Probabilistic neural operators for functional uncertainty quantification

Anonymous authors
Paper under double-blind review

Abstract

Neural operators aim to approximate the solution operator of a system of differential equations purely from data. They have shown immense success in modeling complex dynamical systems across various domains. However, the occurrence of uncertainties inherent in both model and data has so far rarely been taken into account—a critical limitation in complex, chaotic systems such as weather forecasting. In this paper, we introduce the probabilistic neural operator (PNO), a framework for learning probability distributions over the output function space of neural operators. PNO extends neural operators with generative modeling based on strictly proper scoring rules, integrating uncertainty information directly into the training process. We provide a theoretical justification for the approach and demonstrate improved performance in quantifying uncertainty across different domains and with respect to different baselines. Furthermore, PNO requires minimal adjustment to existing architectures, shows improved performance for most probabilistic prediction tasks, and leads to well-calibrated predictive distributions and adequate uncertainty representations even for long dynamical trajectories. Implementing our approach into large-scale models for physical applications can lead to improvements in corresponding uncertainty quantification and extreme event identification, ultimately leading to a deeper understanding of the prediction of such surrogate models.

1 Introduction

Many complex phenomena in the sciences are described via parametric partial differential equations (PDEs), making their study a crucial research topic for systems in biology, physics, or engineering sciences. Since traditional solvers, such as finite element methods are based on a space discretization there exists a tradeoff between solution accuracy and the computational complexity imposed by the discretization resolution. A recent breakthrough has been achieved by neural operators (Kovachki et al., 2023; Lu et al., 2021), which are a class of neural networks that have the capacity to approximate any continuous operator, including any solution operator of a given family of parametric PDEs. Neural operators, and especially Fourier neural operators (Li et al., 2021), are now commonly used in practice and have been applied to problems such as weather forecasting (Pathak et al., 2022), fluid dynamics (Renn et al., 2023), seismic imaging (Li et al., 2023), carbon capture (Wen et al., 2023), or plasma modeling in nuclear fusion (Gopakumar et al., 2024). However, in application scenarios, the entire operator learning process is usually corrupted by different sources of noise. In addition, the model itself can be misspecified or—as in the case of dynamical systems—long-term predictions might be too chaotic to be predictable in a reliable manner. For these cases, it becomes crucial to have a precise understanding of the uncertainty of the corresponding model predictions.

A common approach to uncertainty quantification for standard neural networks is the Laplace approximation (Daxberger et al., 2021), which has been extended to Fourier neural operators by Weber et al. (2024) and Magnani et al. (2024). While this approach is efficient to implement and does not require additional training, it has notable limitations. Specifically, it does not incorporate uncertainty during network training, and its assumption of a Gaussian predictive distribution may be overly restrictive. Guo et al. (2023) introduce an information bottleneck structure that utilizes a decoder with a predictive Gaussian distribution. While innovative, this method shares the limitations of the Gaussian assumption.

Our work addresses these challenges by proposing a theoretically grounded approach to incorporate predictive uncertainty during training in the infinite-dimensional setting of neural operators. By leveraging strictly proper scoring rules, our method enables better-calibrated uncertainty estimates while supporting more flexible predictive distributions.

1.1 Our contributions

In this article, we introduce the *probabilistic neural operator (PNO)*, which extends the neural operator framework to a probabilistic setting, where the output is in the form of a (functional) probability distribution. Our novel approach is based on recent advances in training generative neural networks via proper scoring rules (Pacchiardi et al., 2024). PNO is designed to generate samples from an (empirical) probability distribution and is trained on an objective function that fits the predictive distribution to the corresponding observations, thereby integrating uncertainty directly into the training process. We build upon prior research by introducing a theoretical framework for proper scoring rules in separable Hilbert spaces. Specifically, we prove that the energy score is a strictly proper scoring rule in these spaces and, therefore, leads to desirable properties as a loss function. The resulting predictions of the network are well-calibrated and can be—due to their sample-based character—applied for different downstream tasks, including extreme event analysis and estimation of dependencies. We evaluate PNO on PDE datasets and a real-world data task, namely the prediction of surface temperature across Europe. PNO requires minimal adjustment to existing architectures and leads to well-calibrated predictive distributions and adequate uncertainty representations even for long dynamical trajectories. To summarize, we make the following key contributions:

- We introduce the *probabilistic neural operator (PNO)* as an extension of the neural operator to the probabilistic setting based on proper scoring rules.
- We prove that the *energy score* is a strictly proper scoring rule in separable Hilbert spaces and, therefore, a suitable loss function for the PNO.
- In extensive experiments including 2- and 3-dimensional PDEs, real-world weather data, and different model architectures we show that the PNO setup improves the calibration error while maintaining the accuracy of competing approaches.

1.2 Related work

In order to assess uncertainties in machine learning models, a variety of approaches are available; for a detailed overview compare Abdar et al. (2021). The most common way to perform uncertainty quantification is under the **Bayesian paradigm**. By utilizing Monte Carlo dropout during the inference phase of a neural network, Gal & Ghahramani (2016) show that the corresponding predictive distribution is a Bayesian approximation of a deep Gaussian process. While the approach requires no additional training, the approximation does not hold for all architectures and no uncertainty information is used in the training process. Another method that has been popularized in deep learning recently is the Laplace approximation (Daxberger et al., 2021). It builds on the idea that, in a Bayesian setting, the posterior distribution over the network weights can be approximated by a Gaussian distribution, centered around a local maximum. The advantage is that this local maximum is available by regular training, while the covariance matrix of the Gaussian can be recovered by efficient approximations of the Hessian of the loss function (Botev et al., 2017).

A different class of approaches to uncertainty quantification is given by **ensemble methods**, where the main idea is to obtain an ensemble of predictions that differ slightly from each other and act as an empirical predictive distribution. A common way to utilize this for neural networks is to use so-called deep ensembles (Lakshminarayanan et al., 2017), where the ensemble is created by training several networks separately, with each optimization leading to a different local minimum. Although this has shown to perform well in practice, it is hindered by the computational demands of training multiple neural networks. Another method to obtain ensembles is by perturbing the input data before the forward pass through the model, which leads to an ensemble-based distribution in the output. This approach is used mainly in dynamical systems and weather forecasting (Bi et al., 2023; Bülte et al., 2024). However, this requires additional fine-tuning of the input perturbations and can lead to instabilities in the model predictions.

Finally, another way of uncertainty quantification is provided by **scoring rule**-based approaches. A scoring rule (Gneiting & Raftery, 2007) assigns a numerical score to a distribution-observation pair and can be minimized as a loss function, in order to fit a predictive distribution. Scoring rule approaches have been combined with nonparametric distributional regression (Walz et al., 2024), or neural networks (Rasp & Lerch, 2018), including multivariate cases (Pacchiardi et al., 2024; Chen et al., 2024). However, to the best of our knowledge, scoring rule approaches have not yet been transferred to the infinite-dimensional setting.

Since neural operators map to function spaces, the previously mentioned approaches need to be adjusted for **uncertainty quantification in infinite-dimensional spaces**. Weber et al. (2024) and Magnani et al. (2024) extend the Laplace approximation to Fourier neural operators, by performing the approximation in the last layers of the Fourier domain. While this approach is efficient to implement and requires no additional training, the assumption of a Gaussian can be restrictive, especially when it comes to calibration. Furthermore, the method is only applied post-hoc and no uncertainty information is used in the training process. Guo et al. (2023) propose an uncertainty quantification method that utilizes an information bottleneck and a Gaussian distribution to recover mean and standard deviations in the output space. However, this approach only leads to an estimate of the mean and standard deviation and does not allow for other measures of uncertainty. Ma et al. (2024) propose a quantile neural operator that is based on conformal prediction and provides a risk-controlling prediction set that includes the true solution with a user-specified probability. While this approach leads to a risk control for the error of the neural operator, it does not lead to a predictive distribution that can be used for assessing the full uncertainty in the prediction. In a different line of work, Ma et al. (2024) utilize the deep ensemble method as a baseline, which can be directly extended to the infinite-dimensional case by training multiple neural operators, leading to a functional ensemble prediction. However, usability is limited by the computational overhead of training multiple models independently, especially if a large amount of ensembles is required.

To the best of our knowledge, there are no other methods than those mentioned above that are specifically designed to combine uncertainty quantification with neural operators. Our work overcomes the previously discussed limitations, as it provides a predictive distribution that is directly incorporated into the training process by combining neural operators with scoring rule training.

1.3 Outline

This article is structured as follows: Section 2 provides preliminaries regarding the problem setting and neural operators, while Section 3 introduces neural network training via scoring rule minimization in separable Hilbert spaces. Section 4 is dedicated to the numerical experiments for several benchmark tasks, as well as a real-world application. The article finishes with the conclusion in Section 5.

2 Operator learning

Let $\mathcal{A} = \mathcal{A}(\mathcal{D}, \mathbb{R}^{d_a})$ and $\mathcal{U} = \mathcal{U}(\mathcal{D}; \mathbb{R}^{d_u})$ denote separable Banach spaces of functions over a bounded domain $\mathcal{D} \subset \mathbb{R}^d$ with values in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$, respectively. In this work, we consider parametric partial differential equations of the form:

$$\begin{aligned} \mathcal{L}_a[u](x) &= f(x), \qquad x \in \mathcal{D}, \\ u(x) &= u_b(x), \qquad x \in \partial D, \end{aligned} \tag{1}$$

for some $a \in \mathcal{A}, u \in \mathcal{U}, f \in \mathcal{U}$ and boundary condition $u_b \in \mathcal{U}$. We assume that the solution $u \in \mathcal{U}$ exists and is bounded for every u_b and that $\mathcal{L}_a \in \mathcal{L}(\mathcal{U}, \mathcal{U})$ is a mapping from the parameter space $\mathcal{A}$ to the (possibly unbounded) space of linear operators mapping $\mathcal{U}$ to its dual. This gives rise to the nonlinear solution operator $G^\dagger : (a, u_b) \mapsto u$ mapping the parameters to the corresponding solution. Note that this includes initial condition as well as time-dependent problems, as $\mathcal{D}$ can be chosen as a spatio-temporal domain. We typically assume that we have observations $\{(a_j, u_j)\}_{j=1}^N$, where $a_j \sim \mu$ is generated from some probability measure μ supported on $\mathcal{A}$ and $u_j = \mathcal{G}^\dagger(a_j)$.

2.1 Neural operators

The aim of neural operators (Kovachki et al., 2023) is to generalize deep neural networks to the infinite-dimensional operator setting in order to learn an approximation of $\mathcal{G}^\dagger$ via a parametric map

$$\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}, \quad \theta \in \Theta, \tag{2}$$

with parameters from a finite-dimensional space $\Theta \subseteq \mathbb{R}^p$. By minimizing a suitable loss function one wants to choose $\theta^* \in \Theta$ such that $\mathcal{G}_{\theta^*} \approx \mathcal{G}^\dagger$. Following the framework of Li et al. (2020), neural operators are constructed using a sequence of layers consisting of learnable integral operators combined with pointwise non-linearities. For an input function $v_i \in \mathcal{U}(\mathcal{D}; \mathbb{R}^{d_v})$ at the ith layer, the map $G_\phi : \mathcal{U}(\mathcal{D}; \mathbb{R}^{d_v}) \to \mathcal{U}(\mathcal{D}; \mathbb{R}^{d_v}), v_i \mapsto v_{i+1}$ is computed as

$$G_\phi v_i(x) \coloneqq \sigma\Big(\left(\mathcal{K}(a;\phi) v_i\right)(x) + W_{i,\phi} v_i(x)\Big), \quad \forall x \in \mathcal{D}.$$

Here, $W_{i,\phi} : \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$ is a linear transformation, $\sigma : \mathbb{R} \to \mathbb{R}$ denotes a pointwise nonlinearity, and $\mathcal{K} : \mathcal{A} \times \Theta_{\mathcal{K}} \to \mathcal{L}\left(\mathcal{U}(\mathcal{D}; \mathbb{R}^{d_v}), \mathcal{U}(\mathcal{D}; \mathbb{R}^{d_v})\right)$ is a map to the bounded linear operators on $\mathcal{U}(\mathcal{D}; \mathbb{R}^{d_v})$, which is parameterized by $\phi \in \Theta_{\mathcal{K}} \subseteq \mathbb{R}^p$. Typically, $\mathcal{K}(a;\phi)$ is chosen as a kernel integral transformation that is parameterized by a neural network.

