Uncertainty Quantification for Forward and Inverse Problems of PDEs via Latent Global Evolution

Tailin Wu[1*], **Willie Neiswanger**[2*], **Hongtao Zheng**[1*], **Stefano Ermon**[3], **Jure Leskovec**[3]

[1] School of Engineering, Westlake University
[2] Department of Computer Science, University of Southern California
[3] Computer Science Department, Stanford University
{wutailin, zhenghongtao}@westlake.edu.cn, neiswang@usc.edu, {ermon, jure}@cs.stanford.edu

Abstract

Deep learning-based surrogate models have demonstrated remarkable advantages over classical solvers in terms of speed, often achieving speedups of 10 to 1000 times over traditional partial differential equation (PDE) solvers. However, a significant challenge hindering their widespread adoption in both scientific and industrial domains is the lack of understanding about their prediction uncertainties, particularly in scenarios that involve critical decision making. To address this limitation, we propose a method that integrates efficient and precise uncertainty quantification into a deep learning-based surrogate model. Our method, termed Latent Evolution of PDEs with Uncertainty Quantification (LE-PDE-UQ), endows deep learning-based surrogate models with robust and efficient uncertainty quantification capabilities for both forward and inverse problems. LE-PDE-UQ leverages latent vectors within a latent space to evolve both the system's state and its corresponding uncertainty estimation. The latent vectors are decoded to provide predictions for the system's state as well as estimates of its uncertainty. In extensive experiments, we demonstrate the accurate uncertainty quantification performance of our approach, surpassing that of strong baselines including deep ensembles, Bayesian neural network layers, and dropout. Our method excels at propagating uncertainty over extended auto-regressive rollouts, making it suitable for scenarios involving long-term predictions. Our code is available at: https://github.com/AI4Science-WestlakeU/le-pde-uq.

Introduction

Partial differential equations have wide-ranging applications in both scientific and engineering domains. It is noteworthy that time-dependent partial differential equations characterize the evolution of complex system states over time, serving as crucial tools for forward prediction and reverse optimization across various disciplines. These applications span a wide spectrum, including weather forecasting (Lynch 2008; Bi et al. 2023), nuclear fusion (Carpanese 2021), jet engine design (Sircombe, Arber, and Dendy 2006), astronomical simulation (Courant, Friedrichs, and Lewy 1967), molecular modeling (Lelievre and Stoltz 2016), and physical simulation modeling (Wu et al. 2022, 2023), to name just a few. When addressing real-world challenges in science and engineering, the sheer volume of cells per time step can readily escalate into the millions or beyond. This complexity poses a substantial hurdle for conventional PDE solvers to ensure rapid solutions. In addition, inverse optimization such as inverse inference of system parameters also face similar scale challenges, besides the modeling of forward evolution (Biegler et al. 2003). Consequently, numerous deep learning-based alternative models have emerged that can accelerate the speed of partial differential equation solving by orders of magnitude (typically 10 to 1000 times), such as (Li et al. 2020).

However, recent neural network-based PDE solvers share a common drawback – they typically fail to provide any form of uncertainty estimation for their proposed solutions. This can lead to an overconfidence or underconfidence in the accuracy of the approximate solutions generated by PDE solvers, potentially resulting in relying on inaccurate approximations without any indication of associated risks. The concept of uncertainty quantification refers to the process of assessing and measuring the uncertainty associated with outcomes during prediction or optimization. In various scientific and engineering applications, the predictive results of models can be influenced by multiple factors, such as data noise, model uncertainty, parameter estimation, and more. Uncertainty quantification aims to provide information about the credibility or confidence level of the predictive outcomes, enhancing the understanding of the model's reliability. This information is valuable for considering uncertainty during decision-making processes. By quantifying uncertainty, we can better assess the stability, accuracy, and reliability of the model's predictions across different scenarios, enabling more informed decision-making.

Currently, most of the work related to uncertainty quantification in the process of PDE solving largely excludes consideration of temporal states (Winovich, Ramani, and Lin 2019; Zhu et al. 2019). This is mainly because, for time-domain partial differential equations, surrogate models need to perform autoregressive rollouts, which result in the accumulation of uncertainty over time. In the Julia programming language, there are libraries for uncertainty quantification (UQ), but none is designed specifically for neural surrogate models dealing with time-varying PDEs. An important

[*]These authors contributed equally.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

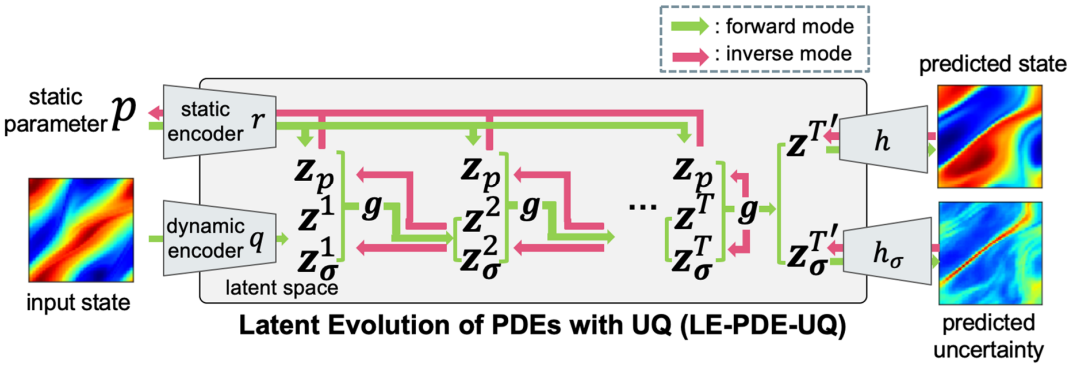

Figure 1: Schematic representation of the LE-PDE-UQ framework. In the forward mode(green), LE-PDE-UQ evolves the dynamics in the global latent space. In the inverse optimization mode (red), it optimizes the parameter p(e.g., the boundary) by unrolling the latent vectors. The compressed latent vectors and dynamics can significantly speed up both modes. The latent evolution model g deterministically predicts a global latent vector z encoding the state and a global latent vector Z_σ encoding the uncertainty. On demand, they are decoded into the predicted state and the predicted uncertainty, respectively.

