Scaling Up Bayesian Uncertainty Quantification for Inverse Problems using Deep Neural Networks

Shiwei Lan*, Shuyi Li†, and Babak Shahbaba†

Abstract. Due to the importance of uncertainty quantification (UQ), Bayesian approach to inverse problems has recently gained popularity in applied mathematics, physics, and engineering. However, traditional Bayesian inference methods based on Markov Chain Monte Carlo (MCMC) tend to be computationally intensive and inefficient for such high dimensional problems. To address this issue, several methods based on surrogate models have been proposed to speed up the inference process. More specifically, the calibration-emulation-sampling (CES) scheme has been proven to be successful in large dimensional UQ problems. In this work, we propose a novel CES approach for Bayesian inference based on deep neural network models for the emulation phase. The resulting algorithm is computationally more efficient and more robust against variations in the training set. Further, by using an autoencoder (AE) for dimension reduction, we have been able to speed up our Bayesian inference method up to three orders of magnitude. Overall, our method, henceforth called *Dimension-Reduced Emulative Autoencoder Monte Carlo (DREAMC)* algorithm, is able to scale Bayesian UQ up to thousands of dimensions for inverse problems. Using two low-dimensional (linear and nonlinear) inverse problems we illustrate the validity of this approach. Next, we apply our method to two high-dimensional numerical examples (elliptic and advection-diffussion) to demonstrate its computational advantages over existing algorithms.

Key words. Bayesian Inverse Problems, Ensemble Kalman Methods, Emulation, Convolutional Neural Network, Dimension Reduction, Autoencoder

AMS subject classifications. 6208, 65M75, 68U07

1. Introduction.

There is a growing interest in uncertainty quantification (UQ) in the field of applied mathematics and its applications in physical sciences, biological sciences, and engineering, where UQ is commonly used to calibrate model inadequacy, carry out sensitivity analysis, or solve optimal control problems under uncertainty. As a result, Bayesian methods for inverse problems (e.g., reservoir modeling, weather forecasting) have become increasingly popular. Models in these application domains are usually constrained to physical or biological laws and are typically represented as ordinary or partial differential equation (ODE/PDE) systems. Implementing Bayesian UQ for such inverse problems is quite difficult because they involve computationally intensive simulations for 1) solving the ODE/PDE systems, and 2) sampling from the resulting high dimensional posterior distributions. To address these issues, we follow the work of [15] and propose a more scalable framework for Bayesian UQ that combines ensemble Kalman methods and infinite-dimensional Markov Chain Monte Carlo (MCMC) algorithms.

Ensemble Kalman (EnK) methods, originated from geophysics [24], have achieved significant success in state estimation for complex dynamical systems with noisy observations [28, 43, 1, 26, 25, 50, 55, 44, 48]. More recently, these methods have been used to solve in-

*School of Mathematical and Statistical Sciences, Arizona State University, AZ (slan@asu.edu, https://math.la.asu.edu/~slan/).

†Department of Statistics, University of California, Irvine, CA

verse problems with the objective of estimating the model parameters instead of the states [74, 12, 23, 46, 45, 27, 51, 31]. As a gradient-free optimization algorithm based on a small number of ensembles, these methods gained increasing popularity for solving inverse problems since they can be implemented non-intrusively and in parallel. However, due to the collapse of ensembles [84, 85, 22, 11], they tend to underestimate the posterior variance and thus fail to provide a rigorous basis for systematic UQ. To alleviate this issue, Cleary et al. combine Kalman methods with MCMC in three steps [15]: (i) calibrate models with ensemble Kalman methods, (ii) emulate the parameter-to-data map using evaluations of forward models, and (iii) generate posterior samples using MCMC based on cheaper emulators. The resulting approach is called *Calibration-Emulation-Sampling (CES)*. Two immediate benefits of such a framework are: 1) reusing expensive forward evaluations, and 2) computationally efficient surrogates in the MCMC procedure.

For the emulation component, [15] rely on Gaussian Process (GP) models, which have been widely used for emulating computer models [82], uncertainty analysis [71], sensitivity analysis [72], and computer code calibration [52, 39, 73]. While not adopted in [15], the derivative information can be extracted from GP [91, 58] to improve the sampling component [52, 71]. In general, however, GP emulation involves intensive computation ($\mathcal{O}(N^3)$ with N input/training data pairs) and does not scale well to high dimensions. Additionally, the prediction accuracy of GP emulator highly depends on the training set, which usually demands a substantial effort through careful experimental design [82, 83]. For these reasons, we propose to use deep neural network (NN) [34] for the emulation component of CES.

A deep artificial NN consists of multiple layers mapping the input (predictors) to the output (response). Each layer contains neurons (units) and synapses (connections) defining the architecture of NN. The resulting map also depends on a set of weights, biases, and activation functions, which are respectively analogous to regression coefficients, intercepts, and link functions in generalized linear models. Composition of multiple layers in different types makes the whole network capable of learning complex non-linear relationships, hence making NN a flexible functional approximation tool [3, 62, 86]. Deep NN has achieved enormous successes in image processing [56, 14], speech recognition [35], natural language processing [33, 67], bioinformatics [13, 87], and many other areas [89, 90]. Using stochastic batch optimization algorithms such as stochastic gradient descent [9], the computational cost of NN can be reduced to $\mathcal{O}(Np)$ with N being the size of the training data, and p being the total number of NN parameters [10].

In this paper, we particularly focus on convolutional neural networks (CNN), which are commonly used in image recognition [56]. In inverse problems, parameter functions (i.e., functions treated as parameters) are defined on finite 2d or 3d domains that can be viewed as images. Therefore, we expect CNN to be more suitable than regular dense neural networks (DNN) for such problems. This novel observation could lead to more effective and computationally efficient emulators. Although there have been a few related works where CNN is used on actual images [47, 66, 95] or as a prior for spatiotemporal dynamics [99] in inverse problems, to the best of our knowledge, this is the first application of CNN for training generic emulators in Bayesian inverse problems.

Besides computational challenges associated with building emulators, sampling from posterior distributions in inverse problems is also a challenging task due to the high dimensionality

of the target distribution. Traditional Metropolis-Hastings algorithms defined on the finite-dimensional space suffer from deteriorating mixing times upon mesh-refinement or increasing dimensions [79, 80, 7]. To overcome this deficiency, a new class of 'dimension-independent' MCMC methods [8, 6, 17, 61, 4, 5] has been proposed on infinite-dimensional Hilbert space. Despite of the robustness to increasing dimensions, these ∞-dimensional MCMC algorithms are still computationally demanding. Several recent papers have attempted to address this issue by using dimension reduction methods to find an intrinsic low-dimensional subspace that contains the most amount of information on the posterior distribution [19, 16, 57]. In this paper, we adopt autoencoder (AE) [40] to learn a low-dimensional latent representation of the parameter space with an encoder (original $\to$ latent) and a decoder (latent $\to$ reconstruction).

Combining CNN and AE, we propose a class of hybrid MCMC algorithms called *Dimension Reduced Emulative Autoencoder Monte Carlo (DREAMC)*, which can improve and scale up the application of the CES framework for Bayesian UQ from hundreds of dimensions (with GP emulation) [15] to thousands of dimensions (with NN emulation). In summary, this paper makes multiple contributions as follows:

1. apply CNN to train emulators for Bayesian inverse problems;
2. embed AE in CES to significantly improve its computational efficiency;
3. scale Bayesian UQ for inverse problems up to thousands of dimensions with DREAMC.

The rest of the paper is organized as follows. Section 2 provides a brief overview of Bayesian inverse problems and the CES framework for UQ. Further, gradient based MCMC algorithms (∞-MALA and ∞-HMC, not adopted in [15]) are reviewed for sampling. In Section 3, we apply various neural networks, including DNN, CNN, and AE, to scale up Bayesian UQ for inverse problems. Details of emulating functions, extracting gradients and reducing dimensions will also be discussed. Then, we illustrate the validity of DREAMC in Section 4 with simulated linear and nonlinear inverse problems. In Section 5, using two high-dimensional inverse problems involving an elliptic equation and an advection-diffusion equation, we demonstrate that our methods can speed up traditional MCMC algorithms up to three orders of magnitude. Section 6 concludes with some discussions on future research directions.

2. Bayesian UQ for Inverse Problems: Calibration-Emulation-Sampling.

In many PDE-constrained inverse problems, we are interested in finding an unknown function, u (symbolizing an 'unknown' function), given the observed data, y. The function u usually appears as a parameter (hence termed 'parameter function' thereafter) in the inverse problem. For example, u could be the (log)-transmissivity of subsurface flow that appears as a coefficient function in an elliptic PDE (Section 5.1) or the initial condition of a time-dependent advection-diffusion problem (Section 5.2). Let $\mathbb{X}$ and $\mathbb{Y}$ be two separable Hilbert spaces. (This is assumed for the convenience of developing probability spaces and can be relaxed to separable subspaces [21, 92].) There is a forward parameter-to-observation mapping $\mathcal{G} : \mathbb{X} \to \mathbb{Y}$, $u \mapsto \mathcal{G}(u)$ from the parameter space $\mathbb{X}$ to the data space $\mathbb{Y}$ (e.g. $\mathbb{Y} = \mathbb{R}^m$ for $m \geq 1$) that connects u to y as follows:

$$y = \mathcal{G}(u) + \eta, \qquad \eta \sim \mathcal{N}(0, \Gamma) \tag{2.1}$$

We can then define the potential function (negative log-likelihood), $\Phi : \mathbb{X} \times \mathbb{Y} \to \mathbb{R}$, as follows:

$$\Phi(u; y) = \frac{1}{2}\|y - \mathcal{G}(u)\|_{\Gamma}^2 = \frac{1}{2}\langle y - \mathcal{G}(u), \Gamma^{-1}(y - \mathcal{G}(u))\rangle \tag{2.2}$$

The forward mapping $\mathcal{G}$ represents physical laws usually expressed as large and complex ODE/PDE systems and could be highly non-linear. Therefore, repeated evaluations of the likelihood (and $\mathcal{G}(u)$) could be computationally demanding for different values of u.

In the Bayesian setting, a prior measure μ_0 is imposed on u, independent of η. For example, we could assume a Gaussian prior $\mu_0 = \mathcal{N}(0, \mathcal{C})$ with the covariance $\mathcal{C}$ being a positive, self-adjoint and trace-class (a.k.a. nuclear) operator on $\mathbb{X}$. (A trace-class operator $\mathcal{C}$ has summable eigenvalues $\lambda_n(\mathcal{C})$, i.e. $\mathrm{tr}(\mathcal{C}) := \sum_n \lambda_n(\mathcal{C}) < \infty$.) Then we can obtain the posterior of u, denoted as $\mu_{u|y}$, using Bayes' theorem [92, 21]:

$$\frac{d\mu_{u|y}}{d\mu_0}(u) = \frac{1}{Z}\exp(-\Phi(u; y)) \ , \quad \text{if } 0 < Z := \int_{\mathbb{X}} \exp(-\Phi(u; y))\mu_0(du) < +\infty \ .$$

For simplicity, we drop y and denote the posterior distribution and potential function as $\mu(du)$ and $\Phi(u)$ respectively. The Bayesian inverse problems involve estimating u and quantifying the associated uncertainty. For example, we are interested in tracing back the initial condition based on down-stream observations in an advection-diffusion problem (Section 5.2). This reduces to learning the posterior distribution $\mu(du)$, which can exhibit strong non-Gaussian behavior, posing enormous difficulties for commonly used inference methods such as MCMC.