More specifically, in this article we mainly focus on the Fourier neural operator (FNO) (Li et al., 2021), which specifies the integral kernel as a convolution operator defined in Fourier space as

$$G_\phi v_i(x) = \sigma\Big(\mathcal{F}^{-1}(R_{i,\phi} \cdot \mathcal{F}(v_i))(x) + W_{i,\phi} v_i(x)\Big), \quad \forall x \in \mathcal{D},$$

where $\mathcal{F}$ is the Fourier transform of a function $f : \mathcal{D} \to \mathbb{R}^{d_v}$, $\mathcal{F}^{-1}$ its inverse, and $R_\phi = \mathcal{F}(\kappa_\phi)$, where $\kappa_\phi : \mathcal{D} \to \mathbb{R}^{d_v}$ is a periodic function parameterized by neural network parameters $\phi \in \Theta_{\mathcal{K}}$. This means that in each layer i, $R_{i,\phi}$ is directly parameterized in the Fourier domain by a neural network with parameters $\phi \in \Theta_{\mathcal{K}}$. Typically, several of these layers are combined together with additional pointwise lifting and projection operators (Li et al., 2021). Several other variants and extensions of Fourier neural operators have been proposed, such as U-shaped neural operators (Rahman et al., 2023), spherical Fourier neural operators (Bonev et al., 2023) or Wavelet neural operators (Tripura & Chakraborty, 2023).

3 A probabilistic neural operator

In the neural operator setting introduced above, the data pairs can be corrupted by noise and in time-dependent PDEs long prediction horizons might lead to unstable predictions. Moving to a probabilistic setting, instead of mapping directly to the solution, we require the neural operator to output a probability distribution over the function space $\mathcal{U}$. Assume that we have observational data $\{(a_j, u_j)\}_{j=1}^N$ with prior distribution $a_j \sim P_{\mathcal{A}}$ and conditional distribution $u_j \sim P_{\mathcal{U}}^*(\,\cdot \mid a_j\,)$ and let $\mathcal{P}_{\mathcal{U}}$ denote a set of probability measures over $\mathcal{U}$. We consider a probabilistic neural operator to be a map

$$\mathcal{G}_\theta : \mathcal{A} \times \mathcal{Z} \to \mathcal{U}, \quad \theta \in \Theta,$$

that aims to learn a distribution $P_\theta(\,\cdot \mid a_j\,) \coloneqq \mathcal{G}_\theta(a_j, \cdot) \in \mathcal{P}_{\mathcal{U}}$ parameterized by $\theta \in \Theta$ that fulfills $P_\theta(\,\cdot \mid a_j\,) \approx P_{\mathcal{U}}^*(\,\cdot \mid a_j\,)$. Samples from the conditional distribution are generated via $\mathcal{G}_\theta(a, z)$, where $z \sim \mu_{\mathcal{Z}}$ and $\mu_{\mathcal{Z}}$ is a probability distribution over some space $\mathcal{Z}$, usually chosen as a simple tractable distribution such as a Gaussian. For a fixed initial condition a, we denote the approximate (empirical) conditional distribution parameterized by the neural network as $P_\theta^M \coloneqq \{\mathcal{G}_\theta(a, z_m)\}_{m=1}^M$ with $z_m \sim \mu_{\mathcal{Z}}$, $m = 1, ..., M$, where we drop the dependence on the initial condition a from the notation. In this article, we restrict ourselves to data from separable Hilbert spaces, which include most solution spaces of PDEs, such as the Sobolev spaces $H^k, k \in \mathbb{N}$.

3.1 Proper scoring rules for neural operators

Our methodology builds around the theory of proper scoring rules and its extensions to separable Hilbert spaces (Steinwart & Ziegel, 2021; Ziegel et al., 2024). Let $(\mathcal{X}, \mathcal{A})$ be a measurable space and let $\mathcal{M}_1(\mathcal{X})$ denote the class of all probability measures on $\mathcal{X}$. For $\mathcal{P} \subseteq \mathcal{M}_1(\mathcal{X})$, a *scoring rule* is a function $S : \mathcal{P} \times \mathcal{X} \to [-\infty, \infty]$ such that the integral $\int S(P, x) dQ(x)$ exists for all $P, Q \in \mathcal{P}$. Define the so-called *expected score* as $S(Q, P) := \int S(Q, x)\, dP(x) = \mathbb{E}_{X \sim P}[S(Q, X)]$. Then S is called *proper* relative to the class $\mathcal{P}$ if

$$S(P, P) \leq S(Q, P), \quad \text{for all } P, Q \in \mathcal{P}, \tag{3}$$

and it is called *strictly proper* if equality in (3) implies $P = Q$ (Gneiting & Raftery, 2007). In other words, for a strictly proper scoring rule, P and Q coincide if and only if Q is the unique minimizer of the expected score. This allows for efficient and simple training of neural networks via strictly proper scoring rules, as shown later.

In this article, we focus on the so-called *energy score* (Gneiting & Raftery, 2007), which is a commonly used scoring rule for the multivariate finite-dimensional setting that has been successfully used in combination with neural networks (Pacchiardi et al., 2024; Chen et al., 2024). In the following theorem we prove that the energy score can be extended to a strictly proper scoring rule over separable Hilbert spaces.

Theorem 3.1 (Energy score)**.** *Let $\mathcal{H}$ denote a separable Hilbert space and $x \in \mathcal{H}$. The energy score* $\mathrm{ES} : \mathcal{M}_1^E(\mathcal{H}) \times \mathcal{H} \to \mathbb{R}$ *defined as*

$$\mathrm{ES}(P, x) := \mathbb{E}_P[\|X - x\|_{\mathcal{H}}] - \frac{1}{2}\mathbb{E}_P[\|X - X'\|_{\mathcal{H}}], \tag{4}$$

with $X, X' \overset{i.i.d}{\sim} P \in \mathcal{M}_1^E(\mathcal{H}) := \left\{P \in \mathcal{M}_1(\mathcal{H}) \mid \int_{\mathcal{H}} ||x||_{\mathcal{H}} dP(x) < \infty\right\}$ *is strictly proper relative to the class* $\mathcal{M}_1^E(\mathcal{H})$ *.*

Proof. As a first step, we show that a separable Hilbert space induces a positive definite kernel that is characteristic. A measurable and bounded kernel k is called *characteristic* if the kernel embedding defined by $\Phi(P) := \int k(\cdot, \omega) dP(\omega)$, with $P \in \mathcal{M}_1^k(\mathcal{H}) := \left\{P \in \mathcal{M}_1(\mathcal{H}) \mid \int_{\mathcal{H}} \sqrt{k(x, x)} dP(x) < \infty\right\}$, is injective (Steinwart & Ziegel, 2021).

Let $(\mathcal{X}, d)$ be a metric space and define $\mathcal{M}_1^d(\mathcal{X}) := \left\{P \in \mathcal{M}_1(\mathcal{X}) \mid \exists o \in \mathcal{X} \text{ such that } \int d(o, x)\, dP(x) < \infty\right\}$. It is said to be of *negative type* if d is negative definite, i.e. for all $n \in \mathbb{N}$, $x_1, ..., x_n \in \mathcal{X}$ and $\alpha_1, ..., \alpha_n \in \mathbb{R}$ with $\sum_{i=1}^n \alpha_i = 0$ it holds that $\sum_{i,j=1}^n \alpha_i \alpha_j d(x_i, x_j) \leq 0$ (compare Berg et al., 1984, Definition 3.1.1). Furthermore, let $P_1, P_2 \in \mathcal{M}_1^d(\mathcal{X})$. Then, following Lyons (2013), if $\mathcal{X}$ has negative type, it holds that

$$\int d(x_1, x_2)\, d(P_1 - P_2)^2(x_1, x_2) \leq 0. \tag{5}$$

Furthermore, the space $(\mathcal{X}, d)$ is said to be of *strong negative type* if it is of negative type and equality in (5) holds if and only if $P_1 = P_2$, which is the case for separable Hilbert spaces (Lyons, 2013, Theorem 3.25). Furthermore, a metric space $(\mathcal{X}, d)$ of negative type induces a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ given as $k(x, y) = d(x, z_0) + d(y, z_0) - d(x, y)$ for some fixed $z_0 \in \mathcal{X}$. Finally, Sejdinovic et al. (2013, Proposition 29) state that for a metric space of strong negative type the corresponding induced kernel k is characteristic.

Next, we cite Steinwart & Ziegel (2021, Theorem 1.1), who prove that the *kernel score* $S_k : \mathcal{M}_1^k \times \mathcal{X} \to \mathbb{R}$, defined as $S_k(P, x) := \frac{1}{2}\mathbb{E}_P[k(X, X')] - \mathbb{E}_P[k(X, x)] + \frac{1}{2}k(x, x)$, where $x \in \mathcal{X}$ and $X, X' \overset{i.i.d}{\sim} P \in \mathcal{M}_1^k$, is a strictly proper scoring rule if k is a characteristic kernel. Therefore, it only remains to show that the energy score corresponds to the kernel score S_{k_d} given by the induced kernel $k_d := d(x, z_0) + d(y, z_0) - d(x, y)$ in the Hilbert space $\mathcal{H}$ with metric $d(x, x') := \|x - x'\|_{\mathcal{H}}$ and for some fixed $z_0 \in \mathcal{H}$. We obtain

$$\begin{aligned}
S_{k_d}(P, x) &= -\mathbb{E}_P[k_d(X, x)] + \frac{1}{2}\mathbb{E}_P[k_d(X, X')] + \frac{1}{2}k_d(x, x) \\
&= -\mathbb{E}_P[d(X, z_0) + d(x, z_0) - d(X, x)] + \frac{1}{2}\mathbb{E}_P[d(X, z_0) + d(X', z_0) - d(X, X')] \\
&+ \frac{1}{2}(d(x, z_0) + d(x, z_0) - d(x, x)) \\
&= \mathbb{E}_P[d(X, x)] - \frac{1}{2}\mathbb{E}_P[d(X, X')] - \frac{1}{2}\mathbb{E}_P[d(X, z_0)] + \frac{1}{2}\mathbb{E}_P[d(X', z_0)] - \mathbb{E}_P[d(x, z_0)] + d(x, z_0) \\
&= \mathbb{E}_P[d(X, x)] - \frac{1}{2}\mathbb{E}_P[d(X, X')] = \mathrm{ES}(P, x).
\end{aligned}$$

Since $\mathcal{H}$ is of strong negative type, k_d is characteristic and, thus, by Steinwart & Ziegel (2021), the energy score is strictly proper relative to the class $\mathcal{M}_1^{k_d}(\mathcal{H})$. $\square$