consideration in temporal PDE uncertainty quantification is that the prediction process concurrent with PDE solving may adversely affect the speed and accuracy of the solution. Addressing this issue, we draw inspiration from LE-PDE ((Wu, Maruyama, and Leskovec 2022)), which leverages latent representations to efficiently transfer valid information and capture the global features of the input state, reducing information redundancy and noise. The use of latent representations also significantly reduces the data dimensionality, thereby accelerating model inference and backpropagation.

To bridge the technical gap in uncertainty quantification for time-domain PDEs and address the aforementioned challenges, we introduce a novel framework named Latent Evolution of PDEs with Uncertainty Quantification (LE-PDE-UQ). This approach is simple, fast, and scalable, accurately quantifying the uncertainty arising in both the forward evolution and inverse optimization of PDEs. The comprehensive network structure of LE-PDE-UQ is showcased in Fig. 1. The method evolves the state of the system and the uncertainty estimation of the state by corresponding latent vectors in the latent space, and decodes them as state prediction and uncertainty estimation, respectively. The specific framework of LE-PDE-UQ will be described in Section 3 of this paper.

We have also shown in subsequent experiments that our method achieves state-of-the-art results in uncertainty estimation for forward evolution and inverse optimization of PDEs, is able to propagate uncertainty in long-term autoregressive prediction, outperforming strong baseline methods (e.g., deep ensembles, Bayes layers, Dropout, etc.). This shows that our approach is able to efficiently model the evolution of temporal PDEs and achieve accurate uncertainty estimation, improving the performance and trustworthiness in complex scientific and engineering problems.

Related Work

In recent years, significant efforts have been devoted to addressing the aforementioned challenges. Much of the prior work has revolved around the Bayesian formalism (Bernardo and Smith 2009), wherein a prior distribution is assigned to the parameters of neural networks. Subsequently, given the training data, posterior distributions over the parameters are computed to quantify predictive uncertainty. However, precise Bayesian inference poses computational challenges for neural networks, leading to the development of various approximation methods, including Laplace approximation (Mackay 1992), Markov chain Monte Carlo (MCMC) methods (Neal 2012), as well as variational Bayesian methods (Blundell et al. 2015; Graves 2011; Louizos and Welling 2016), among others. The quality of predictive uncertainty obtained from Bayesian neural networks primarily depends on (1) the level of approximation due to computational constraints, and (2) the correctness of the chosen prior distribution, as convenient priors can result in unreasonable predictive uncertainties (Rasmussen and Quinonero-Candela 2005). In practice, Bayesian neural networks are often more challenging to implement and slower to train compared to non-Bayesian counterparts, necessitating a general-purpose solution that can offer high-quality uncertainty estimates with only minor modifications to the standard training pipeline.

Hence, (Gal and Ghahramani 2016) proposed the use of Monte Carlo dropout (MC-dropout) during testing, utilizing dropout (Srivastava et al. 2014) to estimate predictive uncertainty. Substantial research has also been conducted on approximate Bayesian interpretations of dropout (Gal and Ghahramani 2016; Kingma, Salimans, and Welling 2015). MC-dropout's implementation is relatively straightforward and yields favorable results, making it widely popular in practice. Dropout can also be interpreted as ensemble model combination (Srivastava et al. 2014), where predictions are averaged over an ensemble of neural networks (with shared parameters). The ensemble interpretation appears more reasonable, especially when dropout rates are not adjusted based on training data, as any sensible approximation to the true Bayesian posterior distribution must depend on the training data. This interpretation has spurred investigations into ensembles as an alternative solution for estimat-

ing predictive uncertainty.

Over time, the enhanced predictive performance resulting from the utilization of model ensembles has been increasingly acknowledged by researchers. Ensembles perform model combination, where multiple models are integrated to achieve a more robust model. Ensembles are expected to perform better when the true model lies outside the hypothesis class (Lakshminarayanan, Pritzel, and Blundell 2017; Wenzel et al. 2020).

