XLIV CILAMCE PORTO 2023

Model constrained empirical Bayesian neural networks for inverse problems

Russell S. Philley[1], Hai V. Nguyen[2], Tan Bui-Thanh[1,2]

[1]*Oden Institute for Computational Engineering & Sciences, The University of Texas at Austin*
201 E. 24th Street, C0200, 78712, Austin, Texas, United States of America
rsphilley@utexas.edu, tanbui@oden.utexas.edu
[2]*Dept. of Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin*
2617 Wichita Street, C0600, 78712, Austin, Texas, United States of America
hainguyen@utexas.edu, tanbui@oden.utexas.edu

Abstract. Principled Uncertainty quantification (UQ) in deep learning is still an unsolved problem. Numerous methods have been developed so far, with Bayesian neural networks (BNNs) as the popular approach. BNNs, while inherently UQ-enabled and resistant to over-fitting, suffer from unnatural and artificial priors over their parameters. This paper develops a model-constrained framework for quantifying the uncertainty in deep neural network inverse solutions. At the heart of our approach is an interpretable and physically-meaningful prior over neural network parameters trained through use of Stein variational gradient descent (SVGD). We provide comprehensive numerical results for a 2D inverse heat conductivity problem and a 2D inverse initial conditions problems for both the time-dependent Burgers' and Navier-Stokes equations.

Keywords: Deep learning, inverse problems, Bayesian methods, uncertainty quantification, variational inference

1 Introduction

Inverse problems play a significant role in various scientific and engineering endeavors, facilitating the fusion of measurements and data with models and physics, enabling us to uncover cause from effect. Many specific problems of interest in engineering and science are governed by partial differential equations (PDEs); as such, the solution of any associated inverse problems are also governed by PDEs (Tarantola [1], Kaipio and Somersalo [2]). Although the mathematical description of an inverse problem may be simple and elegant, it is important to note that finding solutions to these problems can be challenging due to their ill-posed nature, with conditions described by Hadamard [3]. Additionally, the computational methods used present their own set of complications, incurring significant costs when dealing with high-dimensional PDE parameters. This phenomenon, known as the curse of dimensionality, calls for careful consideration and resource management. Inverse problems for practical systems (Alifanov [4], Oliver et al. [5], Komatitsch et al. [6], Bui-Thanh et al. [7], Lefebvre et al. [8]) also display this same high-dimensional space challenge; this issue is compounded when adding uncertainty estimates to inverse solutions. Consequently, having solutions to PDE-constrained inverse problems is of great importance.

Deep learning is a subset of machine learning which focuses on deep neural networks (DNNs), these are neural networks which consist of more than one hidden layer (Goodfellow et al. [9], Nielsen [10]). DNNs have exploded in popularity for scientific applications in the past decade, seeing usage in numerous fields (Kojima et al. [11], White et al. [12], Pestourie et al. [13], Tahersima et al. [14], Peurifoy et al. [15], So et al. [16], Jiang et al. [17], Singh et al. [18], Goh et al. [19]). In addition to the pure data-driven approaches, there have been efforts to incorporate constraints from physics information into the training process with the aim of avoiding the overfitting issues associated with pure data-driven approaches, as well as allowing for the use of less data relative to pure data-driven approaches. Physics-informed neural networks (PINNs) make use of physics information by constraining the network training with the PDE residual (Raissi et al. [20], Karniadakis et al. [21], Pang et al. [22]). Other methods ([23, 24]) use networks to learn the unknown parameters via optimization constrained by physics equations. Pakravan et al. [25] uses autoencoders to learn the inverse map by minimizing the data misfit. Jin et al.

[26] account for both data misfit and regularization for seismic inversion. In the work by Nguyen and Bui-Thanh [27], a fully differentiable forward model is incorporated into the loss to learn the Tikhonov regularized inverse solver along with theoretical results for the method.

Inverse problems involve a multitude of inputs, including PDE parameters, initial & boundary conditions, empirical data, and constitutive models, among others. However, it is not always the case that all of these inputs may be available. There could be instances where boundary conditions are missing, where data is sparse, or even cases of model inadequacy. When faced with such situations, it is natural to question the reliability of our solution to the inverse problem. This leads us to uncertainty quantification (UQ), in which we seek to assess and quantify uncertainty in both the inputs and outputs of our simulations (Sullivan [28]). Through UQ we enable informed decision-making processes in their respective application domains.

UQ is often broken into Bayesian and non-Bayesian methods; Gaussian process regression (GPR)[29] is one such example of a Bayesian method, defining Gaussian priors over parameters before pushing the prior uncertainty measure forward to provide uncertainty for predictions. However, in practice GPR struggles with solving PDEs due to nonlinearities. Markov chain Monte Carlo (MCMC) methods are used for sampling from an arbitrary distribution and have been applied to UQ for PDE constrained inversion (Steins et al. [30], Bui-Thanh and Nguyen [31]), although MCMC methods are often intractable in high-dimensional parameter spaces due to a slow convergence rate. Polynomial chaos expansions described by Xiu and Karniadakis [32] are not inherently Bayesian, but have been used in Bayesian formulations in the work from Madankan et al. [33] and Shao et al. [34].

Empirical Bayes methods are those in which the prior distribution is influenced in some way by the data, from a desire to try and combine the strengths of both frequentist and Bayesian methods (Casella [35], Robbins [36], Wasserman [37]). Although they satisfy the most basic requirements to be called a Bayesian method, they are subject to great criticism on account of the "double-dipping" of data, as well as the violation of the Likelihood Principle, which states that all experimental data is incorporated in the likelihood distribution, although there are efforts to minimize these criticisms described by Darnieder [38]. Despite these critiques, empirical Bayes methods have found pragmatic use and adoption. Our proposed method, while empirical Bayes in nature, is focused on deep learning techniques.

