine the following diffusion-reaction system with a source term, $u(x)$, as described below:

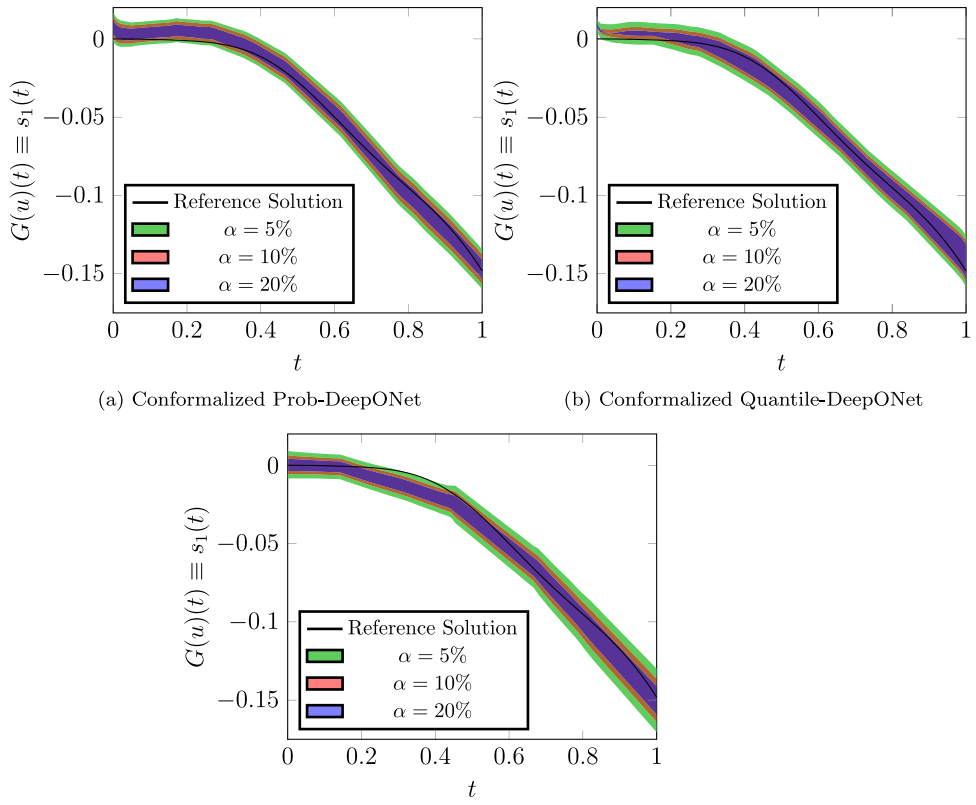
$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), \qquad x \in [0, 1], \ t \in [0, 1],$$

with zero boundary/initial conditions. In the above, $D = 0.01$ is the diffusion coefficient, and $k = 0.01$ is the reaction rate.

Our goal is to approximate the solution operator $G : u(x) \mapsto s(x, 1.0)$, that is, the mapping from the source term to the solution at the terminal time. As in the nonlinear pendulum experiment, we sample the input function (the source term $u(x)$) from the non-zero Gaussian Random Field $u \sim \mathcal{G}(0, k_\ell(x_1, x_2))$ with a radial basis function kernel and a length-scale of $\ell = 0.2$. Finally, we discretize the sampled source terms using $m = 100$ sensors.

Checking Coverage. We assessed the ability of all conformalized and baseline DeepONet models to generate appropriate prediction intervals for a miscoverage rate of $\alpha = 0.05$. Fig. 7 displays the prediction intervals predicted by the proposed conformalized-DeepONet and baseline models for a randomly selected trajectory from the test dataset. Furthermore, Table 4 presents the average coverage of the prediction intervals predicted by both the conformalized and baseline DeepONet models for all $n_{\text{traj}} = 100$ test trajectories of the diffusion-reaction system.

Fig. 7 shows that the prediction intervals predicted by the conformalized-DeepONet appear to be adequate. Table 4 confirms this, indicating that the average coverage for the conformalized-DeepONets is as expected (≈95%), considering a miscoverage rate of $\alpha = 0.05$. The baseline Quantile-DeepONet also has a good coverage level compared to the Prob-DeepONet and B-DeepONet. This superior coverage might enable the conformalized Quantile-DeepONet to perform slightly better than the conformalized Prob-DeepONet and conformalized B-DeepONet, which must compensate for its baseline's poor coverage.



(a) Conformalized Prob-DeepONet

(b) Conformalized Quantile-DeepONet

(c) Conformalized B-DeepONet

Fig. 6. Prediction intervals for the nonlinear pendulum with miscoverage rates of $\alpha = 5\%$, 10%, and 20%, using Conformalized Probabilistic-, Quantile-, and B-DeepONets.

Table 4
The average coverage percentage for the prediction intervals (with a miscoverage rate of $\alpha = 0.05$) calculated based on the 100 test trajectories in the diffusion-reaction experiment.