Preliminaries

LE-PDE-UQ builds upon the prior work of LE-PDE (Wu, Maruyama, and Leskovec 2022). LE-PDE-UQ shares LE-PDE's advantage of fast, accurate and scalable forward prediction and inverse optimization of PDEs, but also with notable innovations for uncertainty quantification. Below, we will provide a brief introduction to LE-PDE. The LE-PDE model architecture comprises four key components:

q: dynamic encoder:$z^k = q(U^k)$
r: static encoder: $z_p = r(p)$
g: latent evolution model: $z^{k+1} = g(z^k, z_p)$
h: decoder: $\hat{U}^{k+1} = h(z^{k+1})$

LE-PDE utilize the temporal bundling technique (Brandstetter, Worrall, and Welling 2022) to enhance the representation of sequential data. This approach involves grouping input states U^k across a fixed interval S of consecutive time steps. Consequently, each latent vector $\mathbf{z}_k$ encodes this bundle of states, and latent evolution predicts the next $\mathbf{z}$ for the subsequent S steps. The parameter S, a hyperparameter, is adaptable to the specific problem, and setting $S = 1$ results in no bundling. It's crucial to note that the dynamic encoder q should feature a flattening operation and a Multi-Layer Perception (MLP) head that transforms feature maps into a single fixed-length vector $\mathbf{z} \in R^{d_z}$. By doing so, the latent space's dimensionality doesn't increase linearly with input dimension, allowing substantial data compression and rendering long-term predictions more efficient.

In addition to enhancing forward simulations, LE-PDE can accelerate inverse optimization. This involves using backpropagation through time (BPTT) to adjust system parameters p within the latent space, minimizing a predefined objective function $L_d[p]$. This is crucial in engineering, where optimizing boundary conditions or equation parameters is essential. LE-PDE encodes initial state U_0 and system parameters p into latent vectors z_0 and z_p using acquired latent space knowledge and the evolutionary model. Latent evolution takes place, and if needed, it returns to the input space to calculate $L_d[p]$. By computing the gradient of $L_d[p]$ with respect to p and using methods like Adam optimization, an approximate optimal p can be found. With the significantly smaller latent space dimension, this method reduces the complexity of inverse optimization. For more details on LE-PDE, refer to Appendix A.

Our Approach LE-PDE-UQ

In this section, we provide a detailed explanation of our LE-PDE-UQ method. We begin by presenting the complete architecture of the algorithm framework, as illustrated in Fig. 1. Subsequently, we introduce the learning objectives for effectively capturing long-term evolution. Finally, we describe the efficient inverse optimization approach within the latent space enabled by our method.

Algorithm Framework

The model architecture of LE-PDE-UQ consists of five components: (1) a dynamic encoder $q : \mathbb{U} \to \mathbb{R}^{d_z}$ that maps the input state $U^t = \{\mathbf{u}_i^t\}_{i=1}^N \in \mathbb{U}$ to a tuple of (latent-vector, latent-uncertainty-vector): $(\mathbf{z}^t, \mathbf{z}_\sigma^t) = q(U^t) \in \mathbb{R}^{d_z}$; (2) an (optional) static encoder $r : \mathbb{P} \to \mathbb{R}^{d_{zp}}$ that maps the (optional) system parameter $p \in \mathbb{P}$ to a static latent embedding $\mathbf{z}_p$=$r(p)$; (3) a decoder $h_\mu : \mathbb{R}^{d_z} \to \mathbb{U}$ that maps the latent vector $\mathbf{z}^t \in \mathbb{R}^{d_z}$ back to the input state U^t; (4) a latent evolution model $g : \mathbb{R}^{d_z} \times \mathbb{R}^{d_{zp}} \to \mathbb{R}^{d_z}$ that maps $\mathbf{z}^t, \mathbf{z}_\sigma^t \in \mathbb{R}^{d_z}$ at time t and static latent embedding $\mathbf{z}_p \in \mathbb{R}^{d_{zp}}$ to $\mathbf{z}^{t+1}, \mathbf{z}_\sigma^{t+1} \in \mathbb{R}^{d_z}$ at time $t+1$; (5) uncertainty decoder $h_\sigma : \mathbb{R}^{d_z} \to \mathbb{U}$ that maps the latent uncertainty vector $\mathbf{z}_\sigma^t \in \mathbb{R}^{d_z}$ back to the predicted uncertainty U_σ^t. Here the latent evolution model g is decomposed as:

$$\begin{aligned} z^{t+1} &= g_\mu(z^t, z_p) \\ z_\sigma^{t+1} &= g_\sigma([z^t, z_\sigma^t], z_p) \end{aligned} \tag{1}$$

Note that the latent vector z^{t+1} only depends on z^t, z_p, while the latent uncertainty vector depends on z^t, z_p, and the latent uncertainty vector at z_σ^t previous time step, modeling the propagation of uncertainty in latent space. We employ the temporal bundling trick (Brandstetter, Worrall, and Welling 2022) where each input state U^t can include states over a fixed length S of consecutive time steps. At inference time, LE-PDE-UQ performs autoregressive rollout in latent space $\mathbb{R}^{d_z}$:

$$\begin{aligned} &(\hat{U}^{t+m}, \hat{U}_\sigma^{t+m}) = (h, h_\sigma) \circ g\left(\cdot, r(p)\right)^{(m)} \circ q(\hat{U}^t) \\ &\equiv (h, h_\sigma)\Bigg(\underbrace{g(\cdot, r(p)) \circ ... \circ g(\cdot, r(p))}_{\text{composing } m \text{ times}} \Big(q(\hat{U}^t)\Big) \Bigg) \end{aligned} \tag{2}$$

Compared to autoregressive rollout in input space, LE-PDE-UQ can significantly improve efficiency with a much smaller dimension of $\mathbf{z}^t \in \mathbb{R}^{d_z}$ compared to $U^t \in \mathbb{U}$. Moreover, it efficiently models the propagation of uncertainty in latent space, using the latent uncertainty vector z_σ^t. Here we do not limit the architecture for encoder, decoder and latent evolution models. Depending on the input U^t, the encoder q and decoder h_μ can be a CNN or GNN with a (required) MLP head. We model the latent evolution model g as an MLP with residual connection from input to output. During forward prediction, we also augment the above architecture with Deep Ensemble (Lakshminarayanan, Pritzel, and Blundell 2017) for improved uncertainty quantification.