Despite the widespread exploration of deep learning in the physical sciences, uncertainty quantification for deep learning remains an unsolved problem as evidenced by Abdar et al. [39], He and Jiang [40], Gawlikowski et al. [41], and Kabir et al. [42]. UQ techniques for deep learning can be broken down into multiple categories, but it is natural to think of methods as being for a single network or for an ensemble, and as being deterministic or probabilistic. Deep-UQ by Tripathy and Bilionis [43] is a method for a single deterministic network which aims to parameterize the structure of a DNN to recover a low-dimensional manifold in the input space to create surrogate models. SDE-Net by Kong et al. [44] is also for a deterministic single network, but it instead aims to combine the training of a DNN with the integration of an SDE to impose uncertainty on network predictions. Ensemble methods are numerous, but notable ones include usage of techniques at training-time like bagging and boosting given by Achrack et al. [45] along with techniques to reduce ensemble size, such as pruning (Cavalcanti et al. [46]) and distillation (Malinin et al. [47], Lindqvist et al. [48]). There are numerous kinds of probabilistic techniques, but among the most noteworthy are Bayesian neural networks (BNNs) (Neal [49], MacKay [50]). BNNs define a prior distribution over the network parameters, and are updated with problem data to reach some trained posterior distribution of network parameters given data, at which point prediction can be made via integration over the posterior distribution. Depending on the algorithm and integrating procedure, Bayesian neural networks can be single or ensemble-based. Due to the principled nature of the Bayesian approach, BNNs have been a popular choice for UQ in scientific deep learning. B-PINNs described by Yang et al. [51] are an extension of PINNs from deterministic networks to BNNs. Agata et al. [52] use a combination of BNNs and PINNs, but they also combine it with a velocity-space Stein Variational Gradient Descent (SVGD) for application to seismic tomography. Yang et al. [53] describes output-constrained BNNs, where the aim is to impose functional constraints about output feasibility onto the prior.

Variational inference is a technique for approximating the intractable integrals that appear in Bayesian inference (Bishop [54], Blei et al. [55], Bishop [56]). SVGD, described by Liu and Wang [57], is a deterministic, gradient-based sampling method for variational inference; with SVGD we define a desired target distribution and transport an ensemble of arbitrarily-sampled particles to approximate the target distribution. D'Angelo et al. [58] compared SVGD with other ensemble-based methods for neural networks and found that enhanced repulsion between ensemble members can improve approximation of the Bayesian posterior. In the work by Hu et al. [59], SVGD was tested with BNNs on prediction of cyclical time-series data. Similar to Agata et al. [52], Sun and Wang [60] used a combination of BNNs, PINNs, and SVGD to reconstruct fluid flow. In the work by Geneva and Zabaras [61], UQ for Reynolds-averaged turbulence modeling was performed through a combination of BNNs and SVGD.

Although published works such as (Yang et al. [51, 53], Sun and Wang [60]) have attempted to fuse BNNs with physics information, and work such as that of Agata et al. [52] has gone further and used SVGD to try and define a physically-interpretable Bayesian prior, no work has yet developed an empirical Bayesian UQ-enabled, interpretable deep-learning based inverse solver with a physically intuitive prior.

In this paper we introduce a novel algorithm that provides UQ capabilities for model-constrained empirical Bayesian neural networks for inverse problems. Our formulation provides a framework for eliciting an interpretable and physically-meaningful prior over neural network parameters. Furthermore, our variational inference formulation allows us to train the BNN even if network parameters are not initially sampled from the Bayesian prior, allowing us to maintain the interpretability of the Bayesian formulation while circumventing the difficulties involved with initializing parameters from complicated Bayesian priors.

2 Results:

We propose new methods that are analogues of the naive DNN (`NDNN`), the model-constrained DNN (`MCDNN`), and the `TNet` method described in our previous work from Nguyen and Bui-Thanh [27]. Respectively, we refer to our UQ extensions as `NBNN`, `MCBNN`, and `TBNN`. Although `NBNN` is just a naive BNN and thus is not a novel method, `MCBNN` and `TBNN` are our novel extensions of `MCDNN` and `TNet`. We provide results for three different problems of interest: Poisson's equation, Burgers' equation, and the Navier-Stokes equations. We test performance between `NBNN`, `MCBNN`, `TBNN`, and the Tikhonov inverse solution. Shared training parameters for all problems are listed in Table 1; most values are chosen from our previous work shown in Nguyen and Bui-Thanh [27].

Table 1. Summary of shared training parameters for `NBNN`, `MCBNN` and `TBNN` for nonlinear inverse problems in section 2.1, 2.2 and 2.3.

Network	Architecture	1 layer with 5000 neurons
	Activation function	ReLU
	Weight initializer	$\mathcal{N}(0, 0.02)$
	Bias initializer	$\mathbf{0}$
Training	Optimizer	`ADAM`
	Learning rate	$1e-3$
Test data		500 samples (drawn independently)
Precision		Double precision

We do not perform batching and instead train with all samples in one batch as all training set sizes are relatively small. We provide additive noise to our observations $\boldsymbol{y}$ via samples from $\mathcal{N}(0, \delta \max \boldsymbol{y})$, where δ is a problem-specific relative noise level. Optimal regularization parameters are selected on a problem-dependent basis for `NBNN`, `MCBNN`, `TBNN`, and the Tikhonov solution from their deterministic analogues per our previous experiments shown in Nguyen and Bui-Thanh [27].