In addition to the above-mentioned challenges in Bayesian UQ for inverse problems, the high dimensionality of the discretization of u makes the forward evaluation computationally intensive and imposes challenges on the robustness of sampling algorithms. To this end, the CES framework has been recently proposed for approximate Bayesian parameter learning. It consists of the following three stages [15]:

1. **Calibration**: using optimization-based algorithms (ensemble Kalman) to obtain parameter estimation and collect expensive forward evaluations for the emulation step;
2. **Emulation**: recycling forward evaluations in the calibration stage to build an emulator for sampling;
3. **Sampling**: sampling the posterior approximately based on the emulator, which is much cheaper than the original forward mapping.

The CES scheme is promising for high-dimensional Bayesian UQ in inverse problems. Emulation bypasses the expensive evaluation of original forward models (dominated by the cost of repeated forward solving of ODE/PDE systems) and reduces the cost of sampling to a small computational overhead. The sampling also benefits from the calibration, which provides MCMC algorithms with a good initial point in the high density region so that the burning time can be reduced.

In this paper, we choose NN for emulation instead of GP (used in [15]) for the computational efficiency and flexibility. Moreover, we extract the gradient evaluations directly from NN to implement gradient-based ∞-dimensional MCMC for sampling. We also adopt AE as a dimension reduction technique to further improve the efficiency of the original CES method [15]. In the following, we review ensemble Kalman methods for calibration and ∞-dimensional MCMC algorithms for sampling.

2.1. Calibration – Ensemble Kalman (EnK) Methods. For state-space models, Kalman filter [49] and its ensemble variants [24, 29] have become standard solvers because of their linear computational complexity. Recently, EnK methods have been introduced to solve inverse problems with the objective of estimating parameters rather than states [12, 46, 45, 27, 31].

Initializing J ensemble particles $\{u^{(j)}\}_{j=1}^J$ with, for example, prior samples, the basic ensemble Kalman inversion (EKI) method evolves each ensemble according to the following equation [84]:

$$\frac{du^{(j)}}{dt} = \frac{1}{J}\sum_{k=1}^{J}\left\langle \mathcal{G}(u^{(k)}) - \overline{\mathcal{G}}, y - \mathcal{G}(u^{(j)}) + \sqrt{\Sigma}\frac{dW^{(j)}}{dt}\right\rangle_{\Gamma}(u^{(k)} - \overline{u}) \tag{2.3}$$

where $\overline{u} := \frac{1}{J}\sum_{j=1}^J u^{(j)}$, $\overline{\mathcal{G}} := \frac{1}{J}\sum_{j=1}^J \mathcal{G}(u^{(j)})$, and $\{W^{(j)}\}$ are independent cylindrical Brownian motions on $\mathbb{Y}$. We can set $\Sigma = 0$ to remove noise for an optimization algorithm, or $\Sigma = \Gamma$ to add noise for dynamics that transform the prior to the posterior in one time unit for linear forward maps [84, 31].

A variant of EKI for approximate sampling from the posterior $\mu(du)$ is an ensemble Kalman sampler (EKS) [31, 32]. This is obtained by adding a prior-related damping term as in [11], and modifying the position-dependent noise in Equation (2.3):

$$\frac{du^{(j)}}{dt} = \frac{1}{J}\sum_{k=1}^{J}\left\langle \mathcal{G}(u^{(k)}) - \overline{\mathcal{G}}, y - \mathcal{G}(u^{(j)})\right\rangle_{\Gamma}(u^{(k)} - \overline{u}) - C(u)\mathcal{C}^{-1}u^{(j)} + \sqrt{2C(u)}\frac{dW^{(j)}}{dt} \tag{2.4}$$

with $C(u) := \frac{1}{J}\sum_{j=1}^J (u^{(j)} - \overline{u}) \otimes (u^{(j)} - \overline{u})$.

Both EKI and EKS are implemented by discretizing (2.3) and (2.4) respectively and updating $u_n^{(j)}$ iteratively for $n = 0, \cdots, N$. Due to the collapse of ensembles [84, 85, 22, 11], the sample variance estimated by ensembles $\{u_N^{(j)}\}_{j=1}^J$ tends to underestimate the true uncertainty. EKS [31] generally requires a large number of ensembles to faithfully sample from the true posterior. Figure 1 illustrates that both EKI and EKS with $J = 100$ ensembles severely underestimate the posterior standard deviation (plot in the 2d domain) of the parameter function in an elliptic inverse problem (see more details in Section 5.1). Figures **??** and **??** show more ensembles (500) might improve UQ, but the results highly depend on the specific problem at hand. Therefore, these methods do not provide a rigorous basis for systematic UQ, especially in high dimensions. To address this issue, we propose to implement scalable (dimension-independent) inference methods in the sampling step of CES.

2.2. Sampling – Infinite-Dimensional MCMC (∞-MCMC). Traditional Metropolis-Hastings algorithms are characterized by deteriorating mixing times upon mesh-refinement or with increasing dimensions. In contrast, a new class of dimension-independent algorithms – including *preconditioned Crank-Nicolson (pCN)* [17], *infinite-dimensional MALA (∞-MALA)* [8], *infinite-dimensional HMC (∞-HMC)* [6], and *infinite-dimensional manifold MALA (∞-mMALA)* [4] – has been recently developed. These algorithms are well-defined on the infinite-dimensional Hilbert space, and thus provide computational benefits with respect to mixing times for finite, but high-dimensional problems in practice.

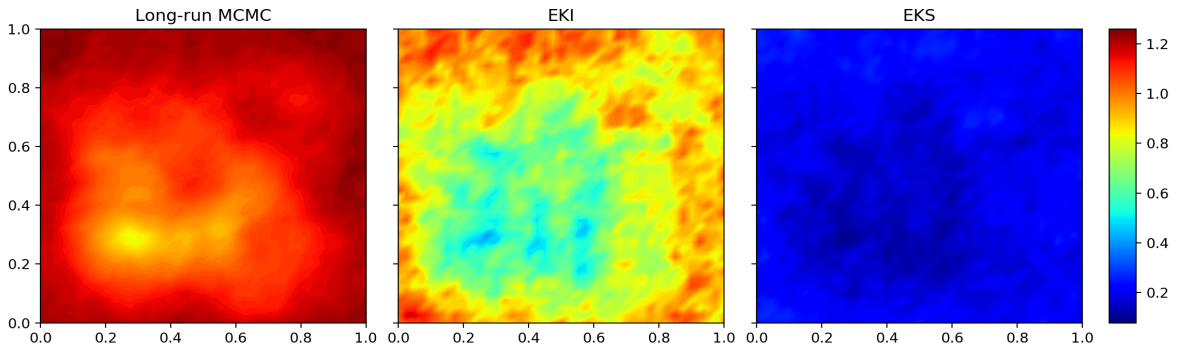


Figure 1: Comparing the estimation of standard deviation by MCMC (left panel) and ensemble Kalman methods (middle and right panels) with 100 ensembles in an elliptic inverse problem (Section 5.1).

Consider the following continuous-time Hamiltonian dynamics:

$$\frac{d^2u}{dt^2} + \mathcal{K}\left\{\mathcal{C}^{-1}u + D\Phi(u)\right\} = 0, \quad \left(v := \frac{du}{dt}\right)\Big|_{t=0} \sim \mathcal{N}(0, \mathcal{K}) \ . \tag{2.5}$$

If we let $\mathcal{K} \equiv \mathcal{C}$, Equation (2.5) preserves the total energy $H(u,v) = \Phi(u) + \frac{1}{2}\|v\|^2_{\mathcal{K}}$. HMC algorithm [70] solves (2.5) using the following Störmer-Verlet symplectic integrator [93]:

$$\begin{aligned} v^- &= v_0 - \tfrac{\alpha\varepsilon}{2}\mathcal{C}D\Phi(u_0) \ ; \\ \begin{bmatrix} u_\varepsilon \\ v^+ \end{bmatrix} &= \begin{bmatrix} \cos\varepsilon & \sin\varepsilon \\ -\sin\varepsilon & \cos\varepsilon \end{bmatrix} \begin{bmatrix} u_0 \\ v^- \end{bmatrix} \ ; \\ v_\varepsilon &= v^+ - \tfrac{\alpha\varepsilon}{2}\mathcal{C}D\Phi(u_\varepsilon) \ . \end{aligned} \tag{2.6}$$

Equation (2.6) gives rise to the leapfrog map $\Psi_\varepsilon : (u_0, v_0) \mapsto (u_\varepsilon, v_\varepsilon)$. Given a time horizon τ and current position u, the MCMC mechanism proceeds by concatenating $I = \lfloor \tau/\varepsilon \rfloor$ steps of leapfrog map consecutively: $u' = \mathcal{P}_u\{\Psi^I_\varepsilon(u,v)\}$, $v \sim \mathcal{N}(0,\mathcal{K})$, where $\mathcal{P}_u$ denotes the projection onto the u-argument. In ∞-HMC, the proposal u' is accepted with probability $a(u,u') = 1 \wedge \exp(-\Delta H(u,v))$[6]. We can use different step-sizes in (2.6): ε_1 for the first and third equations, and ε_2 for the second equation. Setting $I = 1$, $\varepsilon_1^2 = h, \cos\varepsilon_2 = \frac{1-h/4}{1+h/4}, \sin\varepsilon_2 = \frac{\sqrt{h}}{1+h/4}$, ∞-HMC reduces to ∞-MALA, which can also be derived from Langevin dynamics [8, 5]. When $\alpha = 0$, ∞-MALA further reduces to pCN [5], which does not use gradient information and can be viewed as an infinite-dimensional analogue of random walk Metropolis. While the original CES [15] only uses pCN in the sampling stage, we propose using ∞-MALA and ∞-HMC with the gradient extracted from NN emulation.

3. Scaling Up Bayesian UQ with Neural Networks. As mentioned above, there are two major challenges limiting the scalability of Bayesian UQ for inverse problems: 1) intensive computation required for repeated evaluations of likelihood (potential), $\Phi(u)$, and 2) high

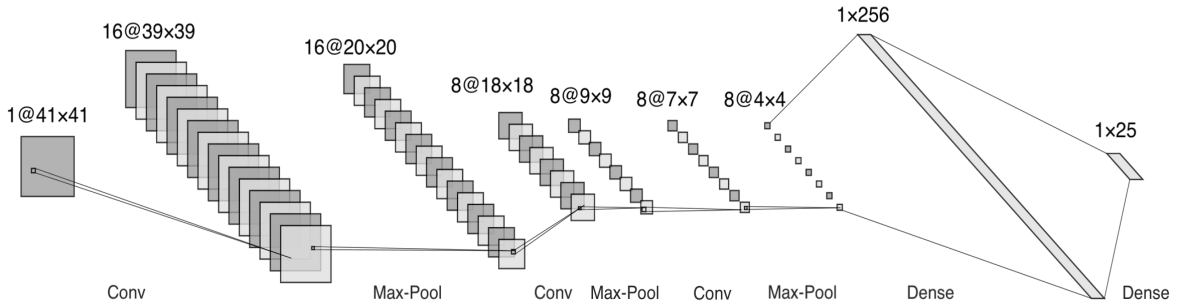


Figure 2: A typical architecture of convolutional neural network (CNN).

dimensionality of the discretized space. If we use ∞-MALA or ∞-HMC, we also need the gradient $D\Phi(u)$, which is typically not available. Here, we will use NNs to address these issues. More specifically, we train CNN to emulate the forward evaluation and AE to reduce the parameter dimensionality. In the following, we discretize the parameter function u into a vector of dimension d. When there is no confusion, we still denote the discretized parameter vector as $u \in \mathbb{R}^d$, which is the input of NNs. We also assume the observation $y \in \mathbb{R}^m$.