Learning Objective

Given discretized inputs $\{U^t\}, t = 1, ...K + M$, our LE-PDE-UQ model is trained with the following objective that combines negative log-likelihood in the input space, reconstruction, and long-term consistency in the latent space:

$$L = \frac{1}{T}\sum_{t=1}^{T}(L_{\text{multi-step}}^t + L_{\text{recons}}^t + L_{\text{consistency}}^t) \tag{3}$$

Here ℓ is the loss function for individual predictions, which can typically be MSE or L2 loss. $\hat{U}^{t+m}$ is given in Eq. (2). L^t_{recons} aims to reduce reconstruction loss. $L^t_{\text{multi-step}}$ performs latent multi-step evolution given in Eq. (2) and compare with the target U^{t+m} in input space, up to time horizon M. α_m are weights for each time step, which we find that $(\alpha_1, \alpha_2, ...\alpha_M) = (1, 0.1, 0.1, ...0.1)$ works well. Besides encouraging better prediction in input space via $L^t_{\text{multi-step}}$, we also want a stable long-term rollout in latent space. This is because in inference time, we want to mainly perform autoregressive rollout in latent space, and decode to input space only when needed. Thus, we introduce a novel latent consistency loss $L^t_{\text{consistency}}$, which compares the m-step latent rollout $g\left(\cdot, r(p)\right)^{(m)} \circ q(U^t)$ with the latent target $q(U^{t+m})$ in *latent* space. The denominator $||q(U^{t+m})||_2^2$ serves as normalization to prevent the trivial solution that the latent space collapses to a single point. Taken together, the three terms encourage a more accurate and consistent long-term evolution both in latent and input space.

Inverse Optimization

In addition to improved efficiency for forward simulation, LE-PDE-UQ also allows more efficient solving of inverse problems, via backpropagation through time (BPTT) in latent space. Given a specified objective $L_d[p, U^0] = \sum_{k=k_s}^{k_e} \ell(U^t)$ which is a discretized version of $L_d[\mathbf{a}, \partial\mathbb{X}]$, we define the objective:

$$L_d[p, U^0] = \sum_{m=k_s}^{k_e} \ell_d(\hat{U}^m(p, U^0)) \tag{4}$$

For inverse problems that infer unknown parameters or initial state (so that all future state can be known), the objective L_d can be an MSE between predicted future states $\hat{U}^m$ and the observed future states U^m. For inverse design tasks, L_d can be specific design objectives such as lift-drag ratio for plane shape design. $\hat{U}^m = \hat{U}^m(p, U^0)$ is given by Eq. (4) setting $k = 0$ using our learned LE-PDE-UQ, which starts at initial state of U^0, encode it and p into latent space, evolves the dynamics in latent space and decode to $\hat{U}^m$ as needed. The static latent embedding $\mathbf{z}_p = r(p)$ influences the latent evolution at each time step via $g(\cdot, r(p))$. The initial state U^0 influences the future state via the latent vectors $(\mathbf{z}^0, \mathbf{z}^0_\sigma) = q(U^0)$. To perform inverse optimization w.r.t. the high-dimensional initial state U^0, we optimize w.r.t. $(\mathbf{z}^0, \mathbf{z}^0_\sigma)$ first and then use the decoder h to decode $\mathbf{z}^0$ to an estimated $\hat{U}^0$. This is different from LE-PDE where we optimize directly w.r.t the input variable. This is because $\hat{U}^0$ has a much higher dimension than $(\mathbf{z}^0, \mathbf{z}^0_\sigma)$, and optimizing w.r.t. $\hat{U}^0$ can lead to adversarial modes, as is also seen in (Zhao, Lindell, and Wetzstein 2022). Instead, optimizing w.r.t. $(\mathbf{z}^0, \mathbf{z}^0_\sigma)$ then decode leads to more physical estimation of U^0. To obtain uncertainty for the inverse optimization, we employ Deep Ensemble (Lakshminarayanan, Pritzel, and Blundell 2017) to obtain estimated uncertainty.

Experiments

In the experiments, our goal is to answer the following questions: (1) Can LE-PDE-UQ accurately quantify the uncertainty arising from the long-term evolution of complex systems and compete with state-of-the-art methods? (2) Which components of LE-PDE-UQ effectively enhance its uncertainty quantification capability in forward problem inference? (3) How does LE-PDE-UQ perform in quantifying uncertainty during the model's inverse optimization process? The experimental section on Forward Problems is primarily aimed at addressing questions (1) and (2), while the section on Inverse optimization is mainly focused on addressing question (3). We evaluate the models with two aspects: quality in uncertainty quantification measured by miscalibration area (MA), mean absolute calibration error (MACE), and root mean squared calibration error (RMSCE); and quality in point prediction, measured by relative L2 loss and mean absolute error (MAE) (see Appendix B for more details). The evaluation is generated using the Uncertainty Toolbox package (Chung et al. 2021).