To measure performance on the test set, we calculate relative error, mean absolute error, and mean standard deviation. We calculate relative error for the inverse solution on the test set as follows,

$$\text{RErr} = \frac{1}{n_s} \sum_{i=1}^{n_s} \frac{||\hat{\boldsymbol{u}}^i - \boldsymbol{u}^{\text{true}}||^2}{||\boldsymbol{u}^{\text{true}}||^2}. \tag{1}$$

We compute mean absolute error on the test set similarly,

$$\text{MAErr} = \frac{1}{n_s} \sum_{i=1}^{n_s} \frac{1}{m} ||\hat{\boldsymbol{u}}^i - \boldsymbol{u}^{\text{true}}||^2. \tag{2}$$

And we calculate the mean standard deviation of the inverse solution over all samples in the test set as follows:

$$\sigma = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\mathbf{E}\left((\hat{\boldsymbol{u}}^i)^2\right) - \left(\mathbf{E}(\hat{\boldsymbol{u}}^i)\right)^2} \tag{3}$$

When plotting error or standard deviation across the problem domain, we compute the respective metric on a pointwise basis.

2.1 Poisson Equation:

We consider the following equation,

$$\begin{aligned} -\nabla \cdot \left(e^{\omega} \nabla y\right) &= 20 \quad \text{in } \Omega = (0,1)^2, \\ y &= 0 \quad \text{on } \Gamma^{\text{ext}}, \\ \mathbf{n} \cdot \left(e^{\omega} \nabla y\right) &= 0 \quad \text{on } \Gamma^{\text{root}}, \end{aligned} \tag{4}$$

where ω is the conductivity field, y is the temperature field, and **n** is the unit outward normal vector on Neumann boundary part Γ^{root}. Figure 1 shows the domain (left subfigure) and a 16×16 mesh (right subfigure) together with the locations of 10 observational points of the state y. In this problem, we are interested in reconstructing the conductivity field given a set of 10 pointwise observations; observation locations were chosen at random.

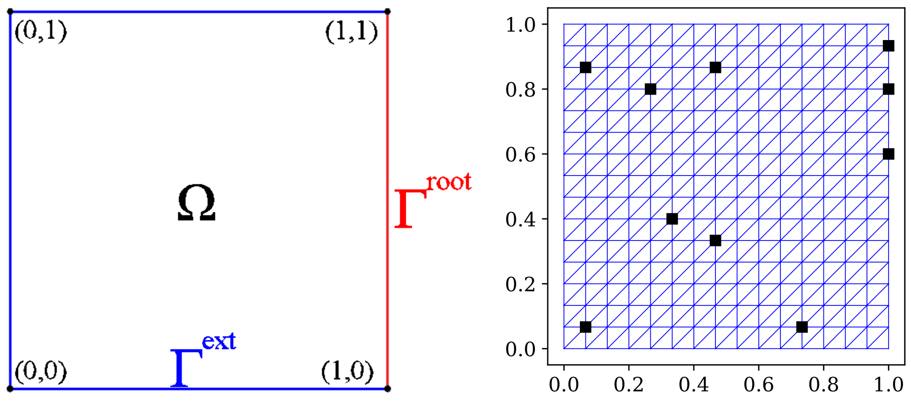

Figure 1. **2D heat conductivity inverse problem.** *Left*: the domain and the boundaries; *Right*: A 16×16 finite element mesh with 10 observational locations.

Construction of training and testing data sets. We start with drawing the parameter conductivity samples via a truncated Karhunen-Loève expansion

$$\omega(x) = \sum_{i=1}^{n} \sqrt{\lambda_i} \phi_i(x) u_i, \quad x \in [0,1]^2, \tag{5}$$

where (λ_i, ϕ_i) are the eigenpairs of the following two-point correlation function given by Constantine et al. [62]:

$$\mathcal{C}(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|_1}{\beta}\right) \tag{6}$$

where $\|\cdot\|_1$ is the 1-norm on $\mathcal{R}^2$ and $\beta = 0.02$ is the correlation length. Here, $\boldsymbol{u} = (u_i)_{i=1}^n \sim \mathcal{N}(0, \mathbf{I})$ is a standard Gaussian random vector. Instead of directly inverting for the physical parameter ω, we reconstruct the coefficient vector $\boldsymbol{u}$. Specifically, we select $n = 15$ eigenvectors corresponding to the first 15 largest eigenvalues. For each sample, we discretize ω and then solve the heat equation for the temperature $\boldsymbol{y}$ by the finite element method. Observations of the temperature field are taken at 10 observational points, which are then corrupted with additive Gaussian noise with a relative noise level δ. We generate test pairs $(\omega, \boldsymbol{y})$ using the same process.

We look at multiple hyperparameters to test performance. First, in case (A), we test performance with varying training set sizes n_t and observational noise magnitudes δ. In case (B), we test performance with varying lengths of the warm-start to intelligently sample initial network parameters. In case (C), we test performance with varying ensemble sizes r.

Case (A): We construct the training set to have $n_t = \{50, 100, 200\}$ samples, the magnitude of additive noise to be $\delta = \{0\%, 0.5\%, 1\%, 2\%\}$, and the SVGD ensemble size to be $r = 10$. For each training set $\mathcal{T}_{n_t}$ with its size denoted by n_t, we select the elements of the training set such that $\mathcal{T}_{50} \subset \mathcal{T}_{100} \subset \mathcal{T}_{200}$. Each simulation was executed eight times with different seeds and the results averaged to get a reliable measure of performance.

Table 2. Minimum relative error (%) and corresponding standard deviation of predictions at $\delta = 0.5\%$ for all methods. Left column is error, right column is standard deviation

	`NBNN`		`MCBNN`		`TBNN`		Tikhonov	
$n_t = 50$	61.52	0.0497	56.60	0.0051	46.44	0.0026	45.38	0.0022
$n_t = 100$	54.16	0.0519	50.57	0.0046	45.81	0.0030		
$n_t = 200$	50.44	0.0438	48.31	0.0041	45.61	0.0028		

Table 3. Minimum relative error (%) and corresponding standard deviation for `TBNN` and the Tikhonov solution. Left column is error, right column is standard deviation.