3.1. Emulation – Convolutional Neural Network (CNN). The ensemble-based algorithms in the calibration phase produce parameters and forward solutions $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}_{j=1}^J$ for $n = 0, \cdots, N$. These input-output pairs can be used to train a neural network model (DNN or CNN) as an emulator $\mathcal{G}^e$ of the forward mapping $\mathcal{G}$ [34]:

$$\mathcal{G}^e(u;\theta) = F_{K-1} \circ \cdots \circ F_0(u), \quad F_k(\cdot) = g_k(W_k \cdot + b_k) \in C(\mathbb{R}^{d_k}, \mathbb{R}^{d_{k+1}}) \tag{3.1}$$

where $d_0 = d$, $d_K = m$; $W_k \in \mathbb{R}^{d_{k+1} \times d_k}$, $b_k \in \mathbb{R}^{d_{k+1}}$, $\theta_k = (W_k, b_k)$, $\theta = (\theta_0, \cdots, \theta_{K-1})$; and g_k's are (continuous) activation functions. There are multiple choices of activation functions, e.g. $g_k(x) = (\sigma(x_1), \cdots, \sigma(x_{d_{k+1}}))$ with $\sigma \in C(\mathbb{R}, \mathbb{R})$ including rectified linear unit (ReLU, $\sigma(x_i) = 0 \vee x_i$) and leaky ReLU ($\sigma(x_i;\alpha) = x_i I(x_i \geq 0) + \alpha x_i I(x_i < 0)$). Alternatively, we can set $g_k(x) = (\sigma_1(x), \cdots, \sigma_{d_{k+1}}(x)) \in C(\mathbb{R}^{d_{k+1}}, \mathbb{R}^{d_{k+1}})$, with $\{\sigma_i\}$ defined as softmax: $\sigma_i(x) = e^{x_i} / \sum_{i'=1}^{d_{k+1}} e^{x_{i'}}$. In our numerical examples, activation functions for DNN/CNN are chosen such that the errors of emulating functions (and their extracted gradients) are minimized.

In many inverse problems, the parameter function u is defined over a 2-d or 3-d field, which possesses unique spatial features resembling an image. This has motivated our choice of CNN for emulators. Inspired by biological processes, where the connectivity pattern between neurons resembles the organization of visual cortex [30], CNN has become a powerful tool in image recognition and classification [56]. As a regularized NN with varying depth and width, CNN has much fewer connections and thus fewer training parameters compared to standard fully connected DNN of similar size. Therefore, CNN is preferred to DNN in the CES framework due to its flexibility and computational efficiency.

In general, CNN consists of a series of convolution layers with filters (kernels), pooling layers to extract features, and hidden layers fully connected to the output layer. The

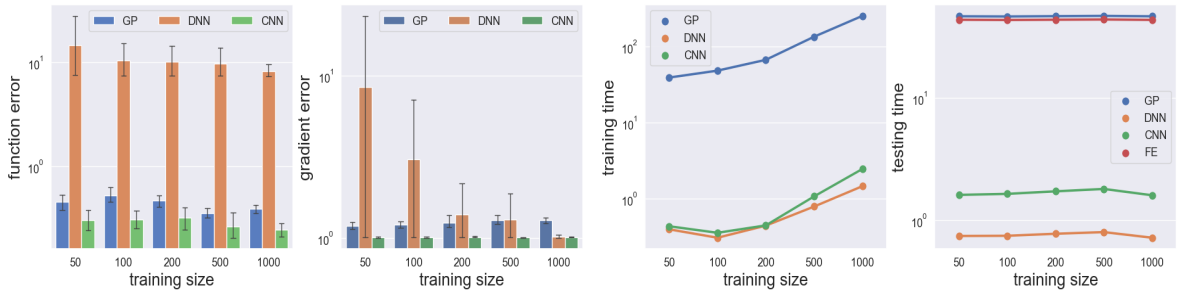


Figure 3: Comparing the emulation $\mathcal{G}^e : \mathbb{R}^{1681} \to \mathbb{R}^{25}$ in an elliptic inverse problem (Section 5.1) by GP, DNN and CNN in terms of error (left: function error $\|\Phi - \Phi^e\|$ and gradient error $\|D\Phi - D\Phi^e\|$) and time (right). Time is also compared with exact calculation of gradients by the finite element method (labeled 'FE') using adjoint codes in testing.

convolutional layer is introduced for sparse interaction, parameter sharing, and equivariant representations [34]. At convolutional layer k, instead of full matrix multiplication, we use discrete convolution [63] with a kernel function $w_k^{(c)}$ defined on one of the C_k feature channels:

$$F_k(\cdot) = [F_k^1(\cdot), \cdots, F_k^{C_k}(\cdot)], \quad F_k^{(c)}(\cdot) = g_k(w_k^{(c)} * \cdot + b_k^{(c)}) \in C(\mathbb{R}^{d_k}, \mathbb{R}^{d_{k+1}}) \tag{3.2}$$

On each channel c, the convolution operation $w_k^{(c)}*$ is defined by multiplying its discrete format, a circulant matrix $W_k^{(c)*}$, to its operand [34, 100]. Together $\{w_k^{(c)}*\}_{c=1}^{C_k}$ amount to a tensor operation. For each image input, CNN implements the convolution using a sliding window of pre-specified size (kernel size $s \geq 2$) with certain stride (step size) over the image. The resulting operation typically reduces dimension but can be made dimension preserving or expanding through padding [34].

After the convolutional layer, a pooling layer (e.g. max-pooling, average-pooling, or sum-pooling) is added to reduce the number of parameters by generating summary statistics (e.g., max, mean, or sum) of the nearby outputs. Such an operation is a form of non-linear down-sampling that sparsifies the NN but retains the most important information of the input image (function). Multiple pairs of convolutional and pooling layers (with different configurations) could be concatenated before passing the information to a dense layer to generate forward outputs $\{\mathcal{G}(u)\}$. Figure 2 illustrates the structure of a CNN used in the elliptic inverse problem (Section 5.1).

CES [15] adopts GP for the emulation step. Although infinitely wide NN with Gaussian priors can converge to GP [69, 64, 68] under certain conditions, the corresponding units do not represent 'hidden features' that capture important aspects of the data [69], e.g. the edge of the true log-transmissivity in Figure 9, or the truncated Gaussian blob as the true initial condition in Figure 12. Here, we show that a finite but potentially deep CNN as an emulator can provide multiple advantages over GP: 1) it is computationally more efficient for large training sets, 2) it is less sensitive to the locations of training samples (even if they are not spread out enough), and 3) it is possible to take advantage of all the ensemble samples collected by EKI or EKS to train CNN without the need to carefully "design" a training set of controlled size as it is

common in GP. After the emulator is trained, we could approximate the potential function using the predictions from CNN:

$$\Phi(u^*) \approx \Phi^e(u^*) = \frac{1}{2}\|y - \mathcal{G}^e(u^*)\|_\Gamma^2 \tag{3.3}$$

In the sampling stage, the computational complexity could be significantly reduced if we use Φ^e instead of Φ in the accept/reject step of MCMC. If the emulator is a good representation of the forward mapping, then the difference between Φ^e and Φ is small. Then, the samples by such emulative MCMC have the stationary distribution with small discrepancy compared to the true posterior $\mu(du)$.

In gradient-based MCMC algorithms, we need to calculate $D\Phi(u)$, i.e., the derivatives of (log) density function $\Phi(u)$ with respective to parameter function u. This is not always available because the forward mapping may not be differentiable (or available) in the solutions of ODE/PDE. However, (almost everywhere) differentiability is required for each layer of NN as it uses back-propagation [63, 41] in the training. Therefore, the gradient of the emulated potential function can be obtained by the chain rule

$$D\Phi^e(u^*) = -\langle y - \mathcal{G}^e(u^*), D\mathcal{G}^e(u^*)\rangle_\Gamma \tag{3.4}$$

where $D\mathcal{G}^e(u^*)$ can be the output from CNN's back-propagation, e.g. implemented in `GradientTape` of `TensorFlow`. Note that $D\Phi(u^*) \approx D\Phi^e(u^*)$ if $D\Phi(u^*)$ exists.

The universality of deep CNN for continuous functions has been established in [100]. This is generalized by the following theorem, which also gives the error bound of CNN emulator in approximating both the true potential Φ and its gradient $D\Phi$. To obtain the approximation rate, some regularity conditions are imposed on the target functions being approximated.

Theorem 3.1. *Let $2 \le s \le d$ and $\Omega \subset [-1,1]^d$. Assume $\mathcal{G}_j \in H^r(\mathbb{R}^d) \cap L^\infty([-1,1]^d)$ with $r \ge 1$ such that $v_{\mathcal{G}_j,2} := \int_{\mathbb{R}^d} \|\omega\|_1^2 |\widehat{\mathcal{G}}_j(\omega)| d\omega < \infty$ for $j = 1, \cdots, m$. If $K \ge 2d/(s-1)$, then there exist $\mathcal{G}^e$ by CNN with ReLU activation function such that*

$$\|\Phi - \Phi^e\|_{H^1(\Omega)} \le c v_{\mathcal{G},2} \sqrt{\log K} K^{-\frac{1}{2}-\frac{1}{2d}} \tag{3.5}$$

where we have $\|\Phi\|_{H^1(\Omega)} = \left(\|\Phi\|_{L^2(\Omega)}^2 + \|D\Phi\|_{L^2(\Omega)}^2\right)^{\frac{1}{2}}$, c is an absolute constant, and $v_{\mathcal{G},2} = \max_{1\le j\le m} v_{\mathcal{G}_j,2}$.

Proof. See Appendix **??**. ∎

Remark 1. *If $r > 2 + d/2$, then $v_{\mathcal{G},2} \le c\|\mathcal{G}\|$. Therefore, we have*

$$\|\Phi - \Phi^e\|_{H^1(\Omega)} \le c\|\mathcal{G}\| \sqrt{\log K} K^{-\frac{1}{2}-\frac{1}{2d}} \tag{3.6}$$

where $\|\mathcal{G}\| = \max_{1\le j\le m} \|\mathcal{G}_j\|_{H^r(\mathbb{R}^d)}$ with $\|\mathcal{G}_j\|_{H^r(\mathbb{R}^d)} := \|(1+|\omega|^2)^{r/2}\widehat{\mathcal{G}}_j(\omega)\|_{L^2(\mathbb{R}^d)}$.