3.2 Training neural operators via the energy score

After introducing the underlying concepts and showing that the energy score is strictly proper in separable Hilbert spaces, we now introduce the corresponding training regime for neural networks. This follows the general framework of Pacchiardi et al. (2024), adapted to operator learning. Given a strictly proper scoring rule, the minimization objective is given as

$$\operatorname*{argmin}_{\theta} \mathbb{E}_{a \sim P_{\mathcal{A}}} \mathbb{E}_{u \sim P^*_{\mathcal{U}}(\,\cdot \mid a\,)} S(P_\theta(\,\cdot \mid a\,), u) \tag{6}$$

and leads to $P^*_\theta(\,\cdot \mid a\,) = P^*_{\mathcal{U}}(\,\cdot \mid a\,)$ almost everywhere (Pacchiardi et al., 2024). In the finite data setting, the objective can be approximated with a Monte-Carlo estimator. While closed-form expressions of S are not always available, the energy score has a representation that admits an unbiased estimator, which requires the output from our neural network to consist of multiple samples of the predictive distribution, i.e., $M > 1$.[1] The minimization objective for the neural network with the energy score is then given by

$$\operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{M} \sum_{j=1}^{M} \|u_i^j - u_i\|_{\mathcal{H}} - \frac{1}{2M(M-1)} \sum_{\substack{j,h=1 \\ j \neq h}}^{M} \|u_i^j - u_i^h\|_{\mathcal{H}} \right). \tag{7}$$

Different methods are available to specify a neural network to output a conditional (empirical) distribution. Here, we focus on two main approaches. One way to obtain samples is to utilize the reparametrization trick (Kingma et al., 2015), which learns a pointwise normal distribution in the last layer of the network, by including two projection layers corresponding to the mean and standard deviation of a normal distribution. Although this has a computational advantage as only one forward pass is required, the predictive distribution is fixed to a Gaussian, which might not be expressive enough for every task. A different way to obtain samples is via stochastic dropout (Gal & Ghahramani, 2016). By adding Bernoulli or Normal random samples to specific weights of the network, a forward pass becomes stochastic and, therefore, leads to a predictive distribution. While this approach has higher computational demands due to the need to perform multiple forward passes, the partial stochasticity of the network leads to a higher expressivity with regard to the conditional distribution (Sharma et al., 2023). We add additional dropout in the Fourier domain, setting random frequency parts as zero with the pre-specified dropout probability, making use of the global structure of the Fourier domain. Since both approaches are quite different and come with different strengths and weaknesses, we included them both in the experiments.

4 Numerical experiments

This section describes our experimental setup. In Subsection 4.1 we introduce the baseline uncertainty quantification methods for comparisons, while Subsection 4.2 includes the evaluation metrics. In the remaining subsections we present the different experiments. All experiments are evaluated on a previously unseen test set with predictions aggregated over ten training runs with different random seeds. The different datasets as well as the corresponding grids are normalized before they are used as input to the network. As architectures, for most experiments, we focus on the Fourier Neural Operator (FNO) (Li et al., 2021) and U-shaped Neural Operator (UNO) (Rahman et al., 2023) architecture. For the spherical shallow water equation we employ the spherical FNO (Bonev et al., 2023). The training process and the corresponding hyperparameters are explained in detail in Appendix A. Although all uncertainty quantification methods use the same underlying architecture in the experiments, the dropout rate has been tuned separately, as it has a significant impact on the shape of the predictive distribution for the different methods. While we keep the dropout rate in the Fourier layers and the standard layers equal during the experiments in section 4, we assess the effect of both changes independently in Appendix B. In order to evaluate our approach beyond the pure performance, we include an additional runtime analysis of the different methods in Appendix C. In Appendix D we assess the influence of the number of samples M during training for PNO_D and PNO_R.

---

[1] Note that this setting still includes the classical neural operator as the special case, where the predictive distribution is a Dirac measure. The objective function then becomes the $\|\cdot\|_{\mathcal{H}}$ loss.

4.1 Baseline models

We compare our method to different baselines, adapted to the task of operator learning. Gal & Ghahramani (2016) show that a neural network with dropout before each layer is mathematically equivalent to a variational approximation of a Gaussian process. This leads to a simple and efficient way of creating a predictive distribution, known as MCDropout (MCD). In our setting, the predictive distribution for a fixed input a is given as

$$P_\theta^M = \{\mathcal{G}_\theta^*(a, \boldsymbol{z}_m)\}_{m=1}^M, \tag{8}$$

where $\boldsymbol{z}_m$ is the random dropout mask and $\mathcal{G}_\theta^*$ denotes the trained neural operator. As opposed to the PNO, the MCDropout model is trained on the L^p loss and no uncertainty information is used in the training process.

Weber et al. (2024) propose to utilize the Laplace approximation (LA) for FNOs, which is based on building a second-order approximation of the weights around the maximum a posteriori (MAP) estimate θ_MAP. By assuming a Gaussian weight prior, the weight-space uncertainty of the LA is given by

$$p(\theta, \mathcal{C}) \approx \mathcal{N}(\theta; \theta_\text{MAP}, \Sigma), \quad \Sigma \coloneqq -(\nabla_\theta^2 \mathcal{L}(\mathcal{C}; \theta)|_{\theta_\text{MAP}})^{-1}, \tag{9}$$

where $\mathcal{C} = \{(a_n, u_n)\}_{n=1}^N$. By sampling from this Gaussian we obtain the predictive distribution P_θ^M. We utilize a last-layer Laplace approximation, which leads to an analytically available Gaussian predictive distribution in function space. Since the Laplace approximation is built around the MAP estimate, it also does not utilize uncertainty information during the training process.

4.2 Evaluation

In order to thoroughly assess the performance of the prediction methods, we use the following evaluation metrics, each focusing on a different aspect of the probabilistic prediction.

$$\mathcal{L}_{L^2}(P_\theta^M, u) \coloneqq \|\overline{u}_M - u\|_{L^2}, \tag{10}$$

$$\text{ES}(P_\theta^M, u) \coloneqq \frac{1}{M} \sum_{m=1}^M \|\hat{u}^m - u\|_{L^2} - \frac{1}{2M(M-1)} \sum_{\substack{m,h=1 \\ m \neq h}}^M \|\hat{u}^m - \hat{u}^h\|_{L^2}, \tag{11}$$

$$\text{CRPS}(P_\theta^M, u) \coloneqq \frac{1}{|\mathcal{D}|} \sum_{(x,t) \in \mathcal{D}} \left(\int_0^1 \text{QS}_\alpha \left(\hat{F}_{x,t}^{-1}(\alpha), y \right) d\alpha \right), \tag{12}$$

$$\text{NLL}(P_\theta^M, u) \coloneqq \frac{1}{2|\mathcal{D}|} \sum_{(x,t) \in \mathcal{D}} \log\left(2\pi\hat{\sigma}^2(x,t)\right) + \frac{\left(\overline{u}_M(x,t) - u(x,t)\right)^2}{\hat{\sigma}^2(x,t)}, \tag{13}$$

$$\mathcal{C}_\alpha(P_\theta^M, u) \coloneqq \frac{1}{|\mathcal{D}|} \sum_{(x,t) \in \mathcal{D}} \mathbb{1} \left\{ u(x,t) \in [\hat{q}_\theta^{\alpha/2}(x,t), \hat{q}_\theta^{1-\alpha/2}(x,t)] \right\}, \tag{14}$$

$$\left|\mathcal{C}_\alpha(P_\theta^M, u)\right| \coloneqq \frac{1}{|\mathcal{D}|} \sum_{(x,t) \in \mathcal{D}} \left| \hat{q}_\theta^{1-\alpha/2}(x,t) - \hat{q}_\theta^{\alpha/2}(x,t) \right|. \tag{15}$$

Here, $\mathcal{D}$ denotes the spatio-temporal domain of the governing equation. The $\mathcal{L}_{L^2}$ loss evaluates the match between the mean of the predictive distribution $\overline{u}_M$ and the observation, while the energy score (ES) evaluates the match for the predictive distribution as a whole. The continuous ranked probability score (CRPS) (Gneiting & Raftery, 2007) evaluates the predictive distribution at a pointwise level and can be interpreted as an integral of all quantile levels over the quantile score $QS_\alpha(q_\alpha, y) \coloneqq 2(\alpha - \mathbb{1}\{y < q_\alpha\})(y - q_\alpha)$, with $\hat{F}_{x,t}$

Model	**Method**	$\mathcal{L}_{L^2}$	ES	CRPS	NLL	$\mathcal{C}_{0.05}$	$\lvert\mathcal{C}_{0.05}\rvert$
FNO	PNO$_\mathrm{D}$	**0.0772** (±0.0015)	**0.0569** (±0.0009)	**0.0462** (±0.0007)	**-1.3444** (±0.0355)	**0.9813** (±0.0032)	**0.3682** (±0.0185)
	PNO$_\mathrm{R}$	0.1031 (±0.0069)	0.0817 (±0.0056)	0.0707 (±0.0040)	685.15 (±913.21)	0.4702 (±0.0559)	0.1249 (±0.0191)
	MCD	0.1002 (±0.0088)	0.0830 (±0.0080)	0.0672 (±0.0066)	1.5973 (±0.8223)	0.5187 (±0.0608)	0.1203 (±0.0031)
	LA	0.0985 (±0.0074)	0.0767 (±0.0040)	0.0621 (±0.0033)	-0.8851 (±0.0531)	**0.9762** (±0.0085)	0.5713 (±0.0076)
UNO	PNO$_\mathrm{D}$	0.1894 (±0.0066)	**0.1354** (±0.0050)	**0.1097** (±0.0041)	**-0.2856** (±0.0614)	0.9069 (±0.0243)	0.6998 (±0.0467)
	PNO$_\mathrm{R}$	0.2005 (±0.0060)	0.1473 (±0.0055)	0.1275 (±0.0044)	20.018 (±11.926)	0.5765 (±0.0698)	0.4218 (±0.0559)
	MCD	**0.1875** (±0.0058)	0.1491 (±0.0076)	0.1233 (±0.0069)	2.2782 (±1.4980)	0.5468 (±0.0600)	0.2822 (±0.0259)
	LA	0.1954 (±0.0074)	0.1530 (±0.0052)	0.1229 (±0.0040)	0.9996 (±0.2835)	0.7477 (±0.0411)	0.4374 (±0.0643)

Table 1: Results for the Darcy Flow equation. The best model is highlighted in bold and the standard deviation is given in the brackets.

denoting the marginal empirical distribution of P_θ^M. The CRPS therefore assesses whether the predicted uncertainty fits the observations at all quantile levels at each gridpoint (x, t), i.e., whether the predictive distribution is well-calibrated. Furthermore, we analyze the negative log-likelihood (NLL) of a predictive pointwise Gaussian distribution $\mathcal{N}(\overline{u}_M(x,t), \hat{\sigma}^2(x,t))$. While not all methods produce Gaussian output, this criterion is commonly used and describes the fit of the predictive distribution in terms of the first two estimated moments. The NLL is also a strictly proper scoring rule (Gneiting & Raftery, 2007). Finally, we report the coverage $\mathcal{C}_\alpha$ of the predictive distribution for selected α-quantile levels, where $q_\theta^\alpha(x)$ denotes the empirical α-quantile of P_θ at (x, t). If the correct distribution is learned, then $\mathcal{C}_\alpha(x) = 1 - \alpha$, however in notions of risk control $\mathcal{C}_\alpha(x) \geq 1 - \alpha$ is usually sufficient. In order to compare methods that achieve similar coverage, we also report the width of the corresponding prediction interval $|\mathcal{C}_\alpha|$. To make the baselines comparable, $M = 100$ predictive samples are drawn from the respective predictive distributions at test time and all metrics are averaged over the test dataset.