	$\delta = 0\%$		$\delta = 0.5\%$		$\delta = 1\%$		$\delta = 2\%$	
`TBNN`, $n_t = 50$	43.05	0.0025	46.44	0.0026	59.76	0.0244	79.36	0.0038
`TBNN`, $n_t = 100$	40.82	0.0031	45.81	0.0030	60.16	0.0037	78.50	0.0037
`TBNN`, $n_t = 200$	39.90	0.0028	45.61	0.0026	59.92	0.0036	77.18	0.0037
Tikhonov, $n_s = 500$	38.71	0.0021	45.38	0.0021	62.52	0.0022	77.00	0.0026

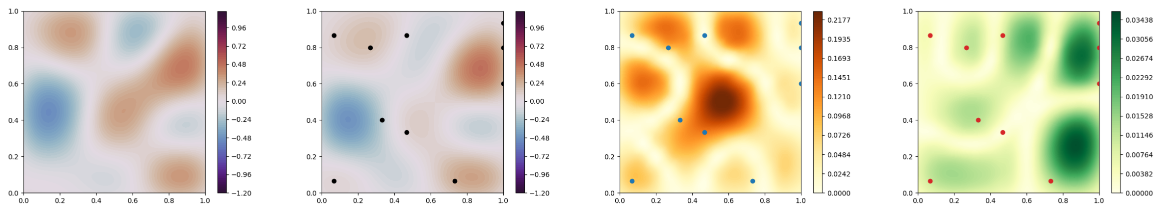

Figure 2. Plots of `TBNN` on a test set sample, dots are observation locations. Left: true solution, center left: predicted solution, center right: absolute error, right: standard deviation.

Table 2 shows that `TBNN` outperforms `NBNN` and `MCBNN` in terms of both accuracy and certainty. It is also noteworthy that `TBNN` performs well with small n_t, in comparison to `NBNN` and `MCBNN` which struggle. Furthermore, observe the values of standard deviation; considering that values for the inverse solution loosely range from $(-1, 1)$, this means that a standard deviation of 0.003 is roughly within the relative noise level of $\delta = 0.5\%$. Put another way, this means that `TBNN` and the Tikhonov solution achieved uncertainty measures roughly comparable to the degree of noise. `MCBNN` also performs well, but is still outperformed by `TBNN`. `NBNN` performs poorly on the uncertainty measure, with standard deviation values roughly 20 to 30 times those of the Tikhonov solution.

Table 3 demonstrates `TBNN` with varying degrees of relative noise; as is expected, relative error and standard deviation increase with increasing noise. Figure 2 illustrates the predictions made by `TBNN`.

Case (B): We construct the training set to only have $n_t = \{200\}$ samples, the magnitude of additive noise to be $\delta = \{0.5\%\}$, and the SVGD ensemble size to be $r = 10$. To test how the algorithm performs with different particle initializations, we apply a warm-start of varying lengths before training; namely, we test with warm-starts of $\{0, 10000, 20000\}$ epochs. Each simulation was executed four times with different seeds and the results averaged to get a reliable measure of performance.

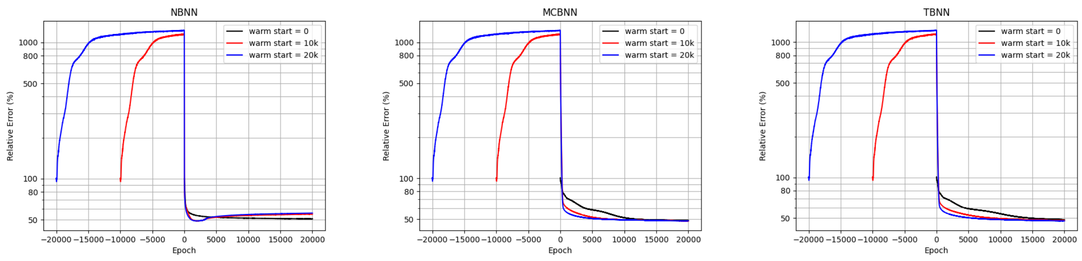

Figure 3. Relative error during training with a warm-start. Negative epochs indicate $\rho_m(\boldsymbol{\theta})$ is the target distribution, positive epochs indicate $\rho(\boldsymbol{\theta}|\mathcal{T})$ is the target distribution. Left: `NBNN`, center: `MCBNN`, right: `TBNN`.

Although surprising, Fig. 3 demonstrates that there is no benefit to the use of a warm-start in practice. Numerical values were omitted in the interest of space, but asymptotically `TBNN` and `MCBNN` performed the same regardless of the warm-start duration; `NBNN` actually sees degraded performance from the warm-start. However, the warm-start does have interesting behavior. First, it raises the relative error on the test set when trying to sample particles from $\rho_m(\boldsymbol{\theta})$. But as soon as the warm-start is finished and the target distribution is set to be $\rho(\boldsymbol{\theta}|\mathcal{T})$, the test error sees a dramatic decrease. More intriguing is the fact that having a warm-start makes the BNN learn faster than a BNN with particles sampled from a Gaussian. We believe that this provides evidence that the model-informed prior $\rho_m(\boldsymbol{\theta})$ is a physically-intuitive choice for the BNN parameter space.

Case (C): We construct the training set to only have $n_t = \{200\}$ samples, and we set the magnitude of additive noise to be $\delta = \{0.5\%\}$. To investigate how SVGD ensemble size affects algorithm performance, we tested with ensemble sizes of $r = \{10, 50, 100\}$. Each simulation was executed four times with different seeds and the results averaged to get a reliable measure of performance.