Remark 2. *By [2], we have a weaker bound with sup-norm under the same condition of Theorem 3.1:*

$$\|\Phi - \Phi^e\|_{W^{1,\infty}(\Omega)} \le \tilde{c}\|\mathcal{G}\| K^{-\frac{1}{2}} \tag{3.7}$$

where $\|\Phi\|_{W^{1,\infty}(\Omega)} = \max_{0\le i\le d} \|D_i\Phi\|_{L^\infty(\Omega)}$ ($D_0\Phi = \Phi$).

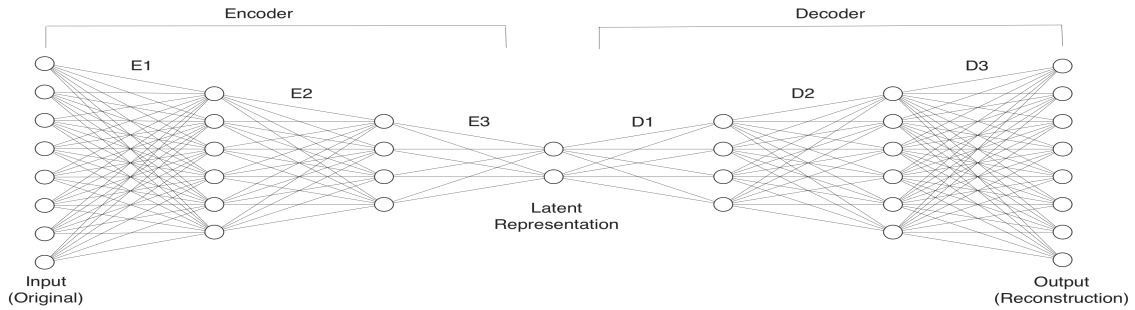


Figure 4: A typical architecture of autoencoder (AE) neural network.

Even if $D\Phi(u^*)$ does not exist, such gradient information, $D\Phi^e(u^*)$, can still be extracted from the emulator $\mathcal{G}^e$ to inform the landscape of Φ in the vicinity of u^*. Note that we train CNN only on $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}$ as opposed to $\{u_n^{(j)}, D\mathcal{G}(u_n^{(j)})\}$. That is, no gradient information is used for training. This is similar to extracting geometric information from GP emulator [91, 58]. Figure 3 compares GP, DNN, and CNN in emulating a forward map that takes a $1681(41\times41)$ dimensional discretized parameter u with 25 observations taken from the solution of an elliptic PDE as the output (see more details in Section 5.1). Given limited training data, CNN outperforms both GP and DNN by providing smaller approximation errors ($\|\Phi - \Phi^e\|$) with lower computational cost.

3.2. Dimension Reduction – Autoencoder (AE). Although we can reduce computation by emulation, the MCMC algorithms we use for Bayesian inference are still defined in high-dimensional spaces. In this section, we discuss using AE for dimensionality reduction to further speed up the UQ process [88]. AE is a special type of feed-forward NN for latent representation learning. The input is encoded into a low-dimensional latent representation (code). The code is then decoded into a reconstruction of the original input (see Figure 4). The model is trained to minimize the difference between the input and the reconstruction. An AE could learn complicated nonlinear dimensionality reduction. Therefore, it is widely used in many challenging tasks such as image recognition and artificial data generation [40].

While AE is commonly used to reduce the dimensionality of the data, here we use it to reduce the dimensionality of the (discretized) parameter space, still denoted as $\mathbb{X} \subset \mathbb{R}^d$. Denote the latent space as $\mathbb{X}_L$ with dimensionality $d_L \ll d$. Let $u_L \in \mathbb{X}_L$ be the latent representation of parameter u. Then the encoder ϕ and the decoder ψ are defined respectively as follows

$$\begin{aligned} \phi &: \mathbb{X} \to \mathbb{X}_L, \quad u \mapsto u_L \\ \psi &: \mathbb{X}_L \to \mathbb{X}, \quad u_L \mapsto u_R \end{aligned} \tag{3.8}$$

where $u_R \in \mathbb{X}$ is a reconstruction of u; ϕ and ψ can be chosen as multilayer NNs similar to Equation (3.1). Depending on the layers and structures, we could have convolutional AE (CAE) [36, 78], variational AE (VAE) [53, 54], etc. According to universal approximation theorem [20, 77, 65], a feed-forward artificial NN can approximate any continuous function given

some mild assumptions about the activation functions. Theoretically, an AE with suitable activation functions could represent an identity map, i.e. $\psi \circ \phi = id$. An accurate reconstruction of the input implies a good low-dimensional representation encoded in ϕ. In practice, the success of the algorithm heavily relies on the quality of the trained AE. There is a trade-off in choosing the proper latent dimensionality, d_L: smaller d_L throttles the information flowing through AE and leads to higher reconstruction errors (See Figure 8); larger d_L could reduce reconstruction error, but it may also negatively impact the computational efficiency of MCMC algorithms defined on the resulting latent subspace. Note that we train the AE with ensembles $\{u_n^{(j)}\}_{j=1,n=0}^{J,N}$ from the calibration stage. Even though $\psi \circ \phi$ might be different from the identity map id, AE could still provide a reconstruction $\psi \circ \phi(u)$ very close to the original parameter u. See Figure 10b (Section 5.1) and Figure 13b (Section 5.2) for examples.

The potential function $\Phi(u)$ and its derivative $D\Phi(u)$ can be projected to the latent space $\mathbb{X}_L$ – denoted as $\Phi_r(u_L)$ and $D\Phi_r(u_L)$ respectively – as follows:

$$\begin{aligned} \Phi_r(u_L) &= \Phi(u) = \Phi(\psi(u_L)) \\ D\Phi_r(u_L) &= \left(\frac{\partial u}{\partial u_L}\right)^T \frac{\partial \Phi(u)}{\partial u} = (d\psi(u_L))^T D\Phi(\psi(u_L)) \end{aligned} \tag{3.9}$$

where $d\psi = \frac{\partial u}{\partial u_L}$ is the Jacobian matrix of size $d \times d_L$ for the decoder ψ. The derivative information $D\Phi_r(u_L)$ needed in the gradient-based MCMC (∞-MALA and ∞-HMC) will be discussed in Section 3.3. In practice, we avoid explicit computation of the Jacobian matrix $d\psi$ by calculating the Jacobian-vector action altogether: $D\Phi_r(u_L) = \frac{\partial}{\partial u}\big|_{u=u_L} [\psi(u)^T D\Phi(\psi(u_L))]$, which is obtained from AE's back-propagation.

Now we are ready to combine emulation with dimension reduction to further improve the computation efficiency. The resulting approximate MCMC algorithms in the latent space involve potential function and its derivative, denoted as $\Phi_r^e(u_L)$ and $D\Phi_r^e(u_L)$ respectively, which are defined by replacing Φ with Φ^e in Equation (3.9).

3.3. Dimension Reduced Emulative Autoencoder MCMC (DREAMC). Next, we combine all the techniques discussed above to speed up Bayesian UQ for inverse problems. More specifically, our main method is composed of the following three stages:

1. **Calibration**: collect JN samples $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}_{j,n}$ from EKI or EKS procedure;
2. **Emulation**: build an emulator of the forward mapping $\mathcal{G}^e$ based on $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}_{j,n}$ (and extract $D\mathcal{G}^e$) using CNN; train an AE (ϕ, ψ) based on $\{u_n^{(j)}\}_{j,n}$;
3. **Sampling**: run approximate MCMC based on emulation to propose u' from u:
   i) obtain the projection of u by $u_L = \phi(u)$;
   ii) propose u'_L from u_L by ∞-MCMC (with Φ_r^e and $D\Phi_r^e$) in the latent space $\mathbb{X}_L$;
   iii) obtain the sample $u' = \psi(u'_L)$

Within the class of ∞-MCMC, we can use the emulated potential and its derivative instead of exact calculation. We refer to the resulting algorithms as *emulative ∞-MCMC (e-MCMC)*. Further, we can use AE to project these approximate MCMC into low-dimensional latent space. We denote these algorithms as *dimension-reduced emulative autoencoder ∞-MCMC (DREAMC)*. Figure 5 illustrates the relationship among various quantities involved in these MCMC algorithms.

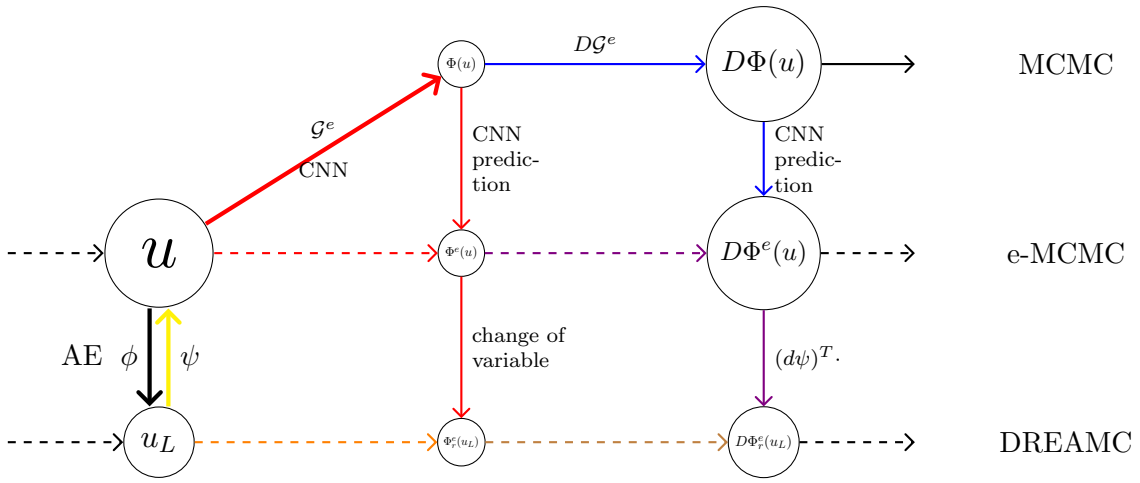


Figure 5: Relationship among quantities in various MCMC algorithms. Node sizes indicate relative dimensions of these quantities. Thick solid arrows mean training neural networks. Dashed arrows with colors represent mappings that are not directly calculated but actually have equivalent compositions indicated by the same color, e.g. $u \mapsto \Phi^e(u)$ (dashed red arrow) obtained by training CNN (thick solid red arrow) followed by network prediction (solid red arrow); or by color mixing, e.g. $u_L \mapsto \Phi_r^e(u_L)$ (dashed orange arrow) as a result of combing the decoder ψ (thick solid yellow arrow), $u \mapsto \Phi^e(u)$ (dashed red arrow), and the change of variable (solid red arrow).

We note that if we accept/reject proposals u_L' in the latent space with Φ_r^e, then there is no need to traverse between the original space and the latent space constantly. The chain can mainly stay in the latent space $\mathbb{X}_L$ to collect samples $\{u_L\}$, as shown in the bottom level of Figure 5, and move back to the original space $\mathbb{X}$ when relevant emulated quantities are needed. In the following, we describe the details of DREAMC algorithms.