DeepONet model	Average coverage % ($\alpha = 0.05$)
Conformalized Prob-DeepONet	**96.26%**
Conformalized Quantile-DeepONet	**95.64%**
Conformalized B-DeepONet	**94.18%**
Prob-DeepONet	8.99%
Quantile-DeepONet	83.09%
B-DeepONet	60.67%

The Two-Dimensional Diffusion-Reaction System. In the previous analysis, we computed the solution operator for the terminal time $t = 1.0$. Here, we extend this analysis to the two-dimensional case by approximating the solution operator for varying values of $t \in [0, 1]$. In this scenario, the scores are no longer computed at different x values alone but at different (x, t) tuples. To conformalize our DeepONets, we select a grid of 100 equidistributed x points and 120 equidistributed t values within $[0, 1]$, alongside 300 corresponding solutions of the two-dimensional diffusion-reaction system.

Fig. 8 presents the conformalized quantiles for a randomly selected test solution. Additionally, Table 5 shows the corresponding coverage values. The results demonstrate that the conformalized DeepONets successfully achieved the desired coverage for the two-dimensional problem, whereas the non-conformalized DeepONet failed to properly quantify the uncertainty in this case.

We conclude the two-dimensional example by assessing the robustness of conformal prediction as the size of the calibration dataset increases. As shown in Fig. 9, and consistent with theoretical expectations, when the calibration dataset is small, the approximate coverage tends to be loose. However, as the size of the calibration dataset grows, the coverage converges more closely to the desired target.

4.3. Experiment: The viscous Burgers' equation

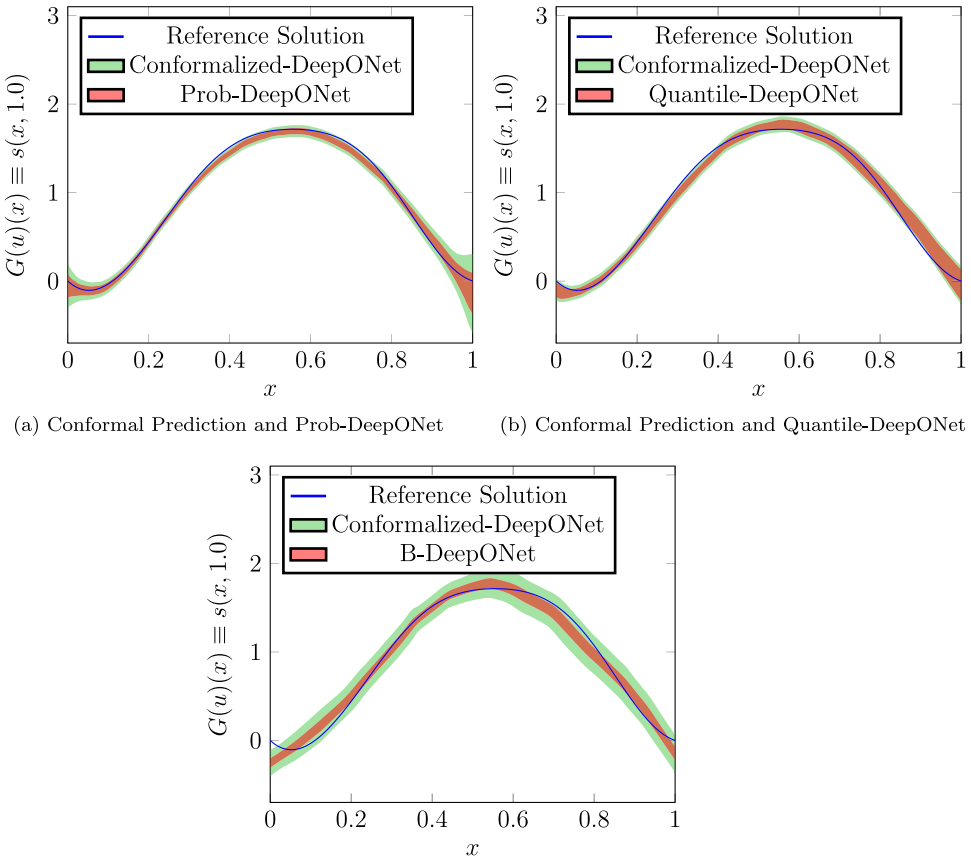
In this section, we explore a common example found in most operator learning literature: the viscous Burgers' equation:

$$\frac{\partial u_s}{\partial t} + \frac{1}{2}\frac{\partial (u_s^2)}{\partial x} = \alpha \frac{\partial^2 u_s}{\partial x^2}, \quad x \in [0, 2\pi],\ t \in [0, 0.3]$$
$$u_s(x, 0) = u_s^0(x),$$
$$u_s(0, t) = u_s(2\pi, t),$$

where $u_s^0(x)$ is the initial condition that depends on the parameter s. In the above, the viscosity is set to $\alpha = 0.05$.