Table 4. Minimum relative error (%) and corresponding standard deviation of predictions for all methods. Left column is error, right column is standard deviation

	`NBNN`		`MCBNN`		`TBNN`		Tikhonov	
$r = 10$	50.23	0.0444	48.17	0.0041	45.48	0.0025	45.25	0.0021
$r = 50$	49.75	0.0311	48.00	0.0047	45.52	0.0043	45.27	0.017
$r = 100$	49.43	0.0151	47.98	0.0074	45.51	0.0051	45.30	0.0022

Table 4 demonstrates ensemble size r does not have a particularly strong effect on test error. This is expected, as SVGD with $r = 1$ will reduce to simple gradient ascent for maximum a posteriori estimation. We also observe that uncertainty for `TBNN`, `MCBNN`, and the Tikhonov solution are relatively unaffected. `NBNN` does see reduced uncertainty with a larger ensemble size r, but this comes at the cost of being an un-interpretable method. Furthermore, while `NBNN` reduced uncertainty, we mention that the terminal performance of `NBNN` degraded with increasing ensemble size. This may be caused by an inability for a simple L2 regularization to handle a large parameter space, in comparison to the informative regularization provided by the model-informed prior.

2.2 Burgers' Equation:

We consider the following viscous 2D Burger's equations

$$\begin{aligned}
\frac{\partial \omega}{\partial t} + \omega\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} &= \nu\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) \qquad x, y \in (0,1), t \in (0, 0.5], \\
\frac{\partial v}{\partial t} + \omega\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \qquad x, y \in (0,1), t \in (0, 0.5],
\end{aligned} \tag{7}$$

subject to periodic boundary conditions, initial velocity components $v(x, y, 0) = v_0(x, y) = 1$, $\omega(x, y, 0) = \omega_0(x, y)$, and viscosity coefficient $\nu = 10^{-2}$. The spatial domain $(0, 1) \times (0, 1)$ is discretized with $n_x = 32$ and $n_y = 32$ mesh points in x and y directions, respectively, while the temporal domain $(0, 0.5)$ is subdivided into 201 time steps (including the initial time step $t = 0$). In this problem, the goal is to invert for the initial x-velocity ω_0

from 20 pointwise values of the vorticity $\omega_{0.5}$ (see the definition below) at the final time $T = 0.5$. Observation locations were chosen such that they cross the domain.

Construction of train and test data sets. To generate training data for learning the inverse map, we draw periodic samples of $\omega(x, y, 0)$ using a truncated Karhunen-Loève expansion

$$\omega(x, y, 0) = \exp\left(\sum_{i=1}^{24} \sqrt{\lambda_i}\, \phi_i(x, y)\, u_i\right), \tag{8}$$

where $\mathbf{u} = \{u_i\}_{i=1}^{24} \sim \mathcal{N}(0, \mathbf{I})$, and (λ_i, ϕ_i) are eigenpairs of the covariance $7^{\frac{3}{2}}\left(-\Delta + 49\mathbf{I}\right)^{-2.5}$ with periodic boundary conditions.

Next, we discretize an initial vorticity $\omega(x, y, 0)$, denoted as ω_0, and we solve the Burgers' equations via a finite difference method to compute the discrete x-velocity component at the final time: $\omega_{0.5}$. Finally, the values of $\omega_{0.5}$ at 20 locations are extracted and corrupted with an additive white noise ($\delta = 2\%$) to form $\boldsymbol{y}^{obs}$. We construct the training set to have $n_t = \{50, 500\}$ samples, and we select the elements of the training set such that $\mathcal{T}_{50} \subset \mathcal{T}_{500}$. We use $r = 50$ particles in our SVGD ensemble. Each simulation was executed three times with different seeds and the results averaged to get a reliable measure of performance.

Table 5. Minimum relative error (%) and standard deviation of predictions for all methods. Left column is error, right column is standard deviation

	`NBNN`		`MCBNN`		`TBNN`		Tikhonov	
$n_t = 50$	2.24	0.0110	2.07	0.0102	1.86	0.0064	0.79	0.0014
$n_t = 500$	1.31	0.0197	1.24	0.0312	1.13	0.0127		

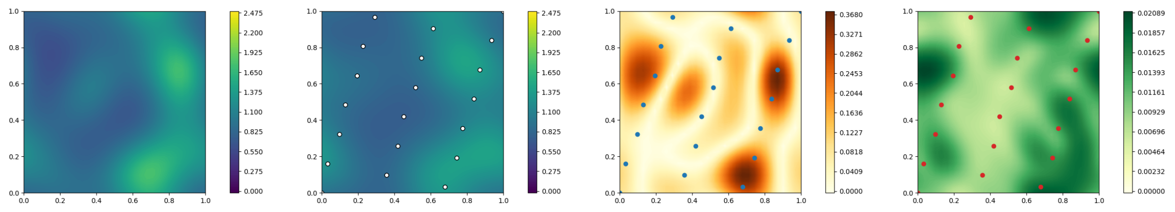

Figure 4. Plots of `TBNN` on a test set sample, dots are observation locations. Left: true initial condition, center left: predicted initial condition, center right: absolute error, right: standard deviation.

Table 5 provides performance metrics for the methods on Burgers' equation. We see that `TBNN` outperforms `MCBNN` and `NBNN` in both test error and standard deviation, Tikhonov outperforms all 3 methods together. We also observe an increase in standard deviation with increasing training set size n_t; we posit that it is because the larger amount of data allows the BNN to better approximate the posterior $\rho(\boldsymbol{\theta}|\mathcal{T})$.