For the convenience of following disposition, we first whiten the coordinates by the transformation $u \mapsto \tilde{u} := \mathcal{C}^{-\frac{1}{2}}u$. The whitened variable $\tilde{u}$ has the prior $\tilde{\mu}_0 = \mathcal{N}(0, \mathcal{I})$, where the identity covariance operator is not a trace-class on $\mathbb{X}$. However, random draws from $\tilde{\mu}_0$ are square-integrable in the weighted space $\mathrm{Im}(\mathcal{C}^{-\frac{1}{2}})$. We can still obtain a well-defined function space proposal for parameter u after inverting the transformation [19, 57]. In the whitened coordinates $\tilde{u}$, the Langevin and Hamiltonian (2.5) dynamics (with algorithmic parameter $\alpha \equiv 1$) can be written as follows respectively:

$$\frac{d\tilde{u}}{dt} = -\frac{1}{2}\left\{\mathcal{I}\tilde{u} + \alpha D\Phi(\tilde{u})\right\} + \frac{dW}{dt} \tag{3.10}$$

$$\frac{d^2\tilde{u}}{dt^2} + \left\{\mathcal{I}\tilde{u} + \alpha D\Phi(\tilde{u})\right\} = 0, \quad \left(\tilde{v} := \frac{d\tilde{u}}{dt}\right)\Big|_{t=0} \sim \mathcal{N}(0, \mathcal{I}) \ . \tag{3.11}$$

where $D\Phi(\tilde{u}) = \mathcal{C}^{\frac{1}{2}}D\Phi(u)$. Then, we can train CNN using $\{\tilde{u}_n^{(j)}, \mathcal{G}(\tilde{u}_n^{(j)})\}_{j,n}$ and AE using $\{\tilde{u}_n^{(j)}\}_{j,n}$.

On the other hand, since the AE model does not preserve the volume ($\psi \circ \phi \approx id$ but $\psi \circ \phi \neq id$), the acceptance of proposals in the latent space needs to be adjusted with a volume

correction term $\frac{V'}{V}$ in order to maintain the ergodicity [59, 88]. Note, the volume adjustment term $\frac{V'}{V}$ breaks into the product of Jacobian determinants of the encoder ϕ and the decoder ψ that can be calculated with Gramian function as follows [88]:

$$\frac{V'}{V} = \det(d\psi(\tilde{u}'_L))\det(d\phi(\tilde{u})) = \sqrt{\det\left[\left(\frac{\partial\tilde{u}'}{\partial\tilde{u}'_L}\right)^T\left(\frac{\partial\tilde{u}'}{\partial\tilde{u}'_L}\right)\right]}\sqrt{\det\left[\left(\frac{\partial\tilde{u}_L}{\partial\tilde{u}}\right)\left(\frac{\partial\tilde{u}_L}{\partial\tilde{u}}\right)^T\right]} \tag{3.12}$$

where terms under square root are determinants of matrices with small size $d_L \times d_L$, which can be obtained by the singular value decomposition of the Jacobian matrices respectively. In practice, we can exclude $\frac{V'}{V}$ from the acceptance probability and use it as a resampling weight as in importance sampling [60]. Alternatively, we can ignore the accept/reject step for an approximate Bayesian UQ [98, 88].

To derive ∞-HMC in the latent space based on the Hamiltonian dynamics in the whitened coordinates (3.11), we also need to project $\tilde{v} \sim \mathcal{N}(0, I_d)$ into d_L-dimensional latent space $\mathbb{X}_L$. We could use the same encoder ϕ as in [88]; however, since $\tilde{v}_i \overset{iid}{\sim} \mathcal{N}(0,1)$ for $i = 1, \cdots, d$, we just set $\tilde{v}_L \sim \mathcal{N}(0, I_{d_L})$ for simplicity. Then, the ∞-HMC proposal $\Psi_\varepsilon : (\tilde{u}_{L,0}, \tilde{v}_{L,0}) \mapsto (\tilde{u}_{L,\varepsilon}, \tilde{v}_{L,\varepsilon})$ in the whitened augmented latent space with emulated gradient becomes

$$\begin{aligned} \tilde{v}_L^- &= \tilde{v}_{L,0} - \tfrac{\alpha\varepsilon}{2} D\Phi_r^e(\tilde{u}_{L,0}) \ ; \\ \begin{bmatrix} \tilde{u}_{L,\varepsilon} \\ \tilde{v}_L^+ \end{bmatrix} &= \begin{bmatrix} \cos\varepsilon & \sin\varepsilon \\ -\sin\varepsilon & \cos\varepsilon \end{bmatrix} \begin{bmatrix} \tilde{u}_{L,0} \\ \tilde{v}_L^- \end{bmatrix} \ ; \\ \tilde{v}_{L,\varepsilon} &= \tilde{v}_L^+ - \tfrac{\alpha\varepsilon}{2} D\Phi_r^e(\tilde{u}_{L,\varepsilon}) \ . \end{aligned} \tag{3.13}$$

The acceptance probability for the resulting DREAMC-∞-HMC algorithm involves $H(\tilde{u}_L, \tilde{v}_L) = \Phi_r^e(\tilde{u}_L) + \frac{1}{2}\|\tilde{v}_L\|^2$ and becomes $a(\tilde{u}_L, \tilde{u}'_L) = 1 \wedge \exp(-\Delta H(\tilde{u}_L, \tilde{v}_L))\frac{V'}{V}$ with $\frac{V'}{V}$ as in (3.12) and

$$\begin{aligned} \Delta H(\tilde{u}_L, \tilde{v}_L) =& H(\Psi_\varepsilon^I(\tilde{u}_L, \tilde{v}_L)) - H(\tilde{u}_L, \tilde{v}_L) \\ =& \Phi(\tilde{u}_{L,I}) - \Phi(\tilde{u}_{L,0}) - \frac{\alpha^2\varepsilon^2}{8}\left\{\|D\Phi_r^e(\tilde{u}_{L,I})\|^2 - \|D\Phi_r^e(\tilde{u}_{L,0})\|^2\right\} \\ & - \frac{\alpha\varepsilon}{2}\sum_{i=0}^{I-1}(\langle\tilde{v}_{L,i}, D\Phi_r^e(\tilde{u}_{L,i})\rangle + \langle\tilde{v}_{L,i+1}, D\Phi_r^e(\tilde{u}_{L,i+1})\rangle) \end{aligned} \tag{3.14}$$

We summarize DREAMC-∞-HMC in Algorithm **??** in Appendix **??**, which includes DREAMC-∞-MALA with $I = 1$ and DREAMC-pCN with $\alpha = 0$ as special cases.

4. Illustrations. In this section, we investigate two low-dimensional inverse problems: a three-dimensional linear Gaussian inverse problem with tractable posterior distribution and a four-dimensional nonlinear banana-biscuit-doughnut (BBD) distribution [58] with complex geometry. We illustrate the validity of our proposed emulative ∞-MCMC and DREAMC algorithms using these two examples. The structure of our NN models (e.g., the number of layers and choice of activation functions) in this section and next section are chosen from a small subset of options to minimize the overall error; therefore, they may not optimal globally. All computer codes are available at `GitHub` https://github.com/lanzithinking/DREAMC-BUQ.

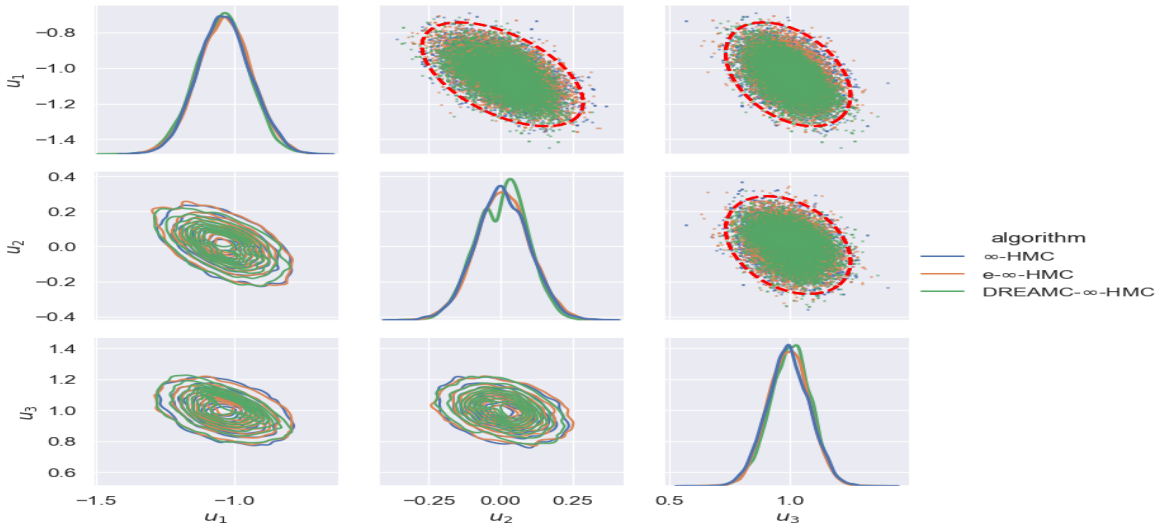


Figure 6: Linear Gaussian inverse problem: pairwise marginal posterior density estimation.

4.1. Linear Gaussian inverse problem. We first consider the following linear Gaussian inverse problem.

$$y = \mathcal{G}(u) + \eta, \quad \mathcal{G}(u) = Au, \quad \eta \sim \mathcal{N}(0, \Gamma)$$
$$u \sim \mathcal{N}(0, \Sigma_0)$$

where A is a matrix of size $m \times d$. In this example, we set $\Gamma = \sigma_\eta^2 I_m$ with $\sigma_\eta^2 = 0.1$ and $\Sigma_0 = \sigma_u^2 I_d$ with $\sigma_u^2 = 1$. We randomly generate A with $a_{ij} \sim \text{unif}[0, 1]$. Further, we assume $d = 3$ and set the true value $u^\dagger = (-1, 0, 1)$. We generate $m = 100$ data points, $y = \{y_n\}_{n=1}^m$, with $\mathcal{G}(u^\dagger) = Au^\dagger$. The inverse problem involves finding the posterior distribution of $u|y$ which has the following analytic form:

$$u|y \sim \mathcal{N}(\mu, \Sigma), \quad \mu = \Sigma A^\mathsf{T} \Gamma^{-1} y, \quad \Sigma^{-1} = \Sigma_0^{-1} + A^\mathsf{T} \Gamma^{-1} A$$

We follow the procedure of CES outlined in Section 3.3. First, we run EKS with ensemble size $J = 100$ for $N = 50$ iterations and collect 5000 ensemble pairs $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}_{j=1,n=1}^{J,N}$. Then we train DNN with 75% of these ensembles and use the remaining 25% for testing. The DNN has 3 layers with 'softplus' activation function for the hidden layers and 'linear' activation for the output layer with the units linearly interpolated between input dimension ($d = 3$) and output dimension ($m = 100$). We also train AE with the same split of training/testing data. The AE has latent dimension $d_L = 2$, with 2 encoder layers and 2 decoder layers. We use 'LeakyReLU($\alpha = 2$)' as the activation function. We run ∞-HMC, emulative ∞-HMC and DREAMC ∞-HMC (Algorithm **??**) to collect 10000 posterior samples of u after burning in the first 10000 respectively. Figure 6 shows that both emulative ∞-HMC and DREAMC ∞-HMC generate samples very close to the original ∞-HMC. All three methods capture the true distribution whose 3-standard deviation contours are plotted as red ellipses.