4.3 Darcy flow

First, we consider a steady-state solution of the two-dimensional Darcy flow over the unit square, which is defined by a second-order linear elliptic equation of the form

$$\begin{aligned} -\nabla(a(x)\nabla u(x)) &= f(x), && x \in (0,1)^2, \\ u(x) &= 0, && x \in \partial(0,1)^2. \end{aligned}$$

Here, $a \in L^\infty((0,1)^2; \mathbb{R}_+)$ is the diffusion coefficient, $u \in H_0^1((0,1)^2, \mathbb{R})$ is the velocity function, and $f \in L^2((0,1)^2; \mathbb{R})$ is the forcing function. The aim is to learn the solution operator $G^* : L^\infty((0,1)^2; \mathbb{R}_+) \to H_0^1((0,1)^2; \mathbb{R})$, $a \mapsto u$. The PDE is usually used for modeling fluid flow in a porous medium. We use data from Takamoto et al. (2022), with the forcing function fixed as $f(x) = 1$, which corresponds to the setting in Li et al. (2021). The provided data consists of 10000 samples with a resolution of 128×128. We train the networks on a spatial resolution of 64×64 and evaluate on 128×128. The results can be found in Table 1. The PNO$_\mathrm{D}$ obtains the best scores across all metrics when using the FNO architecture. While both the PNO$_\mathrm{D}$ and the Laplace approach achieve the coverage goal, the prediction interval of the PNO$_\mathrm{D}$ is significantly narrower despite the coverage of PNO$_\mathrm{D}$ being better as well. For the UNO architecture, the performance seems to be a bit worse in general, with the PNO$_\mathrm{D}$ still performing best on the distributional metrics. A visualization of the different predictions can be found in Appendix E.

4.4 Kuramoto-Sivashinsky equation

Model	Method	$\mathcal{L}_{L^2}$	ES	CRPS	NLL	$\mathcal{C}_{0.05}$	$\lvert\mathcal{C}_{0.05}\rvert$
FNO	PNO_D	0.8793 (±0.0072)	0.6195 (±0.0049)	0.5496 (±0.0097)	3.0389 (±0.5872)	0.7640 (±0.0191)	3.0852 (±0.0422)
	PNO_R	0.8640 (±0.0038)	**0.6081** (±0.0027)	**0.4743** (±0.0037)	**1.2268** (±0.0083)	0.9401 (±0.0030)	3.2594 (±0.0294)
	MCD	**0.8635** (±0.0048)	0.7541 (±0.0049)	0.5974 (±0.0058)	107.10 (±22.941)	0.3600 (±0.0082)	0.6046 (±0.0164)
	LA	0.8772 (±0.0055)	0.7332 (±0.0103)	0.6081 (±0.0077)	43.9302 (±10.8764)	0.3955 (±0.0189)	0.8175 (±0.0669)
UNO	PNO_D	**0.9315** (±0.0046)	**0.6560** (±0.0032)	0.6014 (±0.0088)	**2.0875** (±0.2564)	0.7828 (±0.0315)	3.3892 (±0.1214)
	PNO_R	0.9371 (±0.0022)	0.6595 (±0.0015)	**0.5326** (±0.0023)	5.4528 (±6.2975)	0.9110 (±0.0150)	3.4039 (±0.0442)
	MCD	0.9444 (±0.0014)	0.8500 (±0.0034)	0.7163 (±0.0026)	449.08 (±42.839)	0.1965 (±0.0050)	0.4052 (±0.0109)
	LA	0.9494 (±0.0250)	0.8544 (±0.0265)	0.6928 (±0.0249)	54.338 (±37.019)	0.2560 (±0.0473)	0.5770 (±0.1251)

Table 2: Results for the Kuramoto-Sivashinsky equation. The best model is highlighted in bold and the standard deviation is given in the brackets.

The Kuramoto-Sivashinsky (KS-)equation is a fourth-order parabolic PDE with the following form in one spatial dimension:

$$
\begin{aligned}
\partial_t u(x,t) + u\partial_x u(x,t) + \partial_x^2 u(x,t) + \partial_x^4 u(x,t) &= 0, \qquad && x \in \mathcal{D}, t \in (0,T] \\
u(x,0) &= u_0(x), && x \in \mathcal{D}
\end{aligned}
$$

where $\mathcal{D} \subseteq \mathbb{R}$, $u \in C((0,T]; H^4_{\text{per}}(\mathcal{D};\mathbb{R}))$, and $u_0 \in L^2_{\text{per}}(\mathcal{D};\mathbb{R})$[2]. Furthermore, u_0 is L-periodic, where L is the size of the specific domain. The PDE exhibits strong temporal chaotic behavior and can be seen as a model of Galilean-invariant chaos. The PDE is used for modeling instabilities in a laminar flame front, reaction-diffusion systems, or other chemical reactions. We simulate the KS-equation from random uniform noise $\mathcal{U}(-1,1)$ on a periodic domain $\mathcal{D} = [0, 100]$ using the py-pde package (Zwicker, 2020). We generate 10000 samples with a resolution of 256×300 and train a 2D (spatio-temporal) FNO and UNO, which take the first twenty timesteps as input and solve the equation for the next twenty timesteps with step size $\Delta t = 2$. The results for the KS-equation can be found in Table 2. For the FNO architecture, the MCDropout approach leads to the best L^2 loss, while the PNO_R leads to the best distributional metrics. While no method achieves a coverage of 95%, the PNO_R comes closest. For the UNO architecture, the results differ, with the PNO_D obtaining the best score on most of the metrics. Again the PNO_R leads to the best coverage. A corresponding visualization can be found in Figure 1.

4.5 Spherical shallow water equation

Bonev et al. (2023) have extended the FNO to convolutions on the sphere using spherical harmonic basis functions, leading to the so-called Spherical Fourier Neural Operator (SFNO). Following their experiments, we test our method on the shallow water equation on the rotating sphere. The shallow water equations (SWE) are a set of hyperbolic partial differential equations, derived from the general Navier-Stokes equations and present a framework for modeling free-surface flow problems. In the spherical setting, these are given as

$$
\begin{aligned}
\partial_t h(x,t) + \nabla h(x,t)\boldsymbol{u} &= 0, \qquad && x \in S^2, t \in (0,T] \\
\partial_t h(x,t)\boldsymbol{u} + \nabla\left((\boldsymbol{u} \odot \boldsymbol{u})h(x,t)\right) &= S, && x \in S^2, t \in (0,T],
\end{aligned}
$$

---

[2] $H^k_{\text{per}}(\mathcal{D};\mathbb{R}), L^2_{\text{per}}(\mathcal{D};\mathbb{R})$ denote the periodic Sobolev and L^2 spaces, respectively.

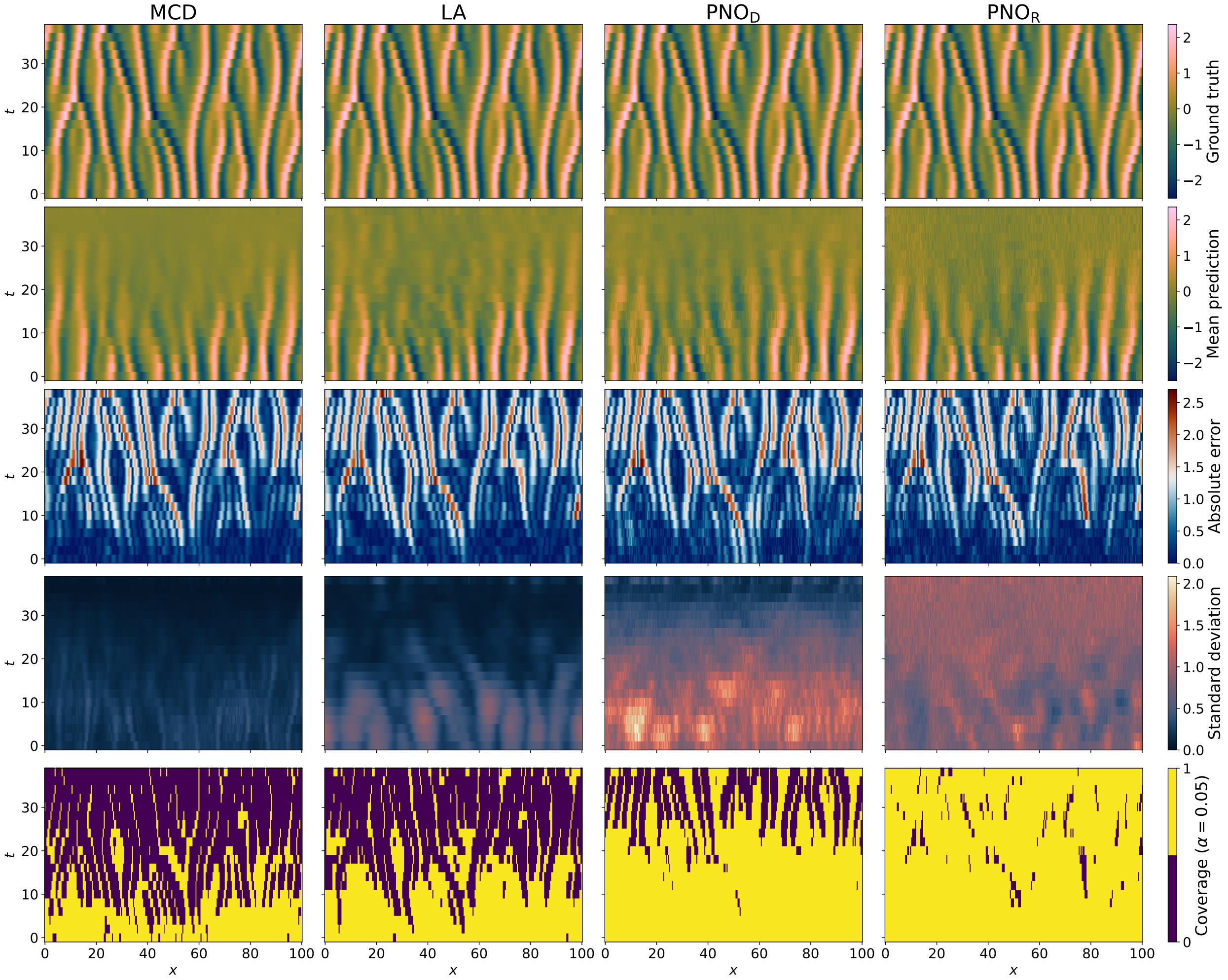

Figure 1: The figure shows the ground truth, mean prediction, absolute error, standard deviation, and coverage for the different methods on a sample of the Kuramoto-Sivashinsky equation.

with the vector of directional velocities $\boldsymbol{u} = (u, v)^\top$. Here, $h \in C((0, T]; H_0^1(S^2; \mathbb{R}))$ is the geopotential of the water and S contains flux terms, such as the Coriolis force. Similarly to Bonev et al. (2023), we simulate 5000 training samples with resolution 128×256 with PDE parameters that model the Earth and an additional 500 evaluation samples. Gaussian random fields are used as initial conditions with a burn-in period of three hours for the numerical solver. The model is trained and evaluated by using the corresponding losses in a weighted version, adjusted to the latitude weighting of the sphere. Furthermore, Bonev et al. (2023) propose a fine-tuning step, where the model is trained with an additional autoregressive step, which leads to more stable results for long rollout times, which we additionally include in our experiments. For evaluation, we report both training versions with one-hour and ten-hour prediction horizons each. The results for the two-step autoregressive training can be found in Table 3 and the results for the single-step training in Table 9 in Appendix F. For the 1 hour prediction, the PNO$_\mathrm{D}$ obtains the best metrics across all models, including a calibrated coverage. For the ten-hour prediction task, the PNO$_\mathrm{R}$ seems to be better for a few metrics. It is notable that the PNO$_\mathrm{R}$ leads to extremely large values for the negative log-likelihood, while still being competitive in other metrics. This might be due to overconfident pointwise predictions, which lead to a small variance and, therefore, to a large NLL. This behavior is also visible in the visualization in Figure 2. The MCD approach does not lead to stable predictions for the ten-hour prediction task, as is reflected in all metrics. However, for the single-step training, the performance is much better (compare Table 9), indicating that the method cannot be combined with the autoregressive training approach.