2.3 Navier-Stokes Equations:

The vorticity form of 2D Navier-Stokes equation for a viscous and incompressible fluid given by Li et al. [63] is written as

$$\begin{aligned} \partial_t \omega(\boldsymbol{x}, t) + \boldsymbol{v}(\boldsymbol{x}, t) \cdot \nabla \omega(\boldsymbol{x}, t) &= \nu \Delta \omega(\boldsymbol{x}, t) + f(\boldsymbol{x}), && \boldsymbol{x} \in (0,1)^2, t \in (0, T] \\ \nabla \cdot \boldsymbol{v}(\boldsymbol{x}, t) &= 0, && \boldsymbol{x} \in (0,1)^2, t \in (0, T] \\ \omega(\boldsymbol{x}, 0) &= \omega_0(\boldsymbol{x}), && \boldsymbol{x} \in (0,1)^2 \end{aligned} \tag{9}$$

where $\boldsymbol{v} \in (0,1)^2 \times (0, T]$ is the velocity field, $\omega = \nabla \times \boldsymbol{v}$ is the vorticity, ω_0 is the initial vorticity, $f(\boldsymbol{x}) = 0.1\left(\sin\left(2\pi\left(x_1 + x_2\right)\right) + \cos\left(2\pi\left(x_1 + x_2\right)\right)\right)$ is the forcing function, and $\nu = 10^{-3}$ is the viscosity coefficient. The spatial domain is discretized with a 32×32 uniform mesh, while the time horizon $t \in (0, 10)$ is subdivided into 1000 time steps with $\Delta t = 10^{-2}$. We target to reconstruct the initial vorticity ω_0 from the measurements of vorticity at 20 observed points at the final time $T = 10$. Observation locations were chosen such that they cross the domain.

Construction of training and testing data sets. To generate data pairs of $(\omega, \boldsymbol{y})$, we draw samples of $\omega(\boldsymbol{x}, 0)$ using the truncated Karhunen-Loève expansion

$$\omega(\boldsymbol{x}, 0) = \sum_{i=1}^{24} \sqrt{\lambda_i}\, \phi_{\mathbf{i}}(\boldsymbol{x})\, u_i, \tag{10}$$

where $u_i \sim \mathcal{N}(0, 1), i = 1, \ldots, 24$, thus $\boldsymbol{u}_0 = \mathbf{0}, \Gamma = \mathbf{I}$, and (λ_i, ϕ_i) are eigenpairs obtained by the eigendecomposition of the covariance operator $7^{\frac{3}{2}}\left(-\Delta + 49\mathbf{I}\right)^{-2.5}$ with periodic boundary conditions. Next, we discretize an initial vorticity $\omega(\boldsymbol{x}, 0)$, denoted as ω_0, and we solve the Navier-Stokes equation by the stream-function formulation with a pseudospectral method shown in Li et al. [63] to obtain a discrete representation ω_t of $\omega(\boldsymbol{x}, t)$ at any time t.

The observation operator is imposed on solution ω_{10} to form the synthetic observables $\boldsymbol{y}$, then a realization of additive white noise with $\delta = 2\%$ is added to generate a noise-corrupted $\boldsymbol{y}$ sample. We construct the training set to have $n_t = \{50, 500\}$ samples, and we select the elements of the training set such that $\mathcal{T}_{50} \subset \mathcal{T}_{500}$. We use $r = 50$ particles in our SVGD ensemble. Each simulation was executed three times with different seeds and the results averaged to get a reliable measure of performance. We document results for all algorithms in Fig. 5, and we provide tabulated results in Table 6

Table 6. Minimum relative error (%) and corresponding standard deviation of predictions for all methods. Left column is error, right column is standard deviation

	NBNN		MCBNN		TBNN		Tikhonov	
$n_t = 50$	61.78	0.0184	54.12	0.0193	32.54	0.0078	21.89	0.0067
$n_t = 500$	36.34	0.0231	27.00	0.0133	24.83	0.0058		

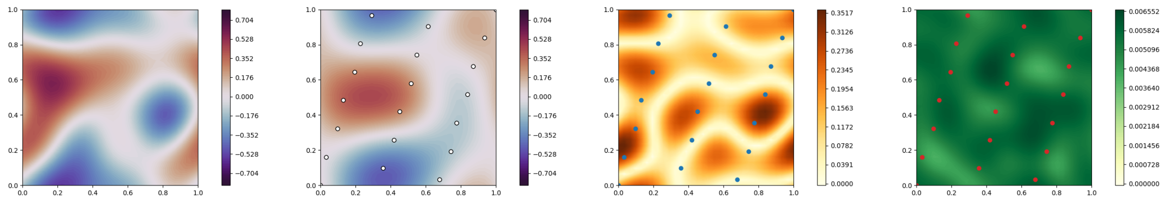

Figure 5. Plots of TBNN on a test set sample, dots are observation locations. Left: true initial condition, center left: predicted initial condition, center right: absolute error, right: standard deviation.

Table 6 illustrates that TBNN outperforms MCBNN and NBNN in both test error and standard deviation yet again. We see that TBNN comes the closest to achieving a test error that is comparable with the Tikhonov solution, and it also achieves a standard deviation comparable with the Tikhonov solution.

3 Conclusions

We argue that TBNN and MCBNN are an effective technique for enabling uncertainty quantification in deep learning for inverse problems. Despite the fact that we use empirical Bayes, we believe that our formulation allows us to elicit a physically interpretable prior on the BNN parameter space. Furthermore, we believe that the increased rate of learning with particles sampled from the model-informed prior $\rho_m(\boldsymbol{\theta})$ is evidence that the framework is interpretable. We have provided numerous numerical experiments demonstrating that TBNN can achieve test error and standard deviations mimicking that of a Tikhonov solver.

Acknowledgements. This research is partially funded by the National Science Foundation awards NSF-OAC-2212442, NSF-2108320, NSF-1808576 and NSF-CAREER-1845799; by the Department of Energy award DE-SC0018147, DE-SC0022211, and DE-SC00275836; and by an UT-Portugal award. The authors also acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC, visualization, database, or grid resources that have contributed to the research results reported within this paper. URL: http://www.tacc.utexas.edu

3.1 Permission

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2005.

[2] J. P. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*, volume 160 of *Applied Mathematical Sciences*. Springer, New York, NY, 2005.

[3] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pp. 49–52. Publisher: Princeton University, 1902.