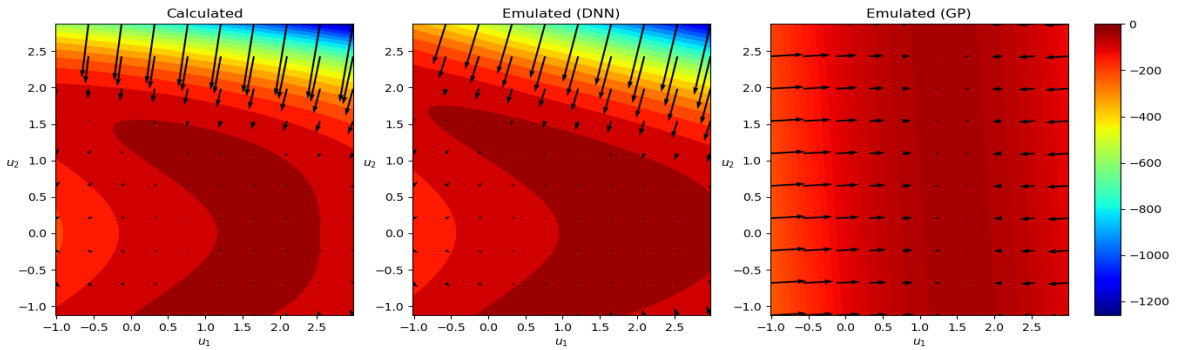


Figure 7: Nonlinear BBD inverse problem: comparing the true (left) potential (colored contours) and its gradient (arrows) with the corresponding emulated quantities given by DNN (middle) and GP (right).

4.2. Nonlinear Banana-Biscuit-Doughnut (BBD) inverse problem. Next, we challenge our proposed methodology with a complex four-dimensional Banana-Biscuit-Doughnut (BBD) distribution [58]. BBD distribution was first proposed in [58] as a benchmark for testing MCMC algorithms and has recently been revisited by [88]. The name of the distribution comes from 3 possible shapes of pairwise marginal distributions resembling a banana in (1,2) dimension, a biscuit in (1,3) dimension, and a doughnut in (2,4) dimension (see Figure **??** in Appendix **??**). It can be cast into a Bayesian inverse problem with parameters $u = (u_1, \cdots, u_d)$:

$$\begin{aligned} y &= \mathcal{G}(u) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma_\eta^2 I_m) \\ \mathcal{G}(u) &= A\mathcal{S}u, \quad \mathcal{S}u = \left(u_1, u_2^2, \cdots, u_k^{p_k}, \cdots, u_d^{p_d}\right), \quad p_k = 2 - (k \bmod 2) \\ u &\sim \mathcal{N}(0, \sigma_u^2 I_d) \end{aligned}$$

where A is a matrix of size $m \times d$. Therefore, $\mathcal{G} : \mathbb{R}^d \to \mathbb{R}^m$ is a linear forward mapping of $\mathcal{S}u$ but a non-linear operator of u.

As before, we generate a matrix A and true vector $u^\dagger$ with elements being random integers between 0 and d. Then, we obtain $m = 100$ data points, $\{y_n\}_{n=1}^m$, with $\mathcal{G}(u^\dagger) = A\mathcal{S}u^\dagger, \sigma_\eta^2 = 1$. In the prior, we set $\sigma_u^2 = 1$. The inverse problem involves finding u for given data $\{y_n\}_{n=1}^m$. The distribution has a complex geometric structure, which makes it challenging for MCMC algorithms to explore (Figure **??** in Appendix **??**).

We collect training samples, $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}_{j=1,n=1}^{J,N}$, using EKS with $J = 100$ ensembles for $N = 50$ iterations. Then, we train DNN (with a similar structure as before, but with 5 layers) and AE (with the same configuration as before, but with latent dimension $d_L = 3$). For comparison, we also train GP with anisotropic squared exponential kernel based on these 5000 ensembles. The GP model is coded in `GPflow`, which uses scalable variational GP (SVGP) algorithms [75, 38]. SVGP determines the value of hyper-parameters (e.g. length scales) by maximizing the evidence lower bound (ELBO). In `GPflow`, stochastic optimizers such as stochastic gradient descent (SGD) and adaptive moment estimation (ADAM) can be

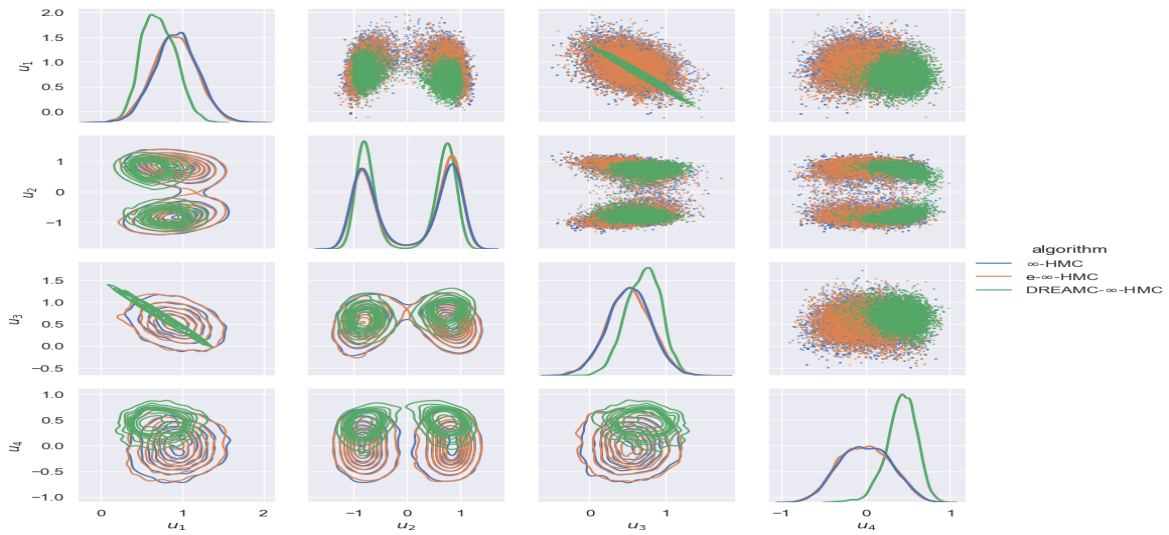


Figure 8: BBD inverse problem: pairwise marginal posterior density estimation.

directly used. The set-up for choosing mini-batches in our GP training is the same as our DNN training. Both emulators, $\mathcal{G}^e : \mathbb{R}^4 \to \mathbb{R}^{100}$, are built for $A\mathcal{S}$, and the log-likelihood is computed using (3.3). Figure 7 shows that the emulation by DNN (middle) is much closer to the truth (left) than that of GP (right). As we can see from these two illustrative examples, GP works equally well as DNN in emulating the low-dimensional linear map, but it performs much worse in the high-dimensional (many-output) nonlinear problems.

Next, we run ∞-HMC, emulative ∞-HMC and DREAMC ∞-HMC to collect 10000 posterior samples of u after discarding the first 10000. We plot the marginal distributions of model parameters estimated by these MCMC samples in Figure 8. The resulting distribution from emulative ∞-HMC is close to that of ∞-HMC. In this example, the intrinsic dimension (the dimension of space containing essential data information) is $d = 4$ but the latent dimension is $d_L = 3$. Therefore, some information is lost in AE. This leads to the discrepancy between DREAMC ∞-HMC and the other two algorithms. Still, the DREAMC algorithm recovers enough details of the posterior.

5. Numerical Experiments. In this section, we consider two high-dimensional inverse problems involving elliptic PDE and advection-diffusion equation. In both problems, the forward parameter-to-observation mappings are nonlinear, and the posterior distributions are non-Gaussian. The high dimensionality of the discretized parameter imposes a big challenge on Bayesian UQ. The second inverse problem involving advection-diffusion equation is even more challenging because it is based on spatiotemporal observations. We demonstrate substaintial numerical advantages of our proposed methods and show that they indeed can scale up the Bayesian UQ for PDE-constrained inverse problems to thousands of dimensions.

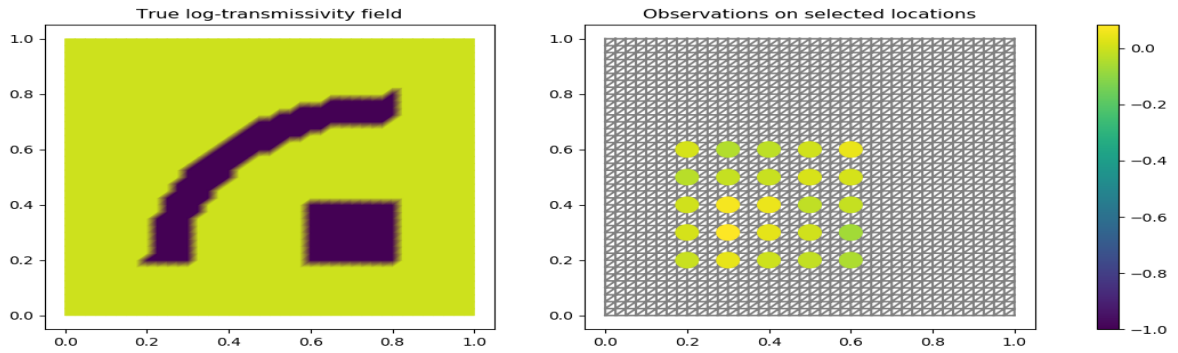


Figure 9: Elliptic inverse problem: true log-transmissivity field $u^\dagger(\mathbf{x})$ (left), and 25 observations on selected locations indicated by circles (right), with color indicating their values.

5.1. Elliptic Inverse Problem. The following elliptic PDE [19, 57] is defined on the unit square domain $\Omega = [0,1]^2$:

$$\begin{aligned} -\nabla \cdot (k(\mathbf{x})\nabla p(\mathbf{x})) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega \\ \langle k(\mathbf{x})\nabla p(\mathbf{x}), \vec{n}(\mathbf{x})\rangle &= 0, \quad \mathbf{x} \in \partial\Omega \\ \int_{\partial\Omega} p(\mathbf{x}) dl(\mathbf{x}) &= 0 \end{aligned} \tag{5.1}$$

where $k(\mathbf{x})$ is the transmissivity field, $p(\mathbf{x})$ is the potential function, $f(\mathbf{x})$ is the forcing term, and $\vec{n}(\mathbf{x})$ is the outward normal to the boundary. The source/sink term $f(\mathbf{x})$ is defined by the superposition of four weighted Gaussian plumes with standard deviation 0.05, centered at $\mathbf{x} = [0.3, 0.3]$, $[0.7, 0.3]$, $[0.7, 0.7]$, $[0.3, 0.7]$, with weights $\{2, -3, 3, -2\}$ respectively, as shown in the left panel of Figure **??**.