Δt	**Method**	$\mathcal{L}_{L^2}$	ES	CRPS	NLL	$\mathcal{C}_{0.05}$	$\lvert\mathcal{C}_{0.05}\rvert$
1h	PNO_D	**0.2970** (±0.0232)	**0.2216** (±0.0142)	**0.0903** (±0.0056)	**-0.4561** (±0.0447)	**0.9828** (±0.0031)	0.8959 (±0.0440)
	PNO_R	0.3871 (±0.0126)	0.2740 (±0.0088)	0.1561 (±0.0050)	2.3×10^{10} ($\pm1.6\times10^{9}$)	0.0274 (±0.0206)	0.0844 (±0.0551)
	MCD	0.3022 (±0.0204)	0.2368 (±0.0196)	0.0944 (±0.0066)	0.8878 (±0.2608)	0.6023 (±0.0250)	0.2550 (±0.0072)
	LA	0.4262 (±0.0522)	0.3159 (±0.0299)	0.1400 (±0.0120)	0.2374 (±0.2229)	0.9096 (±0.0762)	1.0866 (±0.3472)
10h	PNO_D	1.2421 (±0.0650)	0.9915 (±0.0535)	**0.4330** (±0.0215)	**5.4385** (±0.5448)	0.4935 (±0.0200)	0.9210 (±0.0793)
	PNO_R	**1.1803** (±0.0345)	**0.9535** (±0.0413)	0.4995 (±0.0130)	2.1×10^{11} ($\pm1.4\times10^{10}$)	0.0212 (±0.0200)	0.0850 (±0.0546)
	MCD	9.1×10^{5} ($\pm2.7\times10^{6}$)	7.5×10^{5} ($\pm2.2\times10^{6}$)	2.8×10^{5} ($\pm8.3\times10^{5}$)	22.292 (±19.623)	0.3530 (±0.1364)	6.7×10^{5} ($\pm2.0\times10^{6}$)
	LA	1.5405 (±0.3697)	1.2612 (±0.3338)	0.5299 (±0.1336)	6.4958 (±5.5154)	0.5759 (±0.1480)	1.1925 (±0.4394)

Table 3: Results for the spherical shallow water equation with two-step autoregressive training. The best model is highlighted in bold and the standard deviation is given in the brackets.

4.6 Forecasting of 2m surface temperature

The experiments so far have focused on systems in which the underlying equations are known and computationally tractable. We now want to demonstrate our method on a more complex large-scale dynamical system with real data. More concretely, we use our approach to create probabilistic medium-range forecasts of the Earth's surface temperature. Recently, neural networks have shown huge success in medium-range weather prediction tasks, with models such as FourCastNet (Pathak et al., 2022), or PanguWeather (Bi et al., 2023). However, while some approaches are available that directly learn probabilistic predictions (Price et al., 2024), most models are either deterministic or require an additional post-processing step (Bülte et al., 2024).

Here, we want to demonstrate the performance of our model in predicting the 2-meter surface temperature across Europe. For computational reasons we restrict the prediction task to only one input variable. We use the ERA5 dataset (Hersbach et al., 2020), provided via the benchmark dataset WeatherBench (Rasp et al., 2024). The data has a spatial resolution of $0.25^\circ \times 0.25^\circ$ and a time resolution of $6h$. For computational reasons, we restrict ourselves to a spatial domain over Europe. We use a 3D spatio-temporal neural operator that takes the last 10 timesteps (60 hours) as the input and predicts the next 10 timesteps. Therefore, we directly predict several future timesteps, instead of using an autoregressive model. The results of the temperature prediction task can be found in Table 4. All models obtain a fairly similar and consistent L^2 loss, whereas PNO_D exceeds the alternative methods in the remaining metrics. However, none of the methods obtain an optimal coverage. It is noticeable that the PNO_R leads to a very poor NLL score, while the PNO_D outperforms the other methods significantly with regard to that metric. Figure 3 shows the CRPS and coverage across the spatial domain, aggregated over the test data. The CRPS shows a clear performance distinction between land and sea, which is supported by the findings in Bülte et al. (2024). While for MCDropout and the PNO_R, the coverage is clearly different between land and sea, for Laplace and PNO_D this difference is negligible. The latter is preferable, as the models should provide adequate uncertainty even if the underlying spatial domain makes predictions more difficult. Figure 4 shows an additional visualization of the corresponding model predictions.

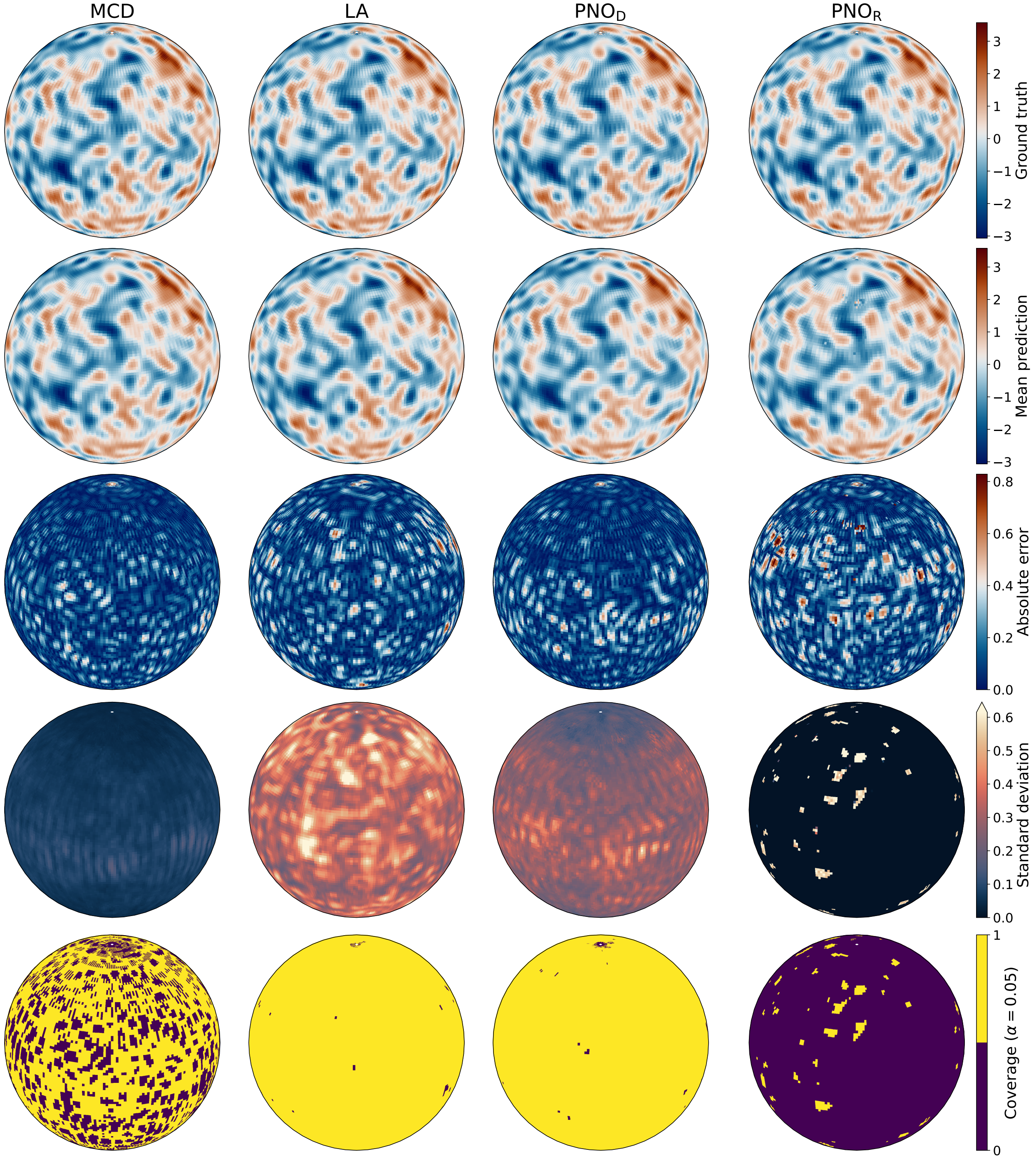

Figure 2: The figure shows the ground truth, mean prediction, absolute error, standard deviation, and coverage for the different methods on a sample of the spherical shallow water equation. The shown variable is the water geopotential and the prediction horizon is one hour with predictions obtained via two-step autoregressive training.

Model	**Method**	$\mathcal{L}_{L^2}$	ES	CRPS	NLL	$\mathcal{C}_{0.05}$	$\lvert\mathcal{C}_{0.05}\rvert$
FNO	PNO$_\text{D}$	0.0242 (±0.0000)	**0.0174** (±0.0000)	**0.0187** (±0.0001)	**-1.9587** (±0.0111)	0.8623 (±0.0033)	0.0975 (±0.0012)
	PNO$_\text{R}$	0.0241 (±0.0001)	0.0187 (±0.0001)	0.0204 (±0.0004)	616.98 ($\pm 1.2\times10^3$)	0.6536 (±0.0222)	0.0576 (±0.0022)
	MCD	**0.0240** (±0.0001)	0.0202 (±0.0000)	0.0214 (±0.0001)	2.3623 (±0.0237)	0.4456 (±0.0015)	0.0351 (±0.0001)
	LA	0.0251 (±0.0001)	0.0195 (±0.0006)	0.0210 (±0.0005)	-0.1502 (±1.0099)	0.6956 (±0.1169)	0.0802 (±0.0344)

Table 4: Results for the ERA5 surface temperature prediction task. The best model is highlighted in bold and the standard deviation is given in the brackets.

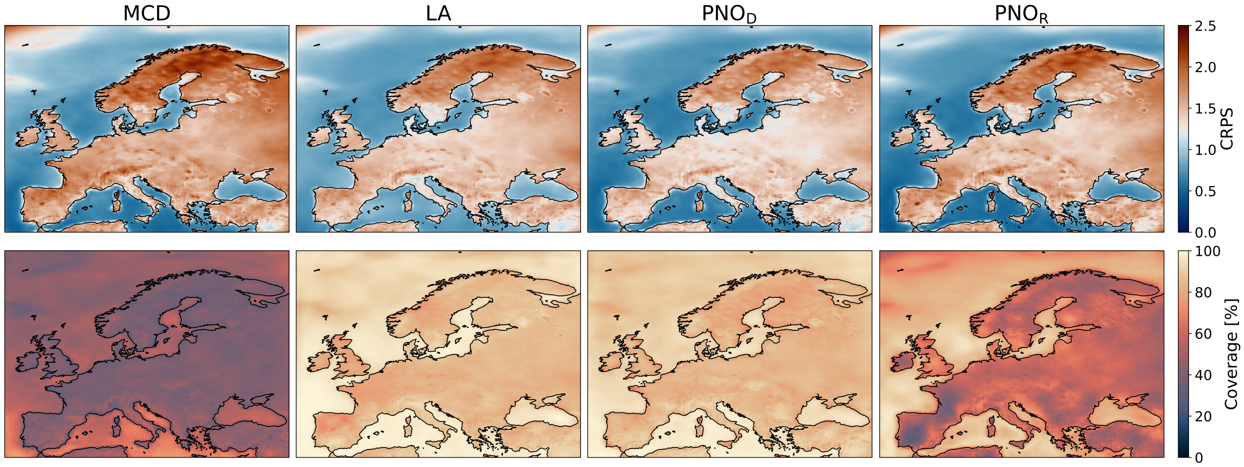

Figure 3: The figure shows the spatial distribution of the CRPS and coverage, aggregated over the test samples for the different methods on the two meter surface temperature prediction task with a prediction horizon of 24h.