[4] O. M. Alifanov. *Inverse Heat Transfer Problems*. International Series in Heat and Mass Transfer. Springer, Berlin, Heidelberg, 1994.

[5] D. S. Oliver, A. C. Reynolds, and N. Liu. *Inverse theory for petroleum reservoir characterization and history matching*. Cambridge University Press, 2008.

[6] D. Komatitsch, J. Ritsema, and J. Tromp. The spectral-element method, Beowulf computing, and global seismology. *Science (New York, N.Y.)*, vol. 298, pp. 1737–1742, 2002.

[7] T. Bui-Thanh, C. Burstedde, O. Ghattas, J. Martin, G. Stadler, and L. C. Wilcox. Extreme-scale UQ for Bayesian inverse problems governed by PDEs. In *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–11. ISSN: 2167-4337, 2012.

[8] M. Lefebvre, E. Bozda, H. Calandra, J. Hill, W. Lei, D. Peter, N. Podhorszki, D. Pugmire, H. Rusmanugroho, J. Smith, and J. Tromp. A data centric view of large-scale seismic imaging workflows. *Supercomputing (SC) 13*, 2013.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[10] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[11] K. Kojima, B. Wang, U. Kamilov, T. Koike-Akino, and K. Parsons. Acceleration of fdtd-based inverse design using a neural network approach. In *Advanced Photonics 2017 (IPR, NOMA, Sensors, Networks, SPPCom, PS)*, pp. ITu1A.4. Optica Publishing Group, 2017.

[12] D. A. White, W. J. Arrighi, J. Kudo, and S. E. Watts. Multiscale topology optimization using neural network surrogate models. *Computer Methods in Applied Mechanics and Engineering*, vol. 346, pp. 1118–1135, 2019.

[13] R. Pestourie, Y. Mroueh, T. V. Nguyen, P. Das, and S. G. Johnson. Active learning of deep surrogates for PDEs: application to metasurface design. *npj Computational Materials*, vol. 6, n. 1, pp. 1–7. Number: 1 Publisher: Nature Publishing Group, 2020.

[14] M. H. Tahersima, K. Kojima, T. Koike-Akino, D. Jha, B. Wang, C. Lin, and K. Parsons. Deep Neural Network Inverse Design of Integrated Photonic Power Splitters. *Scientific Reports*, vol. 9, n. 1, pp. 1368. Number: 1 Publisher: Nature Publishing Group, 2019.

[15] J. Peurifoy, Y. Shen, L. Jing, Y. Yang, F. Cano-Renteria, B. G. DeLacy, J. D. Joannopoulos, M. Tegmark, and M. Soljačić. Nanophotonic particle simulation and inverse design using artificial neural networks. *Science Advances*, vol. 4, n. 6, pp. eaar4206. Publisher: American Association for the Advancement of Science, 2018.

[16] S. So, T. Badloe, J. Noh, J. Bravo-Abad, and J. Rho. Deep learning enabled inverse design in nanophotonics. *Nanophotonics*, vol. 9, n. 5, pp. 1041–1057. Publisher: De Gruyter, 2020.

[17] J. Jiang, M. Chen, and J. A. Fan. Deep neural networks for the evaluation and design of photonic devices. *Nature Reviews Materials*, vol. 6, n. 8, pp. 679–700. Number: 8 Publisher: Nature Publishing Group, 2021.

[18] A. P. Singh, S. Medida, and K. Duraisamy. Machine-Learning-Augmented Predictive Modeling of Turbulent Separated Flows over Airfoils. *AIAA Journal*, vol. 55, n. 7, pp. 2215–2227. Publisher: American Institute of Aeronautics and Astronautics, 2017.

[19] H. Goh, S. Sheriffdeen, J. Wittmer, and T. Bui-Thanh. Solving Bayesian Inverse Problems via Variational Autoencoders. arXiv:1912.04212 [cs, eess, stat] version: 9, 2021.

[20] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. arXiv:1711.10566 [cs, math, stat], 2017.

[21] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, vol. 3, n. 6, pp. 422–440. Number: 6 Publisher: Nature Publishing Group, 2021.

[22] G. Pang, L. Lu, and G. E. Karniadakis. fPINNs: Fractional Physics-Informed Neural Networks. *arXiv:1811.08967 [physics]*. arXiv: 1811.08967, 2018.

[23] T. Fan, K. Xu, J. Pathak, and E. Darve. Solving Inverse Problems in Steady-State Navier-Stokes Equations using Deep Neural Networks. arXiv:2008.13074 [cs, math], 2020.

[24] J. Berg and K. Nyström. Neural network augmented inverse problems for PDEs. arXiv:1712.09685 [math, stat], 2018.

[25] S. Pakravan, P. A. Mistani, M. A. Aragon-Calvo, and F. Gibou. Solving inverse-PDE problems with physics-aware neural networks. *Journal of Computational Physics*, vol. 440, pp. 110414. arXiv:2001.03608 [physics], 2021.

[26] Y. Jin, Q. Shen, X. Wu, J. Chen, and Y. Huang. A Physics-Driven Deep-Learning Network for Solving Nonlinear Inverse Problems. *Petrophysics - The SPWLA Journal of Formation Evaluation and Reservoir Description*, vol. 61, n. 01, pp. 86–98, 2020.

[27] H. V. Nguyen and T. Bui-Thanh. TNet: A Model-Constrained Tikhonov Network Approach for Inverse Problems. arXiv:2105.12033 [cs, math, stat], 2022.

[28] T. Sullivan. *Introduction to Uncertainty Quantification*, volume 63 of *Texts in Applied Mathematics*. Springer International Publishing, Cham, 2015.

[29] J. Wang. An Intuitive Tutorial to Gaussian Processes Regression. arXiv:2009.10862 [cs, stat], 2022.