Our objective is to estimate the solution operator $G : u_s^0(x) \mapsto u_s(x, 0.3)$, which is the mapping from the initial condition to the solution at the terminal time. We generated initial conditions using a sum of two Gaussian distributions with a uniformly randomized weight from $[0, 5]$. The means and standard deviations of the two distributions were sampled uniformly from $[0, 2\pi]$ and $[0.1, 1]$, respectively.

Checking Coverage. We evaluated the proposed conformalized-DeepONet models' capacity to produce rigorous prediction intervals. These intervals have a specified miscoverage rate of $\alpha = 0.05$ for test trajectories generated from the viscous Burgers' model. We also compared these proposed models with the baseline models. Fig. 10



(a) Conformal Prediction and Prob-DeepONet

(b) Conformal Prediction and Quantile-DeepONet

(c) Conformal Prediction and B-DeepONet

Fig. 7. Prediction intervals for a random test trajectory of the diffusion-reaction experiment given a miscoverage rate $\alpha = 0.05$. **(a)** Prediction intervals for the Conformalized Prob-DeepONet and the baseline Prob-DeepONet. **(b)** Prediction intervals for the Conformalized Quantile-DeepONet and the baseline Quantile-DeepONet. **(c)** Prediction intervals for the Conformalized B-DeepONet and the baseline B-DeepONet. The average coverage rate for all samples for all approaches are presented in Table 4..

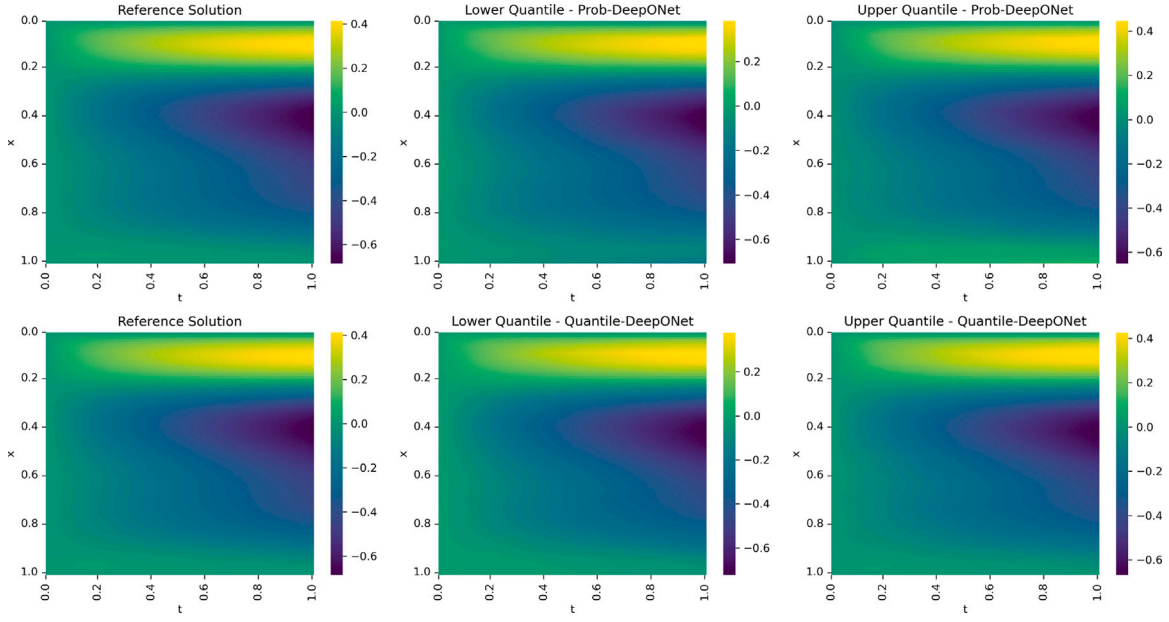


Fig. 8. Prediction interval (conformalized lower and upper quantiles) for a randomly selected test solution of the two-dimensional diffusion-reaction system, with a miscoverage rate of $\alpha = 0.05$.

Table 5
The average coverage (%) of prediction intervals (with a miscoverage rate of $\alpha = 0.05$) calculated using 100 test solutions of the two-dimensional diffusion-reaction system.

DeepONet model	Average coverage % ($\alpha = 0.05$)
Conformalized Prob-DeepONet	**95.12%**
Conformalized Quantile-DeepONet	**95.92%**
Conformalized B-DeepONet	**96.62%**
Prob-DeepONet	1.52%
Quantile-DeepONet	82.13%
B-DeepONet	69.73%