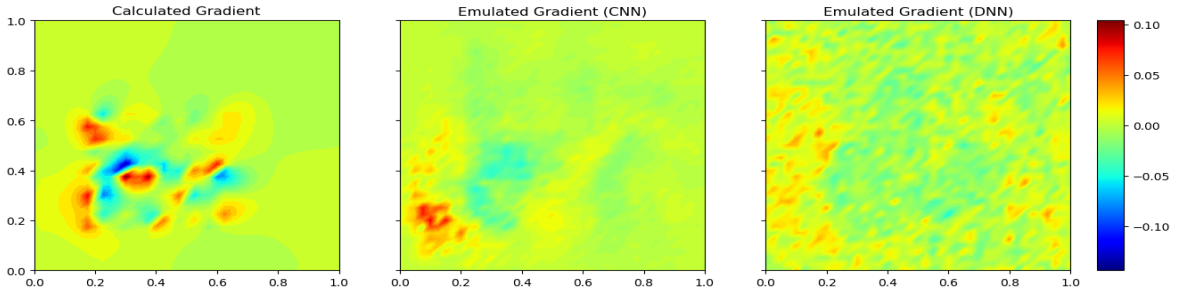
The transmissivity field is endowed with a log-Gaussian prior, i.e.

$$k(\mathbf{x}) = \exp(u(\mathbf{x})), \quad u(\mathbf{x}) \sim \mathcal{N}(0, \mathcal{C})$$

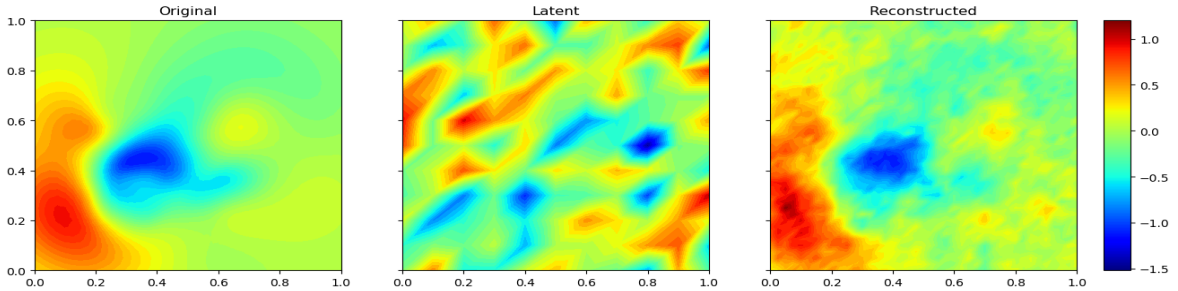
where the covariance operator $\mathcal{C}$ is defined through an exponential kernel function

$$\mathcal{C} : \mathbb{X} \to \mathbb{X}, \; u(\mathbf{x}) \mapsto \int c(\mathbf{x}, \mathbf{x}')u(\mathbf{x}')d\mathbf{x}', \quad c(\mathbf{x}, \mathbf{x}') = \sigma_u^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{2\ell}\right), \text{ for } \mathbf{x}, \mathbf{x}' \in \Omega$$

with the prior standard deviation $\sigma_u = 1.25$ and the correlation length $\ell = 0.0625$. To make the inverse problem more challenging, we follow [19] to use a true log transmissivity field $u^\dagger(\mathbf{x})$ that is not drawn from the prior, as shown in the left panel of Figure 9. The right panel of Figure **??** shows the potential function, $p(\mathbf{x})$, solved with $u^\dagger(\mathbf{x})$, which is also used for generating noisy observations. Partial observations are obtained by solving $p(\mathbf{x})$ on an 81×81 mesh and then collecting at 25 measurement sensors $\{\mathbf{x}_i\}_{i=1}^{25}$ as shown by the circles on the right panel of Figure 9. The corresponding observation operator $\mathcal{O}$ yields the data



(a) CNN (middle) and DNN (right) emulation ($\mathcal{G}^e : \mathbb{R}^{1681} \to \mathbb{R}^{25}$) extracting gradients $D\Phi^e(u^{\text{MAP}})$ compared with the true gradient $D\Phi(u^{\text{MAP}})$ (left).



(b) AE compressing the original function u^{MAP} (left) into latent space u_r^{MAP} (middle) and reconstructing it in the original space $u^{\text{MAP}'}$ (right).

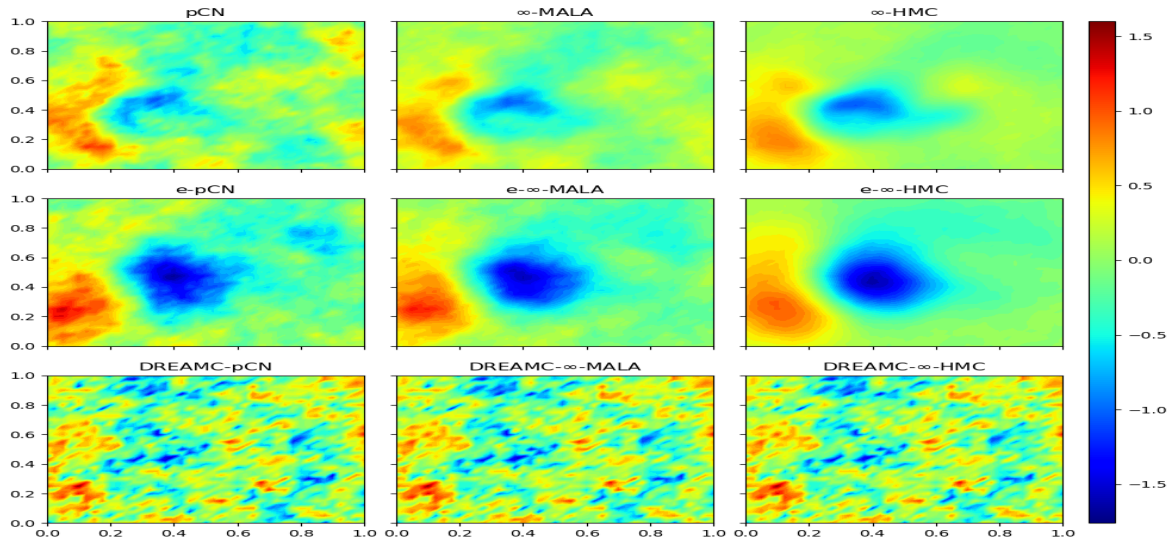
Figure 10: Elliptic inverse problem: outputs by NNs viewed as 2d images.

$y \in \mathbb{R}^{25}$

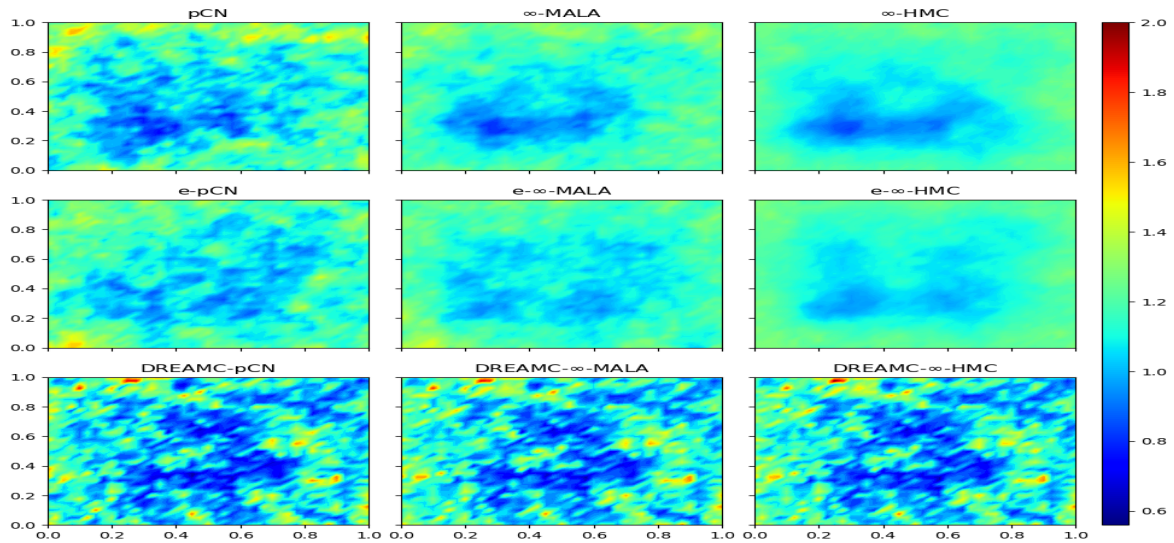
$$y = \mathcal{O}p(\mathbf{x}) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma_\eta^2 I_{25})$$

where we consider the signal-to-noise ratio $\text{SNR} = \max_{\mathbf{x}}\{u(\mathbf{x})\}/\sigma_\eta = 50$ in this example.

The inverse problem involves sampling from the posterior of the log-transmissivity field $u(\mathbf{x})$, which becomes a vector with dimension of 1681 after being discretized on 41×41 mesh (with Lagrange degree 1). We use the CES framework described in Section 3.3. In the calibration stage, we collect $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}_{j=1,n=1}^{J,N}$ from $N = 10$ iterations of EKS runs with ensemble size $J = 500$. For the emulation, we train DNN and CNN with 75% of these 5000 ensembles and test/validate them with the remaining 25%. The DNN has 3 layers with 'softplus' activation function for the hidden layers and 'linear' activation for the output layer and 40% nodes dropped out. The structure of CNN is illustrated in Figure 2 with 'softplus' activation for the convolution layers, 'softmax' activation for the latent layer (dimension 256) and 'linear' activation for the output layer. The trained CNN has drop out rate of 50% on all its nodes. Figure 10a compares the true gradient function $D\Phi(u^{\text{MAP}})$ (left panel) and its emulations $D\Phi^e(u^{\text{MAP}})$ (middle and right panels) as in Equation (3.4) extracted from two types of NNs. These gradient functions are plotted on the 2d domain $[0,1]^2$. We can



(a) Posterior mean estimates of the log-transmissivity field $u(\mathbf{x})$.



(b) Posterior standard deviation estimates of the log-transmissivity field $u(\mathbf{x})$.

Figure 11: Elliptic inverse problem: Bayesian posterior estimates of the log-transmissivity field $u(\mathbf{x})$ based on 5000 samples by various MCMC algorithms.

see that even trained on forward outputs without any gradient information, these extracted gradients provide decent approximations to the true gradient capturing its main graphical feature viewed as a 2d image. The result by CNN is qualitatively better than DNN, which is supported by the numeric evidence of error comparison illustrated in the left panel of Figure 3.

In the sampling stage, we train AE with the structure illustrated in Figure 4. The latent dimension is $d_L = 121$ (11×11) and the sizes of hidden layers between input and latent, between latent and output are linearly interpolated. All the activation functions are chosen as 'LeakyReLU($\alpha = 2$)'. Figure 10b plots the original u^{MAP} (left), the latent representation $u_r^{\text{MAP}} = \phi(u^{\text{MAP}})$ (middle) and the reconstruction $u^{\text{MAP}'} = \psi(u_r^{\text{MAP}})$ (right). Even though the latent representation is not very intuitive, the output function (image) decoded from the latent space can be viewed as a 'faithful' reconstruction of the original function (image), indicating a sufficiently good AE that compresses and restores information. Therefore, our proposed MCMC algorithms, defined on the latent space, generate samples that can be projected back to the original space without losing too much accuracy in representing the posterior distribution.