Dataset

We have tested the LE-PDE-UQ within a 2D benchmark based on the Navier-Stokes equation. The Navier-Stokes equation has wide applications in science and engineering, including fields like weather forecasting and jet engine design. Simulation becomes more challenging when entering the turbulent phase, which exhibits multiscale dynamics and chaotic behavior. Specifically, we test our model in a viscous, incompressible fluid in vorticity form in a unit torus:

$$\begin{aligned} \partial_t w(t,x) + u(t,x) \cdot \nabla w(t,x) &= \nu \Delta w(t,x) + f(x) \\ \nabla \cdot u(t,x) &= 0 \\ w(0,x) &= w_0(x) \\ x \in (0,1)^2, t &\in [0,T] \end{aligned} \tag{5}$$

where $w(t,x) = \nabla \times u(t,x)$ is the vorticity, $\nu \in \mathbb{R}_+$ is the viscosity coefficient. The domain is discretized into 64×64 grid and $Re = 10^4$ (turbulent). The dataset comprises a total of 1200 trajectories (among them, 1000 trajectories are used as the training set, and an additional 200 trajectories are used as the test set), with a total of 20 time points sampled along each trajectory.

Forward Problems

In this section, we address questions (1) and (2). We compare the most widely used uncertainty quantification algorithms, including Bayes layer (Tran et al. 2019), Dropout (Srivastava et al. 2014) and Deep Ensembles (Lakshminarayanan, Pritzel, and Blundell 2017). We also explore the effects of important components of our model, including latent evolution (with ablation model of NoLatent that evolves the state in input space), evolving latent uncertainty vector $\mathbf{z}^t_\sigma$, and ensembling. To ensure a fair comparison, all models utilize the past 10 steps to predict the next step, and autoregressively predict future 10 steps.

The experiment results are shown in Table 1. We see that: (1) Our full LE-PDE-UQ method attains the best performance in UQ and prediction error. (2) As demonstrated by

	MA	MACE	RMSCE	L2	MAE
Bayes layer with Latent	0.0445	0.0440	0.0500	0.2345	0.2051
Bayes layer without Latent	0.2381	0.2357	0.2665	0.2105	0.1830
Dropout, L2=0	0.1778	0.1760	0.1979	0.2079	0.1938
Dropout, L2=10^{-5}	0.1924	0.1905	0.2143	0.2092	0.1958
Dropout, L2=10^{-4}	0.2317	0.2294	0.2588	0.2458	0.2320
Dropout, L2=10^{-3}	0.3281	0.3248	0.3704	0.3534	0.3428
NoLatent (single, with σ)	0.1045	0.1035	0.1175	0.2053	0.1817
NoLatent (ensemble, without σ)	0.2118	0.2096	0.2355	0.1939	0.1657
NoLatent (ensemble, with σ)	0.0602	0.0596	0.0662	0.1939	0.1657
Latent (single, without σ)	-	-	-	**0.1890**	0.1613
Latent (single, with σ)	0.0576	0.0570	0.0649	0.2108	0.1811
Latent (ensemble, without σ)	0.1823	0.1805	0.2024	0.1895	**0.1608**
Latent (**ours**, ensemble, with σ)	**0.0142**	**0.0141**	**0.0160**	0.1895	**0.1608**

Table 1: Accuracy of different methods for uncertainty quantification in forward problems. The Bayes layer with Latent method refers to its combination of the concepts of Bayesian layer and latent space. Dropout, $L2 = 10^{-5}$ indicates that this approach simultaneously utilizes both Dropout technique and L2 regularization. Under NoLatent and Latent models, 'single' refers to using a single model for prediction or analysis, while "ensemble" refers to using an ensemble of 10 models for prediction or analysis. σ refers to that a single model can also predict uncertainty. Bold font represents the best results among the methods, while underline indicates second-best.

our model (Latent) and NoLatent counterparts, latent evolution significantly improves UQ: it can propagate uncertainty via long-term rollout, (3) Comparing with and without single model uncertainty prediction (i.e., with or without σ), we see that without σ, even with ensembling, the UQ is significantly worse. Note that "NoLatent (ensemble, with σ)" denotes the Deep Ensembles method. (4) Bayes layer performs well in UQ, but significantly worse in prediction error. Dropout performs poorly in both UQ and prediction error.

Fig. 2 shows the visualization of the prediction and uncertainty quantification by our algorithm. We see that our model's prediction (first row) matches excellently with the ground-truth (second row), including both global and fine-grained spatial features. More importantly, our model's predicted uncertainty (third row) shows excellent similarity with the actual absolute error (fourth row), demonstrating accurate uncertainty quantification achieved by LE-PDE-UQ. Fig. 3 shows the ordered prediction intervals and average calibration plot of the LE-PDE-UQ algorithm. From the left plot, it can be observed that the actual observed points (in yellow) align closely with the blue predicted region (with blue dots representing the center points of the prediction intervals). Simultaneously, in the right plot, the predicted confidence levels align precisely with the frequency of actual observations (evident from the close fit of the blue prediction line to the diagonal line), demonstrating our model's well-calibrated uncertainty quantification. We observe that the fit of the blue curve to the diagonal in the right panel of Fig. 4 is much worse than in Fig. 3. From this we can conclude that Latent is critical to the ability to quantify model uncertainty.