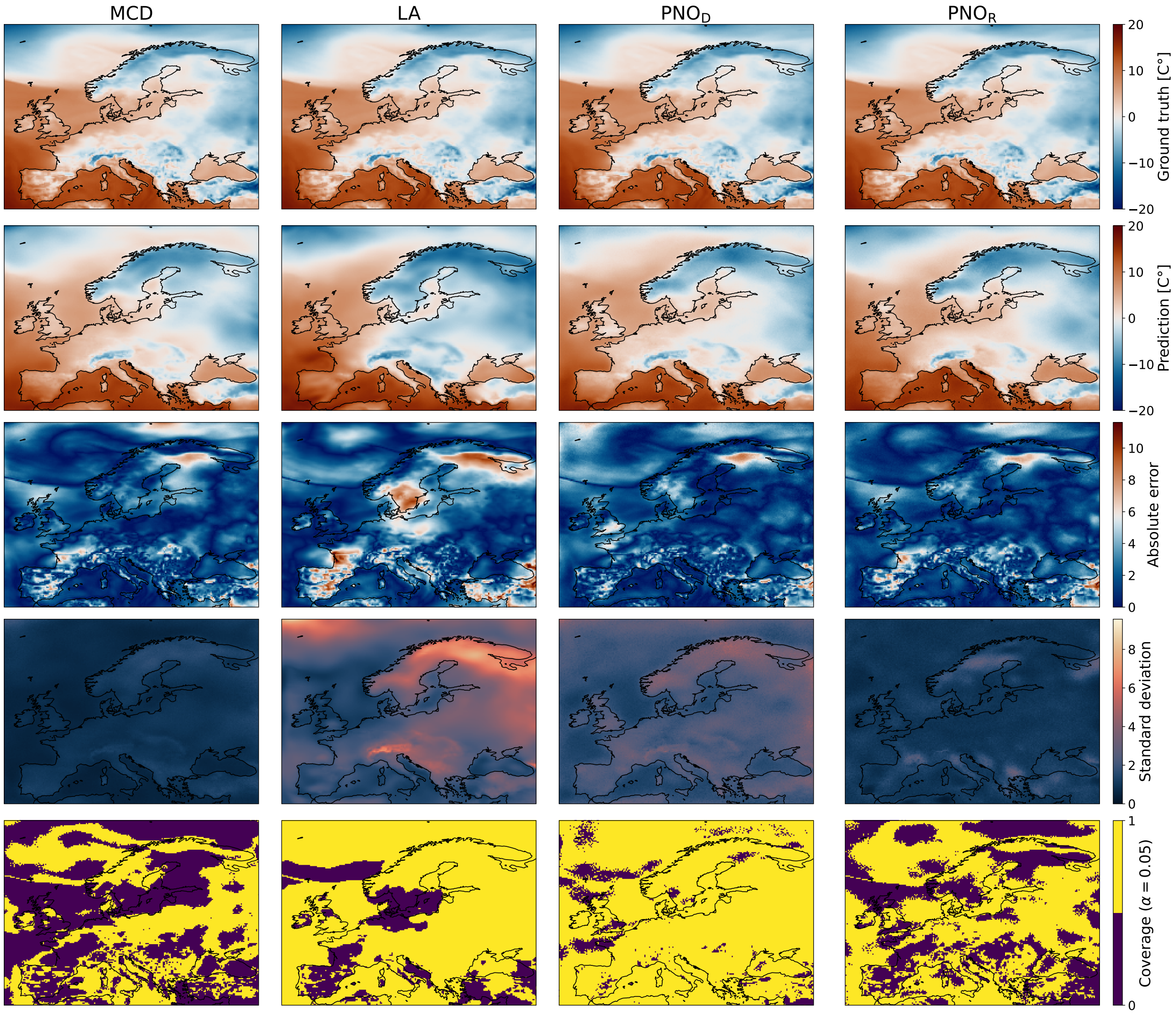
MCD
LA
PNO_D
PNO_R
Ground truth [C°]
20
10
0
−10
−20
Prediction [C°]
Absolute error
0
2
4
6
8
10
Standard deviation
Coverage (α = 0.05)
1

Figure 4: The figure shows the ground truth, mean prediction, absolute error, standard deviation, and coverage for the different methods on a sample of ERA5 dataset. The prediction horizon is 60 hours.

5 Discussion

In this work, we introduce PNO, a framework for learning probability distributions over the output function space of neural operators. Our approach combines neural operators with generative modeling utilizing strictly proper scoring rules. We provide a theoretical justification of the approach and analyze how scoring rules can be extended to separable Hilbert spaces. More specifically, we demonstrate how the energy score can be utilized to create a predictive distribution in the output function space of a neural operator. We present and examine different ways of generating a probability distribution using neural operators that are agnostic to the underlying architecture. Our framework enables more accurate uncertainty quantification while maintaining consistent performance in the respective mean prediction task. These advantages are especially noticeable when the underlying system is highly uncertain or chaotic.

We evaluate our approach across several experiments, including stationary PDE solutions, dynamical systems, and weather forecasting. The experiments demonstrate that PNO is transferrable to different network architectures, as well as different domains. Furthermore, PNO performs well in spatio-temporal as well as autoregressive settings. Compared to the baseline methods, PNO shows the best performance across a variety of metrics, especially with regard to the fit between the predictive distribution and the observations. In addition, the predictions are better calibrated in terms of the coverage of corresponding $1 - \alpha$ prediction intervals.

5.1 Limitations

The main limitation of the PNO is the increased computational complexity and memory requirements. These are mainly induced by the energy score, as it requires multiple samples to evaluate. In the case of the PNO_D this also requires multiple forward passes of the underlying model. A runtime analysis shows that the training time per epoch of the PNO is around 1.5 - 3 times higher than for the comparing methods. Another limiting factor of PNO is that it inherits the expressivity and (lack of) understanding of the underlying generative model. While PNO_R corresponds to a predictive Gaussian in output space, for PNO_D the exact distribution and its properties are not available. A better understanding of the predictive distribution could allow for convergence or coverage guarantees. Finally, the results show that PNO leads to unstable predictions for autoregressive inference with long rollout times.

5.2 Future work

One of the most important tasks for future research revolves around reducing the computational complexity of our method. As a first step, analyzing the capacities of training a deterministic model and fine-tuning the PNO afterwards appears to be a promising direction. In a similar manner, the amount of generated samples could be adapted throughout the training process, increasing performance while reducing training time simultaneously. In addition, other parameters of the training regime could be adjusted to further reduce training and inference time. Another promising research direction is the application of alternative suitable scoring rules such as the Gaussian kernel score. Alternative kernels might induce different characteristics in the predictive distribution and the optimal kernel might be dependent on the underlying data scenario. Future work could also revolve around extending PNO to more complex architectures and larger models. Especially in the context of weather forecasting, models such as the adaptive Fourier neural operator (Pathak et al., 2022) have shown great success and a combination with proper scoring rules might provide a practical method for uncertainty quantification for large-scale systems. In addition, implementing PNO into large-scale models for physical applications such as weather forecasting can lead to improvements in the corresponding uncertainty quantification and extreme event identification, ultimately leading to a deeper understanding of the prediction of such surrogate models.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, December 2021. ISSN 1566-2535. doi: 10.1016/j.inffus.2021.05.008.

Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups*, volume 100 of *Graduate Texts in Mathematics*. Springer, New York, NY, 1984. ISBN 978-1-4612-7017-1 978-1-4612-1128-0. doi: 10.1007/978-1-4612-1128-0.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 619(7970):533–538, July 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06185-3. Publisher: Nature Publishing Group.

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 *ICML'23*, pp. 2806–2823, Honolulu, Hawaii, USA, July 2023. JMLR.org.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton Optimisation for Deep Learning. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 557–565. PMLR, July 2017. ISSN: 2640-3498.

Christopher Bülte, Nina Horat, Julian Quinting, and Sebastian Lerch. Uncertainty quantification for data-driven weather models, March 2024. URL `http://arxiv.org/abs/2403.13458`. arXiv:2403.13458 [physics, stat].

Jieyu Chen, Tim Janke, Florian Steinke, and Sebastian Lerch. Generative machine learning methods for multivariate ensemble postprocessing. *The Annals of Applied Statistics*, 18(1):159–183, March 2024. ISSN 1932-6157, 1941-7330. doi: 10.1214/23-AOAS1784. Publisher: Institute of Mathematical Statistics.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless bayesian deep learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in neural information processing systems*, volume 34, pp. 20089–20103. Curran Associates, Inc., 2021.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33rd international conference on machine learning*, volume 48 of *Proceedings of machine learning research*, pp. 1050–1059, New York, New York, USA, June 2016. PMLR.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. ISSN 0162-1459. doi: 10.1198/016214506000001437.

Vignesh Gopakumar, Stanislas Pamela, Lorenzo Zanisi, Zongyi Li, Ander Gray, Daniel Brennand, Nitesh Bhatia, Gregory Stathopoulos, Matt Kusner, Marc Peter Deisenroth, Anima Anandkumar, the JOREK Team, and MAST Team. Plasma surrogate modelling using Fourier neural operators. *Nuclear Fusion*, 64 (5):056025, April 2024. ISSN 0029-5515. doi: 10.1088/1741-4326/ad313a. Publisher: IOP Publishing.

Ling Guo, Hao Wu, Wenwen Zhou, Yan Wang, and Tao Zhou. IB-UQ: Information bottleneck based uncertainty quantification for neural function regression and neural operator learning, May 2023. URL `http://arxiv.org/abs/2302.03271`. arXiv:2302.03271 [cs, math].

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia De Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, July 2020. ISSN 0035-9009, 1477-870X. doi: 10.1002/qj.3803.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL `http://arxiv.org/abs/1412.6980`. arXiv:1412.6980.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in neural information processing systems*, volume 28. Curran Associates, Inc., 2015.

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. URL `http://jmlr.org/papers/v24/21-1524.html`.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in neural information processing systems*, volume 30. Curran Associates, Inc., 2017.

Bian Li, Hanchen Wang, Shihang Feng, Xiu Yang, and Youzuo Lin. Solving Seismic Wave Equations on Variable Velocity Models With Fourier Neural Operator. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–18, 2023. ISSN 1558-0644. doi: 10.1109/TGRS.2023.3333663. Conference Name: IEEE Transactions on Geoscience and Remote Sensing.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Graph Kernel Network for Partial Differential Equations, March 2020. URL `http://arxiv.org/abs/2003.03485`. arXiv:2003.03485 [cs, math, stat].

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. URL `http://arxiv.org/abs/2010.08895`. arXiv:2010.08895 [cs, math].

Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5.

Russell Lyons. Distance covariance in metric spaces. *The Annals of Probability*, 41(5):3284–3305, September 2013. ISSN 0091-1798, 2168-894X. doi: 10.1214/12-AOP803. Publisher: Institute of Mathematical Statistics.

Ziqi Ma, Kamyar Azizzadenesheli, and Anima Anandkumar. Calibrated Uncertainty Quantification for Operator Learning via Conformal Prediction, February 2024. URL `http://arxiv.org/abs/2402.01960`. arXiv:2402.01960 [cs].

Emilia Magnani, Marvin Pförtner, Tobias Weber, and Philipp Hennig. Linearization Turns Neural Operators into Function-Valued Gaussian Processes, June 2024. URL `http://arxiv.org/abs/2406.05072`. arXiv:2406.05072 [cs, stat].

Lorenzo Pacchiardi, Rilwan A. Adewoyin, Peter Dueben, and Ritabrata Dutta. Probabilistic forecasting with generative networks via scoring rule minimization. *Journal of Machine Learning Research*, 25(45):1–64, 2024.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. FourCastNet: A Global Data-driven High-resolution Weather

Model using Adaptive Fourier Neural Operators, February 2022. URL `http://arxiv.org/abs/2202.11214`. arXiv:2202.11214 [physics].

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. GenCast: Diffusion-based ensemble forecasting for medium-range weather, May 2024. URL `http://arxiv.org/abs/2312.15796`. arXiv:2312.15796 [physics].

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=j3oQF9coJd`.

Stephan Rasp and Sebastian Lerch. Neural Networks for Postprocessing Ensemble Weather Forecasts. *Monthly Weather Review*, 146(11):3885–3900, November 2018. ISSN 0027-0644, 1520-0493. doi: 10.1175/MWR-D-18-0187.1.

Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. WeatherBench 2: A benchmark for the next generation of data-driven global weather models, January 2024. URL `http://arxiv.org/abs/2308.15560`. arXiv:2308.15560 [physics].

Peter I. Renn, Cong Wang, Sahin Lale, Zongyi Li, Anima Anandkumar, and Morteza Gharib. Forecasting subcritical cylinder wakes with Fourier Neural Operators, January 2023. URL `http://arxiv.org/abs/2301.08290`. arXiv:2301.08290 [physics].

Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. Equivalence of distance-based and RKHS-based statistics in hypothesis testing. *The Annals of Statistics*, 41(5):2263–2291, October 2013. ISSN 0090-5364, 2168-8966. doi: 10.1214/13-AOS1140. Publisher: Institute of Mathematical Statistics.

Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do Bayesian Neural Networks Need To Be Fully Stochastic? In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pp. 7694–7722. PMLR, April 2023. ISSN: 2640-3498.

Ingo Steinwart and Johanna F. Ziegel. Strictly proper kernel scores and characteristic kernels on compact spaces. *Applied and Computational Harmonic Analysis*, 51:510–542, March 2021. ISSN 1063-5203. doi: 10.1016/j.acha.2019.11.005.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in neural information processing systems*, volume 35, pp. 1596–1611. Curran Associates, Inc., 2022.

Tapas Tripura and Souvik Chakraborty. Wavelet Neural Operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, February 2023. ISSN 0045-7825. doi: 10.1016/j.cma.2022.115783.

Eva-Maria Walz, Alexander Henzi, Johanna Ziegel, and Tilmann Gneiting. Easy Uncertainty Quantification (EasyUQ): Generating Predictive Distributions from Single-Valued Model Output. *SIAM Review*, 66(1): 91–122, February 2024. ISSN 0036-1445. doi: 10.1137/22M1541915. Publisher: Society for Industrial and Applied Mathematics.

Tobias Weber, Emilia Magnani, Marvin Pförtner, and Philipp Hennig. Uncertainty quantification for fourier neural operators. In *ICLR 2024 workshop on AI4DifferentialEquations in science*, 2024. URL `https://openreview.net/forum?id=knSgoNJcnV`.

Gege Wen, Zongyi Li, Qirui Long, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. Real-time high-resolution CO2 geological storage prediction using nested Fourier neural operators. *Energy & Environmental Science*, 16(4):1732–1741, April 2023. ISSN 1754-5706. doi: 10.1039/D2EE04204E. Publisher: The Royal Society of Chemistry.

Johanna Ziegel, David Ginsbourger, and Lutz Dümbgen. Characteristic kernels on Hilbert spaces, Banach spaces, and on sets of measures. *Bernoulli*, 30(2):1441–1457, May 2024. ISSN 1350-7265. doi: 10.3150/23-BEJ1639. Publisher: Bernoulli Society for Mathematical Statistics and Probability.

David Zwicker. py-pde: A Python package for solving partial differential equations. *Journal of Open Source Software*, 5(48):2158, April 2020. ISSN 2475-9066. doi: 10.21105/joss.02158.

A Model training and hyperparameters

Across the experiments, all uncertainty quantification methods share the same underlying neural operator architecture. The architectures for the Darcy-Flow experiments are taken from Li et al. (2021) and Rahman et al. (2023) for the FNO and UNO, respectively. An overview can be found in table Table 5.

	Model	Batch size	Learning rate	Hidden channels	Projection channels	Lifting channels	Modes
Darcy Flow	FNO	64	0.001	32	256	256	(12,12)
	UNO	64	0.001	[64,128,128, 64,32]	32	32	[(18,18),(8,8),(8,8), (8,8),(18,18)]
KS	FNO	64	0.001	20	128	128	(10,12)
	UNO	32	0.001	[16,32,64,128, 64,32,16]	32	32	[(4,20),(4,14),(4,6),(7,6), (7,6),(10,14),(10,20)]
SSWE	SFNO	64	0.005	32	256	256	(32,32)
ERA5	SNO	16	0.005	20	128	128	(10,12,12)

Table 5: Overview of relevant model hyperparameters.

As the underlying function space, we use the L^2 space and its corresponding norm. The models are trained for a maximum of 1000 epochs and we use early stopping, to stop the training process, if the validation loss does not improve for ten epochs in order to avoid overfitting. For optimization, we employ the popular Adam optimizer (Kingma & Ba, 2017) with gradient clipping. Due to its size, we use an additional learning rate scheduler for the ERA5 dataset, which halves the learning rate if the validation loss does not improve for five epochs. The PNO methods are trained with 3 generated samples, while we use an ensemble size of $M = 100$ across all methods for evaluation. All experiments were implemented in PyTorch and run on an NVIDIA RTX A6000 with 48 GB of RAM. The accompanying code can be found at `https://anonymous.4open.science/r/pfno_anonymous-101C`.

B Effect of Fourier dropout

For the PNO_D stochastic forward passes are required to generate the predictive distribution as the output of the neural operator. As mentioned in Subsection 3.2, we use an additional dropout by masking random components in the Fourier domain (Fourier dropout) of each Fourier block in addition to the standard dropout (weight dropout). In the main paper, we always set the same value for the weight dropout as for the Fourier dropout. Here, we provide an additional study to examine the dependence between the different dropout rates and the performance of the PNO_D. An FNO architecture is trained for the Kuramoto-Sivashinsky equation with different combinations of the dropout. The corresponding results can be found in Table 6, while Figure 5 shows a corresponding visualization. The dropout rates seem to have a structured effect on the predictions, as the L^2 loss and the energy score show better performance with higher dropout rates. However, the NLL seems to be smaller with lower dropout rates. Generally, several of the metrics are minimized by a weight dropout of 0.05 and a higher (L^2, ES) or lower (CRPS) Fourier dropout. However, detecting a significant effect is difficult due to the stochastic nature of the approach and the large computational demand for performing detailed experiments. Still, the experiment suggests that different types of dropout seem beneficial, as they induce more stochasticity in the model that could lead to a higher expressivity with regard to the predictive distribution.

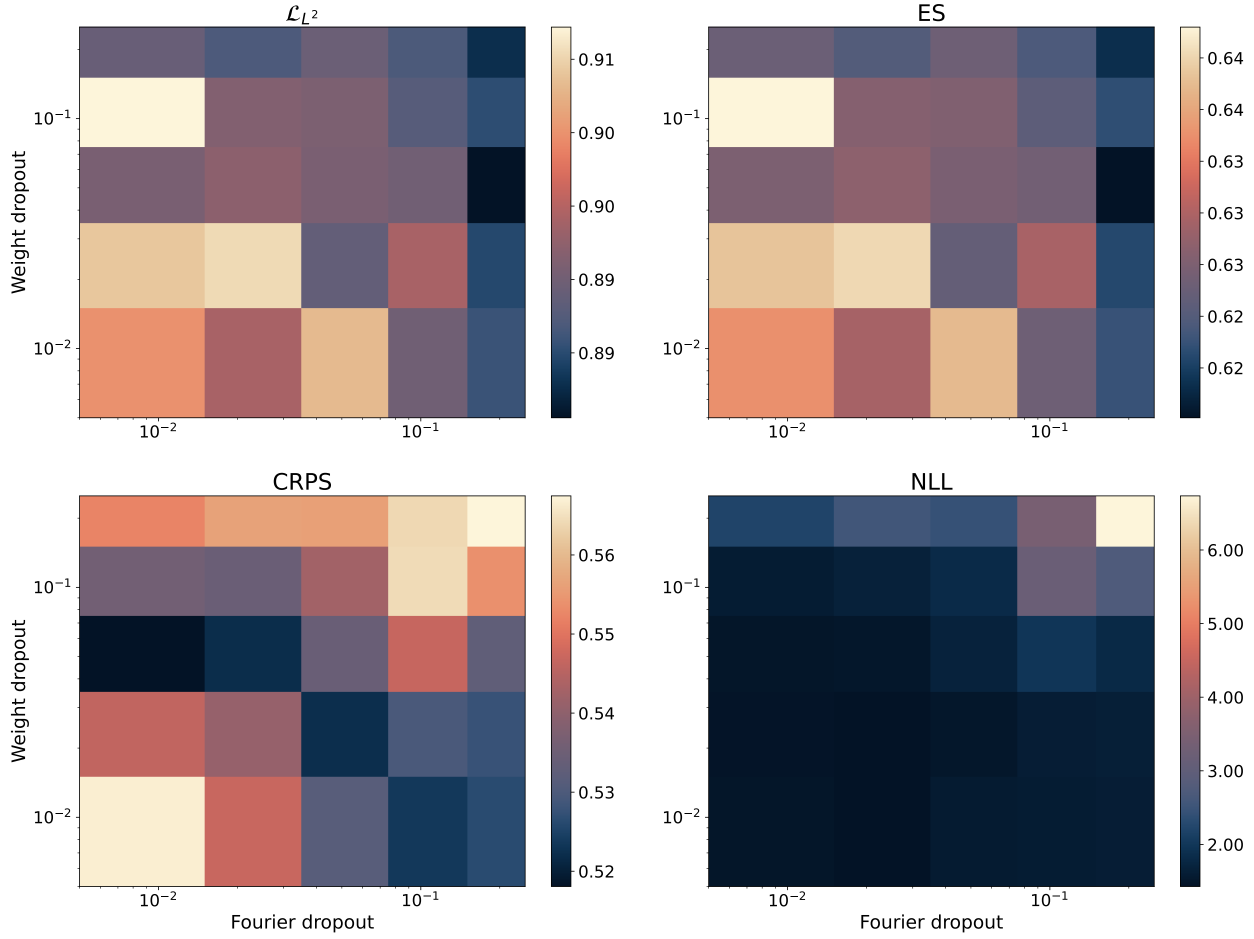

Figure 5: The figure shows the different metrics in dependence of the Fourier and weight dropout for the Kuramoto-Sivashinsky equation. The axes are shown on a log-scale.

Weight dropout	**Fourier dropout**	$\mathcal{L}_{L^2}$	ES	CRPS	NLL	$\mathcal{C}_{0.05}$	$\lvert\mathcal{C}_{0.05}\rvert$
0.01	0.01	0.8998	0.6337	0.5666	1.4866	0.8505	3.4649
	0.02	0.8942	0.6296	0.5469	**1.4281**	0.8746	3.4825
	0.05	0.9032	0.6361	0.5315	1.5714	0.8366	3.2629
	0.1	0.8899	0.6266	0.5236	1.5835	0.8336	3.2151
	0.2	0.8858	0.6237	0.5263	1.6129	0.8294	3.2427
0.02	0.01	0.9042	0.6367	0.5462	1.4577	0.8983	3.6815
	0.02	0.9054	0.6376	0.5410	1.4299	0.9051	3.6425
	0.05	0.8887	0.6259	0.5223	1.5018	0.8667	3.3519
	0.1	0.8942	0.6298	0.5296	1.6132	0.8294	3.2363
	0.2	0.8848	0.6231	0.5276	1.6191	0.8279	3.2578
0.05	0.01	0.8907	0.6274	**0.5181**	1.4798	0.8712	3.2865
	0.02	0.8922	0.6284	0.5221	1.4968	0.8660	3.3190
	0.05	0.8908	0.6273	0.5342	1.6776	0.8198	3.2360
	0.1	0.8900	0.6268	0.5466	1.9742	0.7766	3.1370
	0.2	**0.8806**	**0.6201**	0.5327	1.7949	0.8078	3.2122
0.1	0.01	0.9072	0.6390	0.5357	1.5733	0.8418	3.2286
	0.02	0.8914	0.6280	0.5344	1.6708	0.8151	3.1668
	0.05	0.8910	0.6276	0.5425	1.8126	0.7929	3.1337
	0.1	0.8877	0.6255	0.5643	3.1770	0.7391	3.0874
	0.2	0.8852	0.6234	0.5537	2.7249	0.7638	3.0821
0.2	0.01	0.8891	0.6264	0.5520	2.2083	0.7598	2.9729
	0.02	0.8870	0.6249	0.5563	2.5560	0.7431	2.9557
	0.05	0.8895	0.6266	0.5561	2.4287	0.7527	3.0111
	0.1	0.8869	0.6247	0.5640	3.4221	0.7352	2.9904
	0.2	0.8828	0.6217	0.5675	6.7351	0.7319	3.0469

Table 6: Metrics for different combinations of dropout for training on the Kuramoto-Sivashinsky equation.