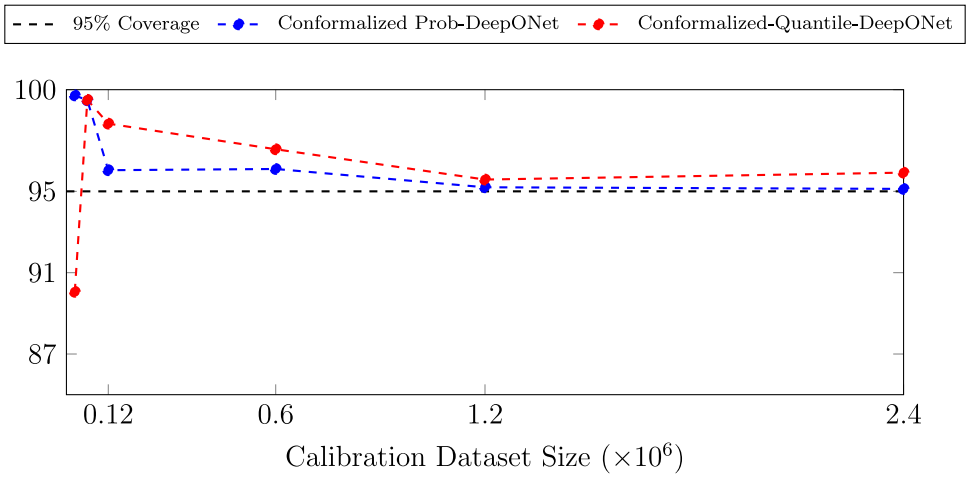
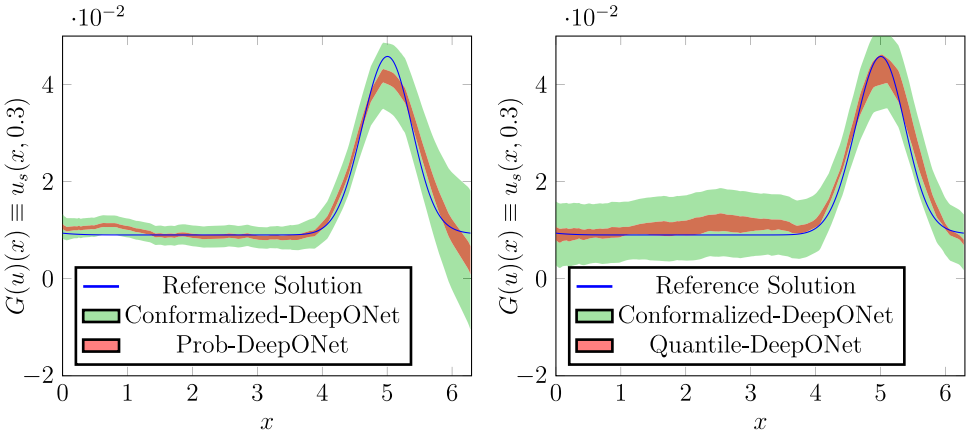


Fig. 9. Coverage (%) vs. calibration dataset size for the two-dimensional Diffusion-Reaction example using Conformalized Prob-DeepONet and Conformalized Quantile-DeepONet.



(a) Conformal Prediction and Prob-DeepONet (b) Conformal Prediction and Quantile-DeepONet

Fig. 10. Prediction intervals for a random test trajectory of the viscous Burgers' experiment given a miscoverage rate $\alpha = 0.05$. **(a)** Prediction intervals for the Conformalized Prob-DeepONet and the baseline Prob-DeepONet. **(b)** Prediction intervals for the Conformalized Quantile-DeepONet and the baseline Quantile-DeepONet. The average coverage for all testing trajectories and approaches are summarized and presented in Table 6.

presents a comparison between the conformalized and baseline models. This comparison uses the prediction interval for a randomly chosen test trajectory from the viscous Burgers' example. The figure indicates that both the conformalized Prob-DeepONet and Quantile-DeepONet effectively capture the reference solution within the prediction interval. In contrast, the baseline models do not capture several portions of the reference solution.

Moreover, Table 6 shows the average coverage of prediction intervals (with a targeted coverage rate of $\alpha = 0.05$) generated from both conformalized and baseline models for all $n_{\text{traj}} = 100$ test trajectories produced using the viscous Burgers' model. The results once again demonstrate that both the conformalized Prob-DeepONet and the conformalized Quantile-DeepONet provide (on average) the desired coverage guarantee. In contrast, both baseline models generally offer poor coverage guarantees. It is noteworthy that the conformalized Prob-DeepONet still produces satisfactory coverage results, even though these are based on the inferior results from the baseline Prob-DeepONet model.

5. Discussion

On Our Results. Our results indicate that split conformal prediction [49,51,57] is an easy-to-use methodology that offers prediction intervals with coverage guarantees. We consider it easy-to-use as it does not require any distributional assumptions and only needs a finite number of DeepONet samples. This allowed us to build an adaptive, reliable, and efficient Uncertainty Quantification (UQ) and regression framework for Deep Operator Networks.

Table 6
The average coverage percentage for the prediction intervals (with a miscoverage rate of $\alpha = 0.05$) calculated based on the 100 test trajectories in the viscous Burgers' experiment.