Effect of latent uncertainty propagation using $\mathbf{z}_\sigma^t$. To further investigate how much of the performance difference is attributed to the ability to model uncertainty propagation rather than model architecture differences, we conducted comparative experiments involving two distinct strategies

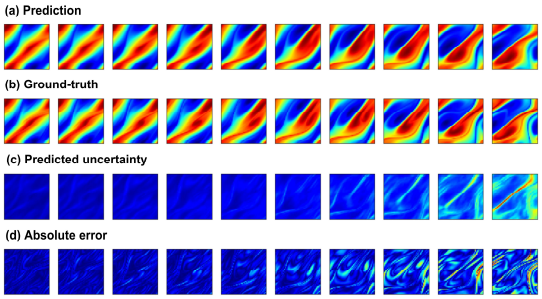

Figure 2: Visualization of forward prediction results of LE-PDE-UQ on 2D Navier-Stokes turbulent flow dataset. The figure predicts the fluid state from 11-20 steps using the actual fluid state data from 1-10 steps. In this context, (a) represents the actual fluid state from 11-20 steps, while (b) indicates the fluid state predicted by LE-PDE-UQ. (c) represents the absolute error between (a) and (b), and (d) represents the uncertainty quantification results obtained by LE-PDE-UQ.

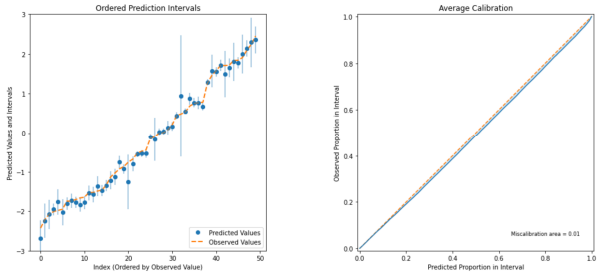

Figure 3: The ordered prediction intervals and average calibration of LE-PDE-UQ.

(Autoregressive rollout and Teacher-Forcing) used by the model during inference. While Autoregressive rollout uses

Augoregressive rollout	MA	MACE	RMSCE	L2	MAE
NoLatent (ensemble, with σ)	0.0602	0.0596	0.0662	0.1939	0.1657
Latent (**ours**, ensemble, with σ)	**0.0142**	**0.0141**	**0.0160**	**0.1895**	**0.1608**

Table 2: Results of Auto-regressive Rollout Experiment

Teacher-forcing	MA	MACE	RMSCE	L2	MAE
NoLatent (ensemble, with σ)	0.0260	0.0258	0.0289	0.1670	0.1420
Latent (**ours**, ensemble, with σ)	**0.0101**	**0.0100**	**0.0124**	**0.1562**	**0.1296**

Table 3: Results of Teacher-Forcing Experiment

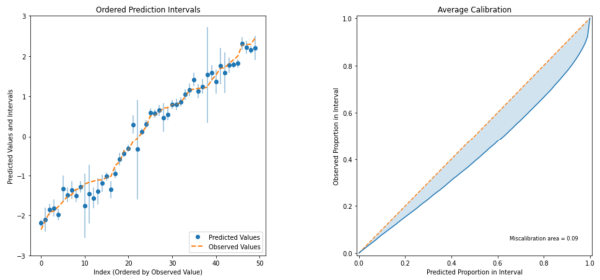

Figure 4: The ordered prediction intervals and average calibration of NoLatent (ensemble, with σ).

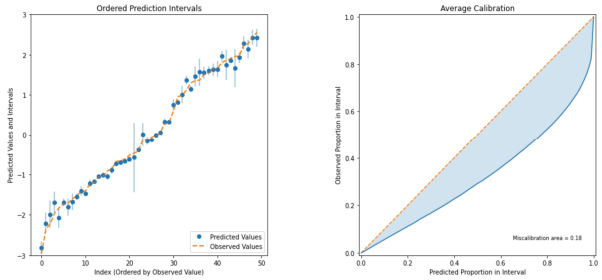

Figure 5: The ordered Prediction Intervals and Average Calibration Plot Analysis for Latent full + Deterministic case.

model's prediction as input to the next-step's prediction (used in Table 1), Teacher-forcing provides the ground-truth as input for each step, eliminating the propagation of uncertainty.

The final experimental results are presented in Table 2 and Table 3. In Table 2, we see that our model (Latent)'s miscalibration area (MA) of 0.0142 is significantly smaller than Nolatent (MA=0.0602) which does not have uncertainty propagation. There are two possible causes of this gap: uncertainty propagation enabled by our model and the slight difference between the two model architectures. If we perform Teacher-forcing (Table 3) which eliminates the effect of uncertainty propagation, the NoLatent MA reduces significantly to 0.0260. This means that the gap in Table 2 between Nolatent (MA=0.0602) and our model (MA=0.0142) is mostly due to the uncertainty propagation enabled by our model. With the latent evolution model g evolving both the latent vector and latent uncertainty vector $\mathbf{z}_\sigma^t$, our model is able to accurately account for the propagation of uncertainty.

Key Factors Influence. In this experiment, we primarily investigate the key factors: Deterministic, L1, and $\mathbf{z}_\sigma^t$, and their respective impacts on the uncertainty quantification performance of the latent evolution framework (Latent full) used in this paper. The specific experimental results are shown in Fig. 5, Fig. 6 and Fig. 7. We compared these three graphs with Fig. 3, but due to the subtle changes in the left graph, we primarily focused on observing the right graph. We see that the effects of Deterministic and $\mathbf{z}_\sigma^t$ on the algorithm in this study are more significant than L1. We have also presented the fluid simulation images corresponding to these three experiments in Appendix C.