[30] E. Steins, T. Bui-Thanh, M. Herty, and S. Müller. Probabilistic constrained Bayesian inversion for transpiration cooling. *International Journal for Numerical Methods in Fluids*, vol. 94, n. 12, pp. 2020–2039, 2022.

[31] T. Bui-Thanh and Q. P. Nguyen. FEM-based discretization-invariant MCMC methods for PDE-constrained Bayesian inverse problems. *Inverse Problems and Imaging*, vol. 10, n. 4, pp. 943–975. Publisher: Inverse Problems and Imaging, 2016.

[32] D. Xiu and G. E. Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, vol. 24, n. 2, pp. 619–644. Publisher: Society for Industrial and Applied Mathematics, 2002.

[33] R. Madankan, P. Singla, T. Singh, and P. D. Scott. Polynomial-Chaos-Based Bayesian Approach for State and Parameter Estimations. *Journal of Guidance, Control, and Dynamics*, vol. 36, n. 4, pp. 1058–1074, 2013.

[34] Q. Shao, A. Younes, M. Fahs, and T. A. Mara. Bayesian sparse polynomial chaos expansion for global sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering*, vol. 318, pp. 474–496, 2017.

[35] G. Casella. Illustrating empirical Bayes methods. *Chemometrics and Intelligent Laboratory Systems*, vol. 16, n. 2, pp. 107–125, 1992.

[36] H. E. Robbins. An Empirical Bayes Approach to Statistics. In S. Kotz and N. L. Johnson, eds, *Breakthroughs in Statistics: Foundations and Basic Theory*, Springer Series in Statistics, pp. 388–394. Springer, New York, NY, 1992.

[37] L. Wasserman. Asymptotic inference for mixture models by using data-dependent priors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 62, n. 1, pp. 159–180, 2000.

[38] W. F. Darnieder. *Bayesian Methods for Data-Dependent Priors*. PhD thesis, The Ohio State University, 2011.

[39] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, vol. 76, pp. 243–297, 2021.

[40] W. He and Z. Jiang. A Survey on Uncertainty Quantification Methods for Deep Neural Networks: An Uncertainty Source Perspective. arXiv:2302.13425 [cs, stat], 2023.

[41] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu. A Survey of Uncertainty in Deep Neural Networks. arXiv:2107.03342 [cs, stat], 2022.

[42] H. M. D. Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi. Neural Network-Based Uncertainty Quantification: A Survey of Methodologies and Applications. *IEEE Access*, vol. 6, pp. 36218–36234. Conference Name: IEEE Access, 2018.

[43] R. K. Tripathy and I. Bilionis. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, vol. 375, pp. 565–588, 2018.

[44] L. Kong, J. Sun, and C. Zhang. SDE-Net: Equipping Deep Neural Networks with Uncertainty Estimates. arXiv:2008.10546 [cs, stat], 2020.

[45] O. Achrack, R. Kellerman, and O. Barzilay. Multi-Loss Sub-Ensembles for Accurate Classification with Uncertainty Estimation. arXiv:2010.01917 [cs, stat], 2020.

[46] G. D. C. Cavalcanti, L. S. Oliveira, T. J. M. Moura, and G. V. Carvalho. Combining diversity measures for ensemble pruning. *Pattern Recognition Letters*, vol. 74, pp. 38–45, 2016.

[47] A. Malinin, B. Mlodozeniec, and M. Gales. Ensemble Distribution Distillation, 2019.

[48] J. Lindqvist, A. Olmin, F. Lindsten, and L. Svensson. A General Framework for Ensemble Distribution Distillation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. ISSN: 1551-2541, 2020.

[49] R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, New York, NY, 1996.

[50] D. J. MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, vol. 4, n. 3, pp. 448–472. Publisher: MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 1992.

[51] L. Yang, X. Meng, and G. E. Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, vol. 425, pp. 109913, 2021.

[52] R. Agata, K. Shiraishi, and G. Fujie. Bayesian seismic tomography based on velocity-space Stein variation gradient descent for physics-informed neural network. arXiv:2301.07901 [physics], 2023.

[53] W. Yang, L. Lorch, M. A. Graule, S. Srinivasan, A. Suresh, J. Yao, M. F. Pradier, and F. Doshi-Velez. Output-Constrained Bayesian Neural Networks. arXiv:1905.06287 [cs, stat], 2019.

[54] C. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 1 edition, 2006.

[55] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, vol. 112, n. 518, pp. 859–877. arXiv:1601.00670 [cs, stat], 2017.

[56] C. M. Bishop. Variational learning in graphical models and neural networks. In L. Niklasson, M. Bodén, and T. Ziemke, eds, *ICANN 98*, pp. 13–22, London. Springer London, 1998.

[57] Q. Liu and D. Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[58] F. D'Angelo, V. Fortuin, and F. Wenzel. On Stein Variational Neural Network Ensembles. arXiv:2106.10760 [cs, stat], 2021.

[59] X. Hu, P. Szerlip, T. Karaletsos, and R. Singh. Applying SVGD to Bayesian Neural Networks for Cyclical Time-Series Prediction and Inference. arXiv:1901.05906 [cs, stat], 2019.

[60] L. Sun and J.-X. Wang. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, vol. 10, n. 3, pp. 161–169, 2020.

[61] N. Geneva and N. Zabaras. Quantifying model form uncertainty in Reynolds-averaged turbulence models with Bayesian deep neural networks. *Journal of Computational Physics*, vol. 383, pp. 125–147, 2019.

[62] P. G. Constantine, C. Kent, and T. Bui-Thanh. Accelerating Markov Chain Monte Carlo with Active Subspaces. *SIAM Journal on Scientific Computing*, vol. 38, n. 5, pp. A2779–A2805. Number: 5 Publisher: Society for Industrial and Applied Mathematics, 2016.

[63] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. arXiv:2010.08895 [cs, math], 2021.