DeepONet model	Average coverage % ($\alpha = 0.05$)
Conformalized Prob-DeepONet	**94.88**%
Conformalized Quantile-DeepONet	**95.96**%
Prob-DeepONet	7.84%
Quantile-DeepONet	72.91%

Table 7
The average coverage percentage for the prediction intervals (with a given miscoverage rate of $\alpha = 0.05$) calculated based on the 100 high-fidelity test trajectories of the 1D jump function experiment.

DeepONet model	Average coverage % ($\alpha = 0.05$)
Multi-fidelity Conformalized Prob-DeepONet	**97.02**%
Multi-fidelity Prob-DeepONet	82.07%

Our results also indicate that split conformal prediction is model agnostic. Note that we applied it on top of two UQ DeepONet frameworks (B-DeepONet and Prob-DeepONet), proposed in our previous works, and a new UQ framework (Quantile-DeepONet) proposed here. Hence, we expect split conformal prediction to be compatible with many existing DeepONet extensions, such as Multi-fidelity DeepONet [12,68,69], BelNet [25], Fed-DeepONet [70], or D2NO [22]. To demonstrate this, we will present a simple experiment that highlights the use of split conformal prediction with Multi-fidelity DeepONets.

Conformal Prediction and Multi-Fidelity DeepONet. In this experiment, we used a multi-fidelity conformalized Prob-DeepONet model to approximate the high-fidelity solution operator for the given 1D jump function:

$$\begin{aligned} y_L(u)(x) &= \begin{cases} 0.5(6x-2)^2\sin(u) + 10(x-0.5) - 5 & x \le 0.5 \\ 0.5(6x-2)^2\sin(u) + 10(x-0.5) - 2 & x > 0.5 \end{cases} \\ y_H(u)(x) &= 2y_L(u)(x) - 20x + 20 \\ u(x) &= ax - 4 \end{aligned} \tag{6}$$

where $x \in [0,1]$ and $a \in [10,14]$. In the above, y_L is the low-fidelity solution, y_H the high-fidelity solution, and u the input.

We built our Multi-fidelity Prob-DeepONet baseline model as follows. Initially, we used a trained classical DeepONet to approximate the low-fidelity solution operator $G : u(x) \mapsto \hat{y}LF(u)(x)$. Next, we trained a Prob-DeepONet to approximate the difference between the high-fidelity and low-fidelity solution operators $(y_{HF} - y_{LF}) \mid X = (u,x) \sim \mathcal{N}(\mu(u)(x), \sigma(u)(x))$. Finally, we recovered the high-fidelity solution operator using the probabilistic model $\mathcal{N}(\mu(u)(x) + \hat{y}(u)(x), \sigma(u)(x))$.

We trained the baseline model using a dataset of $N_{LF} = 3800$ low-fidelity DeepONet triplets and $N_{HF} = 760$ high-fidelity DeepONet triplets. For calibration, we used the half the of number of high-fidelity data samples ($n = \frac{N_{HF}}{2} = 380$).

Checking Coverage. We tested the proposed Multi-Fidelity Conformalized Prob-DeepONet to create prediction intervals with coverage guarantees for a target miscoverage rate of $\alpha = 0.05$. Fig. 11 shows the prediction intervals for a randomly chosen high-fidelity test trajectory. The Multi-fidelity Conformalized model provides a suitable prediction interval, showing increased uncertainty at the high-fidelity solution's jumps and turns. Additionally, Table 7 displays the average coverage of prediction intervals (targeting a miscoverage rate of $\alpha = 0.05$), generated by both the Multi-fidelity conformalized and baseline Prob-DeepONet models for all $n_{\text{traj}} = 100$ high-fidelity test trajectories. The results suggest satisfactory coverage, which can be enhanced by using more calibration examples.

On Our Future Work. The findings discussed in this paper are undoubtedly promising and applicable in scenarios where a reliable surrogate model is required. Thus, in our future work, we plan to investigate the application of conformalized-DeepONets in real-world settings. This includes power systems [27], rib-optimization in fluid systems [36], PDE-based topology optimization [12], and forecasting extreme events. Additionally, we plan to investigate the use of conformal prediction in DeepONet settings where the properties of exchangeability and independent and identically distributed (i.i.d.) do not apply. For instance, in DeepONet extrapolation [20], DeepONet for non-autonomous systems [19], or Federated DeepONet [70].

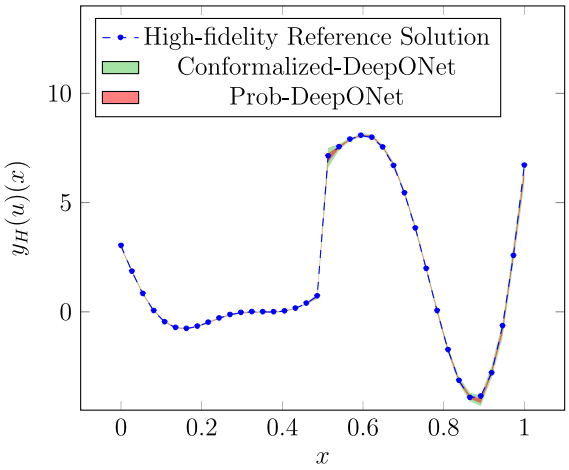


Fig. 11. Prediction intervals predicted by the Multi-fidelity conformalized Prob-DeepONet and the corresponding baseline model for a random high-fidelity test trajectory of the 1D jump function, given a miscoverage rate of $\alpha = 0.05$. We summarize the average coverage rate for all approaches in Table 7.