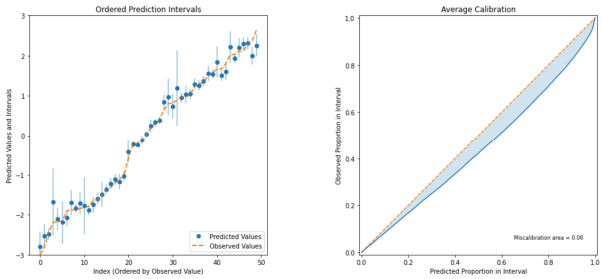

Figure 6: The ordered Prediction Intervals and Average Calibration Plot Analysis for Latent full + L1 case.

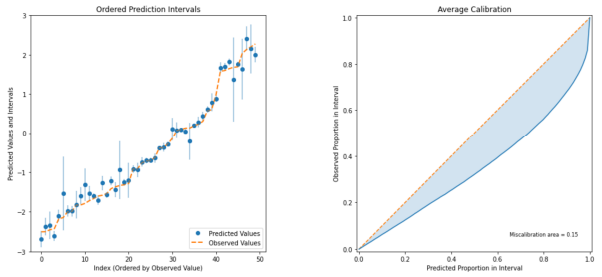

Figure 7: The ordered Prediction Intervals and Average Calibration Plot Analysis for Latent full + No $\mathbf{z}_\sigma^t$ case.

	MA	MACE	RMSCE	L2	MAE
NoLatent (ensemble, without σ)	0.0929	0.0920	0.1055	1.5255	1.2580
Latent (**ours**, ensemble,with σ)	**0.0224**	**0.0222**	**0.0264**	**0.1863**	**0.1505**

Table 4: Results of Inverse Optimization

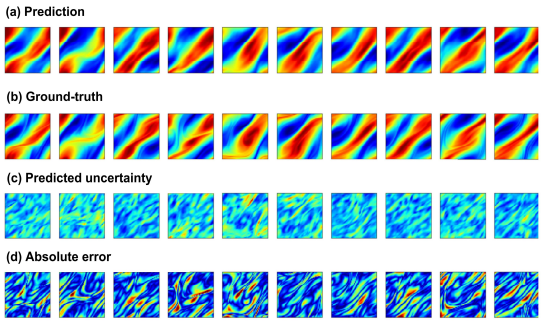

Figure 8: Visualization of LE-PDE-UQ predicting the dynamics of turbulent 2D Navier-Stokes in inverse optimization.

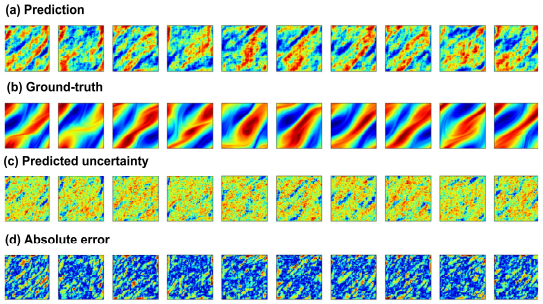

Figure 9: Visualization of NoLatent (ensemble, with σ) predicting the dynamics of turbulent 2D Navier-Stokes in inverse optimization.

Inverse Optimization

In this section, our main goal is to investigate question (3) through a comparison between Latent and NoLatent approaches, and the final results are shown in Table 4. The result shows that Nolatent has significantly larger error, and larger miscalibration error. Meanwhile, Fig. 8 and Fig. 10 respectively illustrate the inverse optimization capabilities of the Latent approach (our proposed algorithm), while Fig. 9 and Fig. 11 depict the inverse optimization abilities of the NoLatent approach. Comparing Fig. 8 vs. Fig. 9 and 10 with Fig. 11, we can intuitively observe that our proposed algorithm's inverse optimization predictions significantly outperform NoLatent. This is because optimizing w.r.t. the high-dimensional input space can easily find non-physical, adversarial models. In contrast, our model optimizes w.r.t. the much smaller latent dimension and then decodes back to the reasonable input space, thus achieving a much better error and uncertainty quantification.

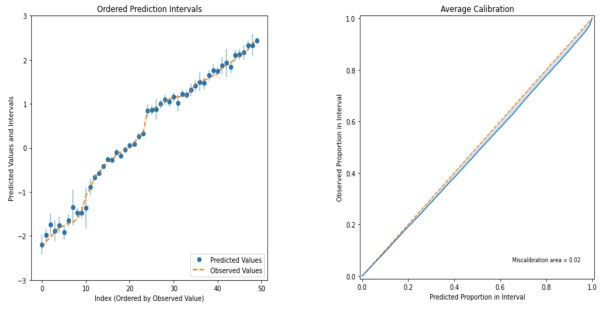

Figure 10: The ordered Prediction Intervals and Average Calibration Plot Analysis for the LE-PDE-UQ in inverse optimization.

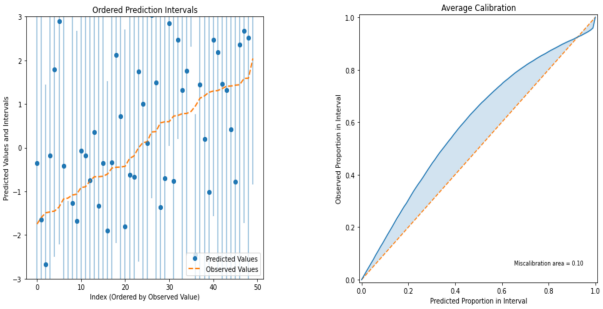

Figure 11: The ordered Prediction Intervals and Average Calibration Plot Analysis for the NoLatent (ensemble, with σ) case in inverse optimization.