C Runtime analysis

Since using a scoring rule as the loss function requires predicting multiple samples for each optimization step, a runtime analysis of PNO is of high interest. In Table 7, we show the estimated runtime of MCD, LA, PNO_D, and PNO_R. As expected, both PNO methods require significantly more time per epoch, with PNO_D taking about three times as long as the baseline methods and PNO_R about 50% longer. The main bottleneck is the evaluation of the energy score, which requires to calculate a distance matrix for all predictive samples. Furthermore, PNO_D requires several forward passes per optimization step, increasing the runtime even further. PNO_R on the other hand requires more memory and had to be trained with a smaller batch size (16 instead of 32), which also increases the computation time per epoch compared to MCD and LA.

Method	Avg. Number of Epochs	Time per Epoch (s)	Total Time (s)
MCD	**112.89**	22.36	**22718**
LA	140.00	**21.56**	27161
PNO_D	132.33	58.71	69920
PNO_R	209.78	31.92	60272

Table 7: Runtime comparison on SSWE with one-step autoregressive training. The number of epochs is averaged across the ten training runs using early stopping with 10 epochs of patience. The timings are averaged over 25 training epochs on an RTXA6000 and the best models are highlighted in bold.

D Influence of the number of predictive samples on the performance of PNO

Our PNO approach depends on utilizing an unbiased estimator of the energy score in Equation 4, which requires the model to generate $M > 1$ samples. While in practice we found that even for only $M = 3$ generated samples, the model shows improved performance, increasing the sample size is mainly hindered by computational reasons. In order to further analyze the model training in dependence of the sample size M, we trained both PNO approaches on the Kuramoto-Sivashinsky equation with different training sample sizes M. For evaluation, all models still use an ensemble size of one hundred members across the test data. Table 8 shows the corresponding results, which are again aggregated over ten runs. For the PNO_R, the L^2 loss and energy score already perform well for $M = 3$ samples, while the other metrics are improved by increasing the sample size. While the training time per epoch increases, the differences seem negligible for less than twenty samples. On the contrary, the PNO_D shows the best performance, if $M \geq 10$ samples are already used during the training process. Especially the negative log-likelihood improves significantly. However, for the PNO_D, the training time per epoch increases drastically, as the model also requires M stochastic forward passes. On the other hand, for the PNO_D, the number of training epochs decreases with the sample size. A solution might be using different training regimes, starting with a low sample size and increasing it during the training process, to reduce the computational demand.

Model	M	t	∅ epochs	$\mathcal{L}_{L^2}$	ES	CRPS	NLL	$\mathcal{C}_{0.05}$
$\mathbf{PNO_D}$	3	**37.72s**	151	0.8793 (±0.0072)	0.6195 (±0.0049)	0.5496 (±0.0097)	3.0389 (±0.5872)	0.7640 (±0.0191)
	5	44.58s	154	0.8790 (±0.0059)	0.6191 (±0.0038)	0.5577 (±0.0080)	5.2872 (±1.8802)	0.7541 (±0.0189)
	10	59.94s	126	0.8799 (±0.0021)	0.6196 (±0.0015)	**0.5348** (±0.0038)	2.3471 (±0.2235)	0.8001 (±0.0095)
	20	90.64s	135	**0.8784** (±0.0091)	**0.6188** (±0.0061)	0.5566 (±0.0088)	5.3147 (±1.5163)	0.7592 (±0.0170)
	50	181.45s	**98**	0.8857 (±0.0050)	0.6239 (±0.0036)	0.5356 (±0.0030)	**1.9174** (±0.1097)	0.8036 (±0.0092)
$\mathbf{PNO_R}$	3	31.08s	163	**0.8640** (±0.0038)	**0.6081** (±0.0027)	0.4743 (±0.0037)	1.2268 (±0.0083)	0.9401 (±0.0030)
	5	**29.75s**	146	0.8661 (±0.0036)	0.6096 (±0.0025)	0.4751 (±0.0032)	1.3768 (±0.4510)	0.9421 (±0.0027)
	10	31.35s	152	0.8649 (±0.0036)	0.6088 (±0.0025)	0.4737 (±0.0038)	1.2180 (±0.0158)	0.9437 (±0.0020)
	20	35.06s	**135**	0.8680 (±0.0053)	0.6110 (±0.0037)	0.4756 (±0.0051)	1.2198 (±0.0181)	0.9453 (±0.0027)
	50	43.43s	181	0.8649 (±0.0035)	0.6088 (±0.0025)	**0.4733** (±0.0027)	**1.2138** (±0.0085)	0.9449 (±0.0028)

Table 8: Comparison of the performance of PNO_D and PNO_R in dependence of the training sample size M for the Kuramoto-Sivashinsky equation. The best results are highlighted in bold, where t denotes the average training time per epoch.

E Additional figures

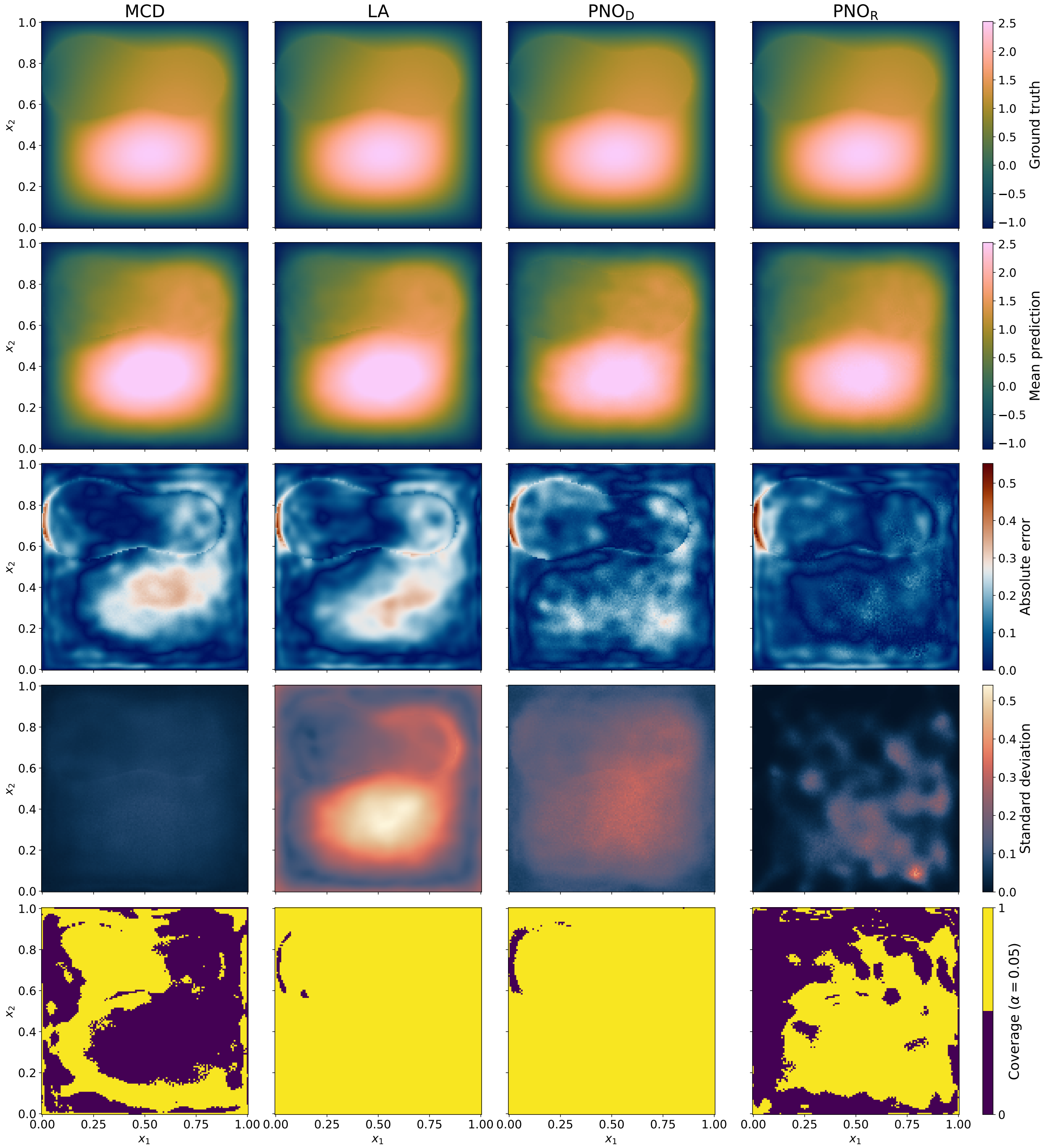

Figure 6: The figure shows the ground truth, mean prediction, absolute error, standard deviation, and coverage for the different methods on a sample of the Darcy-Flow equation.

F Additional results for SSWE

Δt	**Method**	$\mathcal{L}_{L^2}$	ES	CRPS	NLL	$\mathcal{C}_{0.05}$	$\lvert\mathcal{C}_{0.05}\rvert$
1h	PNO_D	**0.2955** ($\pm$0.0207)	**0.2202** ($\pm$0.0132)	**0.0896** ($\pm$0.0051)	**-0.4609** ($\pm$0.0463)	**0.9840** ($\pm$0.0032)	0.8923 ($\pm$0.0467)
	PNO_R	0.3132 ($\pm$0.0244)	0.2208 ($\pm$0.0170)	0.1048 ($\pm$0.0069)	2.5×10^7 ($\pm3.6\times10^7$)	0.6912 ($\pm$0.0897)	0.4971 ($\pm$0.0842)
	MCD	0.3845 ($\pm$0.0206)	0.2972 ($\pm$0.0194)	0.1193 ($\pm$0.0102)	0.4516 ($\pm$0.5340)	0.7063 ($\pm$0.0496)	0.3819 ($\pm$0.0123)
	LA	0.4222 ($\pm$0.0511)	0.3063 ($\pm$0.0345)	0.1346 ($\pm$0.0135)	0.1821 ($\pm$0.2472)	0.9024 ($\pm$0.0694)	0.9606 ($\pm$0.2178)
10h	PNO_D	2.8×10^3 ($\pm8.3\times10^3$)	2.7×10^3 ($\pm8.0\times10^3$)	22.3235 ($\pm$65.581)	**6.0553** ($\pm$0.7769)	0.4695 ($\pm$0.0207)	43.754 ($\pm$128.49)
	PNO_R	9.4×10^3 ($\pm2.0\times10^4$)	7.6×10^3 ($\pm1.6\times10^4$)	3.6×10^3 ($\pm7.7\times10^3$)	1.7×10^{20} ($\pm4.6\times10^{20}$)	0.3056 ($\pm$0.1644)	1.3×10^4 ($\pm3.2\times10^4$)
	MCD	**1.3324** ($\pm$0.2282)	**1.2066** ($\pm$0.2177)	**0.5043** ($\pm$0.0739)	25.394 ($\pm$3.3598)	0.2511 ($\pm$0.0047)	0.4104 ($\pm$0.0304)
	LA	1.5445 ($\pm$0.2696)	1.2835 ($\pm$0.2493)	0.5412 ($\pm$0.1041)	7.2994 ($\pm$6.0597)	0.5254 ($\pm$0.1033)	1.0291 ($\pm$0.2616)

Table 9: Results for the spherical shallow water equation with single-step training. The best model is highlighted in bold and the standard deviation is given in the brackets.