6. Conclusion

In the paper, we presented methods to apply split conformal prediction for providing prediction intervals for Deep Operator Network (DeepONet) prediction. These intervals come with coverage guarantees and do not require distributional assumptions. Specifically, we used split conformal prediction to enhance our previously proposed Probabilistic and Bayesian DeepONets, enabling them to predict rigorous prediction intervals at a predetermined miscoverage rate. Moreover, we designed a novel extension of DeepONet, known as Quantile-DeepONet, which estimated conditional quantiles for DeepONet predictions and provided a more natural setting for applying conformal prediction. By combining split conformal prediction with Quantile-DeepONet, we successfully developed an effective and distribution-free methodology for constructing prediction intervals with guaranteed coverage.

CRediT authorship contribution statement

Christian Moya: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Amirhossein Mollaali:** Writing –

review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Zecheng Zhang:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Lu Lu:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Guang Lin:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT in order to check grammar issues. After using ChatGPT, the authors reviewed and edited the content as needed and took full responsibility for the content of the published article.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Guang Lin reports financial support was provided by National Sleep Foundation. Lu Lu, Guang Lin reports financial support was provided by Department of Energy. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Z. Zhang acknowledges the support of the Department of Energy, USA (DE-SC0025440). L. Lu was supported by the U.S. Department of Energy Office of Advanced Scientific Computing Research under Grants No. DE-SC0025592 and No. DE-SC0025593, and the U.S. National Science Foundation under Grant No. DMS-2347833. G. Lin acknowledges the support of the National Science Foundation, USA (DMS-2053746, DMS-2134209, ECCS-2328241, and OAC-2311848), and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0023161, the Uncertainty Quantification for Multifidelity Operator Learning (MOLUcQ), USA project (Project No. 81739) and DOE–Fusion Energy Science, USA, under grant number: DE-SC0024583.

Data availability

Data will be made available on request.

References

[1] Andrew R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inform. Theory 39 (3) (1993) 930–945.
[2] Ali Rahimi, Benjamin Recht, Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning, Adv. Neural Inf. Process. Syst. 21 (2008).
[3] Maziar Raissi, Paris Perdikaris, George E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[4] Lu Lu, Xuhui Meng, Zhiping Mao, George Em Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[5] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, Liu Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[6] Wing Tat Leung, Guang Lin, Zecheng Zhang, NH-PINN: Neural homogenization-based physics-informed neural network for multiscale problems, J. Comput. Phys. (2022) 111539.
[7] Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, Madeleine Udell, Challenges in training PINNs: A loss landscape perspective, 2024, arXiv preprint arXiv:2402.01868.
[8] Tianping Chen, Hong Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (4) (1995) 911–917.
[9] Tianping Chen, Hong Chen, Approximations of continuous functionals by neural networks with application to dynamic systems, IEEE Trans. Neural Netw. 4 (6) (1993) 910–918.
[10] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, George Em Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[11] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, George Em Karniadakis, A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, Comput. Methods Appl. Mech. Engrg. 393 (2022) 114778.
[12] Lu Lu, Raphaël Pestourie, Steven G. Johnson, Giuseppe Romano, Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport, Phys. Rev. Res. 4 (2) (2022) 023210.
[13] Min Zhu, Shihang Feng, Youzuo Lin, Lu Lu, Fourier-DeepONet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness, 2023, arXiv preprint arXiv:2305.17289.
[14] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.
[15] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, Neural operator: Graph kernel network for partial differential equations, 2020, arXiv preprint arXiv:2003.03485.
[16] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, Sally M. Benson, U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow, Adv. Water Resour. 163 (2022) 104180.
[17] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, Anima Anandkumar, Fourier neural operator with learned deformations for pdes on general geometries, 2022, arXiv preprint arXiv:2207.05209.
[18] Pengzhan Jin, Shuai Meng, Lu Lu, MIONet: Learning multiple-input operators via tensor product, SIAM J. Sci. Comput. 44 (6) (2022) A3490–A3514.
[19] Guang Lin, Christian Moya, Zecheng Zhang, Learning the dynamical response of nonlinear non-autonomous dynamical systems with deep operator neural networks, Eng. Appl. Artif. Intell. 125 (2023) 106689.
[20] Min Zhu, Handi Zhang, Anran Jiao, George Em Karniadakis, Lu Lu, Reliable extrapolation of deep neural operators informed by physics or sparse observations, Comput. Methods Appl. Mech. Engrg. 412 (2023) 116064.
[21] Guang Lin, Christian Moya, Zecheng Zhang, B-DeepONet: An enhanced Bayesian DeepONet for solving noisy parametric PDEs using accelerated replica exchange SGLD, J. Comput. Phys. 473 (2023) 111713.
[22] Zecheng Zhang, Christian Moya, Lu Lu, Guang Lin, Hayden Schaeffer, D2NO: Efficient handling of heterogeneous input function spaces with distributed deep neural operators, 2023, arXiv preprint arXiv:2310.18888.
[23] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al., Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022, arXiv preprint arXiv:2202.11214.
[24] Chensen Lin, Zhen Li, Lu Lu, Shengze Cai, Martin Maxey, George Em Karniadakis, Operator learning for predicting multiscale bubble growth dynamics, J. Chem. Phys. 154 (10) (2021).
[25] Zecheng Zhang, Wing Tat Leung, Hayden Schaeffer, BelNet: Basis enhanced learning, a mesh-free neural operator, 2022, arXiv preprint arXiv:2212.07336.
[26] Zecheng Zhang, Wing Tat Leung, Hayden Schaeffer, A discretization-invariant extension and analysis of some deep operator networks, 2023, arXiv preprint arXiv:2307.09738.
[27] Christian Moya, Shiqi Zhang, Guang Lin, Meng Yue, Deeponet-grid-uq: A trustworthy deep operator framework for predicting the power grid's post-fault trajectories, Neurocomputing 535 (2023) 166–182.
[28] Joel Hayford, Jacob Goldman-Wetzler, Eric Wang, Lu Lu, Speeding up and reducing memory usage for scientific machine learning via mixed precision, 2024, arXiv preprint arXiv:2401.16645.
[29] Yixuan Sun, Christian Moya, Guang Lin, Meng Yue, Deepgraphonet: A deep graph operator network to learn and zero-shot transfer the dynamic response of networked systems, IEEE Syst. J. (2023).
[30] Minglang Yin, Nicolas Charon, Ryan Brody, Lu Lu, Natalia Trayanova, Mauro Maggioni, DIMON: Learning solution operators of partial differential equations on a diffeomorphic family of domains, 2024, arXiv preprint arXiv:2402.07250.
[31] Zhongyi Jiang, Min Zhu, Dongzhuo Li, Qiuzi Li, Yanhua O. Yuan, Lu Lu, Fourier-MIONet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration, 2023, arXiv preprint arXiv:2303.04778.