Conclusion

In this work, we have introduced the LE-PDE-UQ framework to address the challenge of uncertainty quantification in time-dependent partial differential equations within deep learning-based surrogate models. Our method is driven by latent vectors within a dedicated latent space, enhancing the solving capabilities of both forward evolution and inverse optimization through accurate predictions and robust uncertainty estimates. Notably, LE-PDE-UQ can propagate uncertainty over extended auto-regressive rollouts without requiring additional sampling, providing a unique advantage in long-term predictions. Through rigorous experiments, our approach outperformed prominent baselines in uncertainty quantification. It demonstrated exceptional execution precision and stability in both forward and inverse scenarios, while bridging the gap between deep learning-based surrogate models and trustworthy uncertainty quantification.

Acknowledgements

We would like to express our sincere thanks to Wenhao Deng, Xianfeng Wu, Jiaxin Li and Long Wei in Westlake University AI for Scientific Simulation and Discovery Lab for discussions and for providing feedback on our manuscript. We also gratefully acknowledege the support of Westlake University Research Center for Industries of the Future and Westlake University Center for High-performance Computing. Jure Leskovec acknowledges the support of DARPA under Nos. N660011924033 (MCS); NSF under Nos. OAC-1835598 (CINES), CCF-1918940 (Expeditions), DMS-2327709 (IHBEM); Stanford Data Applications Initiative, Wu Tsai Neurosciences Institute, Chan Zuckerberg Initiative, Amazon, Genentech, GSK, Hitachi, Juniper Networks, KDDI, and UCB.

The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding entities.

References

Bernardo, J. M.; and Smith, A. F. 2009. *Bayesian theory*, volume 405. John Wiley & Sons.

Bi, K.; Xie, L.; Zhang, H.; Chen, X.; Gu, X.; and Tian, Q. 2023. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 1–6.

Biegler, L. T.; Ghattas, O.; Heinkenschloss, M.; and van Bloemen Waanders, B. 2003. Large-scale PDE-constrained optimization: an introduction. In *Large-Scale PDE-Constrained Optimization*, 3–13. Springer.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural network. In *International conference on machine learning*, 1613–1622. PMLR.

Brandstetter, J.; Worrall, D. E.; and Welling, M. 2022. Message Passing Neural PDE Solvers. In *International Conference on Learning Representations*.

Carpanese, F. 2021. Development of free-boundary equilibrium and transport solvers for simulation and real-time interpretation of tokamak experiments. Technical report, EPFL.

Chung, Y.; Char, I.; Guo, H.; Schneider, J.; and Neiswanger, W. 2021. Uncertainty Toolbox: an Open-Source Library for Assessing, Visualizing, and Improving Uncertainty Quantification. *arXiv preprint arXiv:2109.10254*.

Courant, R.; Friedrichs, K.; and Lewy, H. 1967. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2): 215–234.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059. PMLR.

Graves, A. 2011. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

Lelievre, T.; and Stoltz, G. 2016. Partial differential equations and stochastic methods in molecular dynamics. *Acta Numerica*, 25: 681–880.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Louizos, C.; and Welling, M. 2016. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International conference on machine learning*, 1708–1716. PMLR.

Lynch, P. 2008. The origins of computer weather prediction and climate modeling. *Journal of computational physics*, 227(7): 3431–3444.

Mackay, D. J. C. 1992. *Bayesian methods for adaptive models*. California Institute of Technology.

Neal, R. M. 2012. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Rasmussen, C. E.; and Quinonero-Candela, J. 2005. Healing the relevance vector machine through augmentation. In *Proceedings of the 22nd international conference on Machine learning*, 689–696.

Sircombe, N. J.; Arber, T.; and Dendy, R. 2006. Kinetic effects in laser-plasma coupling: Vlasov theory and computations. In *Journal de Physique IV (Proceedings)*, volume 133, 277–281. EDP sciences.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958.

Tran, D.; Dusenberry, M.; Van Der Wilk, M.; and Hafner, D. 2019. Bayesian layers: A module for neural network uncertainty. *Advances in neural information processing systems*, 32.

Wenzel, F.; Snoek, J.; Tran, D.; and Jenatton, R. 2020. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33: 6514–6527.

Winovich, N.; Ramani, K.; and Lin, G. 2019. ConvPDE-UQ: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains. *Journal of Computational Physics*, 394: 263–279.

Wu, T.; Maruyama, T.; and Leskovec, J. 2022. Learning to accelerate partial differential equations via latent global evolution. *Advances in Neural Information Processing Systems*, 35: 2240–2253.

Wu, T.; Maruyama, T.; Zhao, Q.; Wetzstein, G.; and Leskovec, J. 2022. Learning Controllable Adaptive Simulation for Multi-scale Physics. In *NeurIPS 2022 AI for Science: Progress and Promises*.

Wu, T.; Maruyama, T.; Zhao, Q.; Wetzstein, G.; and Leskovec, J. 2023. Learning Controllable Adaptive Simulation for Multi-resolution Physics. *arXiv preprint arXiv:2305.01122*.

Zhao, Q.; Lindell, D. B.; and Wetzstein, G. 2022. Learning to solve pde-constrained inverse problems with graph networks. *arXiv preprint arXiv:2206.00711*.

Zhu, Y.; Zabaras, N.; Koutsourelakis, P.-S.; and Perdikaris, P. 2019. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394: 56–81.