[32] Shunyuan Mao, Ruobing Dong, Lu Lu, Kwang Moo Yi, Sifan Wang, Paris Perdikaris, PPDONet: Deep operator networks for fast prediction of steady-state solutions in disk–planet systems, Astrophys. J. Lett. 950 (2) (2023) L12.

[33] Zhiping Mao, Lu Lu, Olaf Marxen, Tamer A. Zaki, George Em Karniadakis, DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators, J. Comput. Phys. 447 (2021) 110698.

[34] Patricio Clark Di Leoni, Lu Lu, Charles Meneveau, George Em Karniadakis, Tamer A. Zaki, Neural operator prediction of linear instability waves in high-speed boundary layers, J. Comput. Phys. 474 (2023) 111793.

[35] Christian Moya, Guang Lin, Tianqiao Zhao, Meng Yue, On approximating the dynamic response of synchronous generators via operator learning: A step towards building deep operator-based power grid simulators, 2023, arXiv preprint arXiv:2301.12538.

[36] Izzet Sahin, Christian Moya, Amirhossein Mollaali, Guang Lin, Guillermo Paniagua, Deep operator learning-based surrogate models with uncertainty quantification for optimizing internal cooling channel rib profiles, Int. J. Heat Mass Transfer 219 (2024) 124813.

[37] Apostolos F. Psaros, Xuhui Meng, Zongren Zou, Ling Guo, George Em Karniadakis, Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons, J. Comput. Phys. 477 (2023) 111902.

[38] Zongren Zou, Xuhui Meng, Apostolos F. Psaros, George E Karniadakis, NeuralUQ: A comprehensive library for uncertainty quantification in neural differential equations and operators, SIAM Rev. 66 (1) (2024) 161–190.

[39] Zecheng Zhang, Christian Moya, Wing Tat Leung, Guang Lin, Hayden Schaeffer, Bayesian deep operator learning for homogenized1 to fine-scale maps for multiscale PDE, 2023.

[40] Yibo Yang, Georgios Kissas, Paris Perdikaris, Scalable uncertainty quantification for deep operator networks using randomized priors, Comput. Methods Appl. Mech. Engrg. 399 (2022) 115399.

[41] Shailesh Garg, Souvik Chakraborty, VB-DeepONet: A Bayesian operator learning framework for uncertainty quantification, Eng. Appl. Artif. Intell. 118 (2023) 105685.

[42] Ling Guo, Hao Wu, Wenwen Zhou, Yan Wang, Tao Zhou, IB-UQ: Information bottleneck based uncertainty quantification for neural function regression and neural operator learning, 2023, arXiv preprint arXiv:2302.03271.

[43] Christian Moya Calderon, Guang Lin, Bayesian, multifidelity operator learning for complex engineering systems-A position paper, J. Comput. Inf. Sci. Eng. (2023) 1–9.

[44] Max Welling, Yee W. Teh, Bayesian learning via stochastic gradient langevin dynamics, in: Proceedings of the 28th International Conference on Machine Learning, ICML-11, Citeseer, 2011, pp. 681–688.

[45] Arnak Dalalyan, Further and stronger analogy between sampling and optimization: Langevin Monte Carlo and gradient descent, in: Conference on Learning Theory, PMLR, 2017, pp. 678–689.

[46] Arnak S. Dalalyan, Avetik Karagulyan, User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient, Stochastic Process. Appl. 129 (12) (2019) 5278–5311.

[47] Maxim Raginsky, Alexander Rakhlin, Matus Telgarsky, Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis, in: Conference on Learning Theory, PMLR, 2017, pp. 1674–1703.

[48] Haoyang Zheng, Wei Deng, Christian Moya, Guang Lin, Accelerating approximate thompson sampling with underdamped langevin Monte Carlo, 2024, arXiv preprint arXiv:2401.11665.

[49] Anastasios N. Angelopoulos, Stephen Bates, A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2021, arXiv preprint arXiv:2107.07511.

[50] Volodya Vovk, Alexander Gammerman, Craig Saunders, Machine-learning applications of algorithmic randomness, 1999.

[51] Vladimir Vovk, Alexander Gammerman, Glenn Shafer, Algorithmic Learning in a Random World, vol. 29, Springer, 2005.

[52] Glenn Shafer, Vladimir Vovk, A tutorial on conformal prediction, J. Mach. Learn. Res. 9 (3) (2008).

[53] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, Alex Gammerman, Inductive confidence machines for regression, in: Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13, Springer, 2002, pp. 345–356.

[54] Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, Regina Barzilay, Conformal language modeling, 2023, arXiv preprint arXiv:2306.10193.

[55] Kexin Huang, Ying Jin, Emmanuel Candes, Jure Leskovec, Uncertainty quantification over graph with conformalized graph neural networks, Adv. Neural Inf. Process. Syst. 36 (2024).

[56] Anastasios Angelopoulos, Emmanuel Candes, Ryan J. Tibshirani, Conformal PID control for time series prediction, Adv. Neural Inf. Process. Syst. 36 (2024).

[57] Yaniv Romano, Evan Patterson, Emmanuel Candes, Conformalized quantile regression, Adv. Neural Inf. Process. Syst. 32 (2019).

[58] Beichuan Deng, Yeonjong Shin, Lu Lu, Zhongqiang Zhang, George Em Karniadakis, Approximation rates of DeepONets for learning operators arising from advection–diffusion equations, Neural Netw. 153 (2022) 411–426.

[59] Guanxun Li, Guang Lin, Zecheng Zhang, Quan Zhou, Fast replica exchange stochastic gradient langevin dynamics, 2023, arXiv preprint arXiv:2301.01898.

[60] Tianqi Chen, Emily Fox, Carlos Guestrin, Stochastic gradient hamiltonian Monte Carlo, in: International Conference on Machine Learning, PMLR, 2014, pp. 1683–1691.

[61] R.N. Bhattacharya, Criteria for recurrence and existence of invariant measures for multidimensional diffusions, Ann. Probab. (1978) 541–553.

[62] Gareth O. Roberts, Richard L. Tweedie, Exponential convergence of langevin distributions and their discrete approximations, Bernoulli (1996) 341–363.

[63] Roger Koenker, Gilbert Bassett Jr., Regression quantiles, Econometrica (1978) 33–50.

[64] Ingo Steinwart, Andreas Christmann, Estimating conditional quantiles with the help of the pinball loss, Bernoulli 17 (1) (2011) 211–225.

[65] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[66] Guang Lin, Yating Wang, Zecheng Zhang, Multi-variance replica exchange SGMCMC for inverse and forward problems via Bayesian PINN, J. Comput. Phys. 460 (2022) 111173.

[67] Wei Deng, Qi Feng, Liyao Gao, Faming Liang, Guang Lin, Non-convex learning via replica exchange stochastic gradient mcmc, in: International Conference on Machine Learning, PMLR, 2020, pp. 2474–2483.

[68] Amanda A. Howard, Mauro Perego, George E. Karniadakis, Panos Stinis, Multifidelity deep operator networks, 2022, arXiv preprint arXiv:2204.09157.

[69] Amirhossein Mollaali, Izzet Sahin, Iqrar Raza, Christian Moya, Guillermo Paniagua, Guang Lin, A physics-guided bi-fidelity Fourier-featured operator learning framework for predicting time evolution of drag and lift coefficients, Fluids 8 (12) (2023) 323.

[70] Christian Moya, Guang Lin, Fed-deeponet: Stochastic gradient-based federated training of deep operator networks, Algorithms 15 (9) (2022) 325.