Method	h [a]	AP [b]	s/iter [c]	ESS(min,med,max) [d]	minESS/s [e]	spdup [f]	PDEsolns [g]
pCN	0.03	0.65	0.49	(7.8,28.93,73.19)	0.0032	1.00	6001
∞-MALA	0.15	0.61	0.56	(29.21,120.79,214.85)	0.0105	3.30	12002
∞-HMC	0.10	0.70	1.65	(547.62,950.63,1411.6)	0.0663	20.82	36210
e-pCN	0.05	0.60	0.02	(10.07,43.9,93.62)	0.0879	27.60	0
e-∞-MALA	0.15	0.67	0.03	(33.23,133.54,227.71)	0.2037	63.95	0
e-∞-HMC	0.10	0.77	0.07	(652.54,1118.08,1455.56)	1.9283	**605.47**	0
DREAMC-pCN	0.10	0.67	0.02	(36.78,88.36,141.48)	0.3027	95.03	0
DREAMC-∞-MALA	1.00	0.66	0.04	(391.53,782.06,927.08)	2.0988	**659.01**	0
DREAMC-∞-HMC	0.60	0.64	0.11	(2289.86,3167.03,3702.4)	4.1720	**1309.97**	0

[a] step size [b] acceptance probability [c] seconds per iteration [d] (minimum, median, maximum) effective sample size [e] minimal ESS per second [f] comparison of minESS/s with pCN as benchmark [g] number of PDE solutions

Table 1: Elliptic inverse problem: sampling efficiency of various MCMC algorithms.

We compare the performance of algorithms including vanilla pCN, ∞-MALA, ∞-HMC, their emulative versions, and the corresponding DREAMC algorithms. For each algorithm, we run 6000 iterations and burn in the first 1000. For HMC algorithms, we set $I = 5$. We tune the step sizes for each algorithm so that they have similar acceptance rates around $60 \sim 70\%$. Figure 11a compares their posterior mean estimates, and Figure 11b compares their estimates of posterior standard deviation. We can see that emulative MCMC algorithms generate results very close to those by the original MCMC methods. DREAMC algorithms introduce more errors due to the information loss in AE, but still provides estimates that reasonably resemble those generated by the original MCMC algorithms.

Table 1 summarizes the sampling efficiency of various MCMC algorithms measured by minimum effective sample size (ESS) among all parameters normalized by the total time consumption, i.e. minESS/s. With this standard, emulative ∞-HMC and DREAMC ∞-MALA achieve more than 600 times speed-up in sampling efficiency and DREAMC ∞-HMC attains 3 orders of magnitude improvement compared to the benchmark pCN. Such comparison focuses on the cost of obtaining uncertainty estimates and does not include the time for training CNN and AE, which is relatively much smaller compared with the overall sampling time.

Figure **??** (Appendix **??**) shows the traceplots of the potential function (data-misfit) on the left panel and autocorrelation functions on the right panel. HMC algorithms make distant proposals with least autocorrelation, followed by MALA algorithms and then pCN algorithms with the highest autocorrelation. This is also verified numerically by ESS of parameters (the

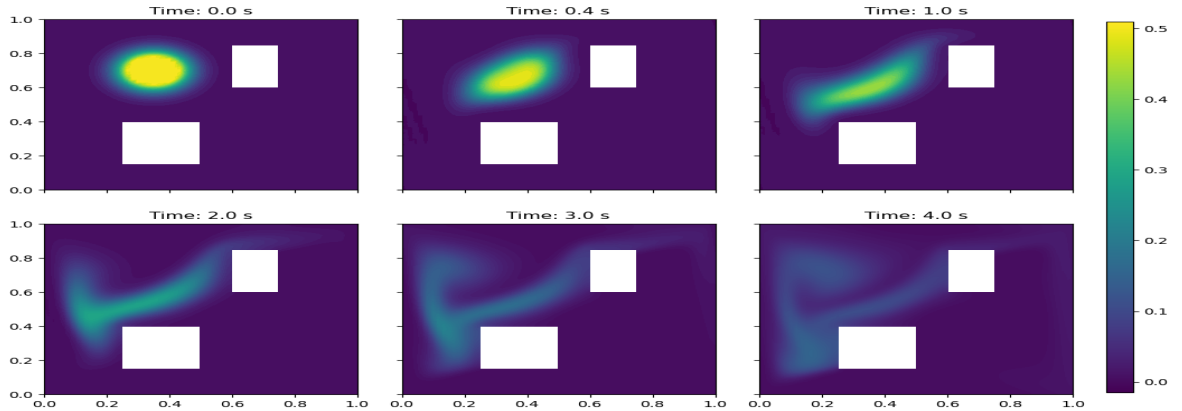


Figure 12: Advection-diffusion inverse problem: true initial condition $u_0^\dagger$ (top left), and the solutions $u(\mathbf{x}, t)$ at different time points.

lower autocorrelation, the higher ESS) in Table 1. Note DREAMC ∞-MALA has similar autocorrelation as HMC algorithms. Finally, we plot the Kullback–Leibler (KL) divergence between the posterior and the prior in terms of iteration (upper) and time (lower) respectively in Figure **??** (Appendix **??**). Among all the MCMC algorithms, emulative MCMC algorithms stabilize such measurement the fastest and attain smaller values for given iterations and time.

5.2. Advection-Diffusion Inverse Problem. In the next example, we quantify the uncertainty in the solution of an inverse problem governed by a parabolic PDE within the Bayesian inference framework. The underlying PDE is a time-dependent advection-diffusion equation in which we seek to infer an unknown initial condition from spatiotemporal point measurements.

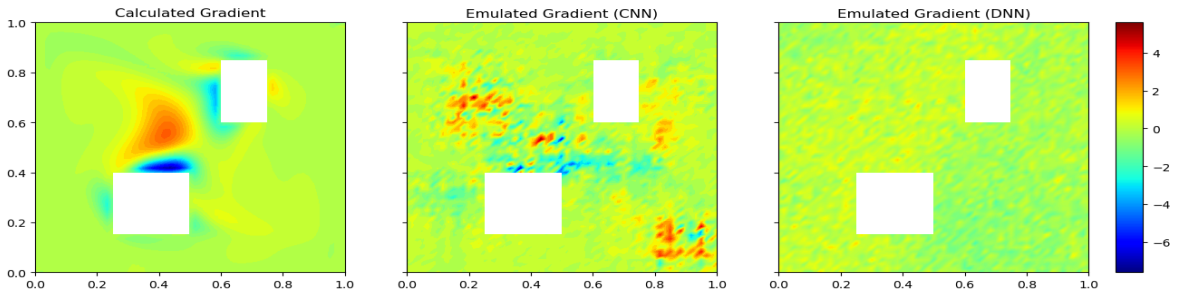
The parameter-to-observable forward mapping $\mathcal{G} : u_0 \to \mathcal{O}u$ maps an initial condition $u_0 \in L^2(\Omega)$ to pointwise spatiotemporal observations of the concentration field $u(\mathbf{x}, t)$ through the solution of the following advection-diffusion equation [76, 94]:

$$\begin{aligned} u_t - \kappa \Delta u + \mathbf{v} \cdot \nabla u = 0 \quad & in\ \Omega \times (0, T) \\ u(\cdot, 0) = u_0 \quad & in\ \Omega \\ \kappa \nabla u \cdot \vec{n} = 0, \quad & on\ \partial\Omega \times (0, T) \end{aligned} \tag{5.2}$$

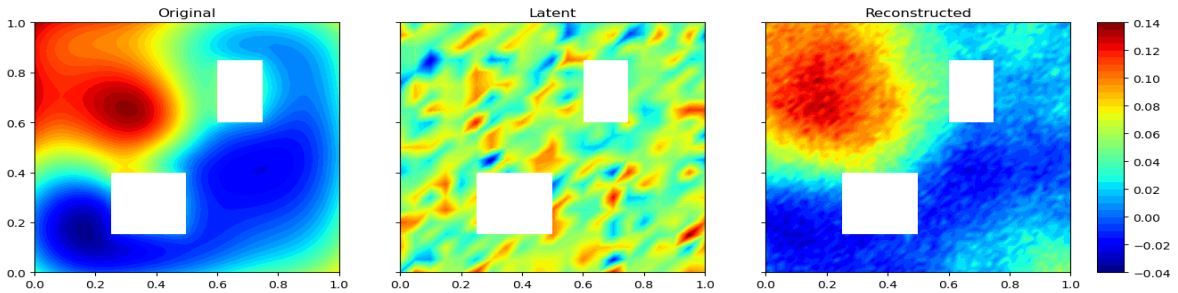
where $\Omega \subset [0, 1]^2$ is a bounded domain shown in Figure 12, $\kappa > 0$ is the diffusion coefficient (set to 10^{-3}), and $T > 0$ is the final time. The velocity field $\mathbf{v}$ is computed by solving the following steady-state Navier-Stokes equation with the side walls driving the flow [76]:

$$\begin{aligned} -\frac{1}{\mathrm{Re}} \Delta \mathbf{v} + \nabla q + \mathbf{v} \cdot \nabla \mathbf{v} = 0 \quad & in\ \Omega \\ \nabla \cdot \mathbf{v} = 0 \quad & in\ \Omega \\ \mathbf{v} = \mathbf{g}, \quad & on\ \partial\Omega \end{aligned} \tag{5.3}$$

Here, q is the pressure, Re is the Reynolds number, which is set to 100 in this example. The Dirichlet boundary data $\mathbf{g} \in \mathbb{R}^2$ is given by $\mathbf{g} = \mathbf{e}_2 = (0, 1)$ on the left wall of the domain, $\mathbf{g} = -\mathbf{e}_2$ on the right wall, and $\mathbf{g} = \mathbf{0}$ everywhere else.



(a) CNN (middle) and DNN (right) emulation ($\mathcal{G}^e : \mathbb{R}^{3413} \to \mathbb{R}^{1280}$) extracting gradients $D\Phi^e(u^{\text{MAP}})$ compared with the true gradient $D\Phi(u^{\text{MAP}})$ (left).



(b) AE compressing the original function u^{MAP} (left) into latent space u_r^{MAP} (middle) and reconstructing it in the original space $u^{\text{MAP}'}$ (right).

Figure 13: Advection-diffusion inverse problem: outputs by NNs viewed as 2d images.

We set the true initial condition $u_0^\dagger = 0.5 \wedge \exp\{-100[(x_1-0.35)^2+(x_2-0.7)^2]\}$, illustrated in the top left panel of Figure 12, which also shows a few snapshots of solutions u at other time points on a regular grid mesh of size 61×61. To obtain spatiotemporal observations, we collect solutions $u(\mathbf{x}, t)$ solved on a refined mesh at 80 selected locations $\{\mathbf{x}_i\}_{i=1}^{80}$ (Figure **??**) across 16 time points $\{t_j\}_{j=1}^{16}$ evenly distributed between 1 and 4 seconds (thus denoted as $\mathcal{O}u$) and inject some Gaussian noise $\mathcal{N}(0, \sigma_\eta^2)$ such that the relative noise standard deviation is 0.5 ($\sigma_\eta / \max \mathcal{O}u = 0.5$):

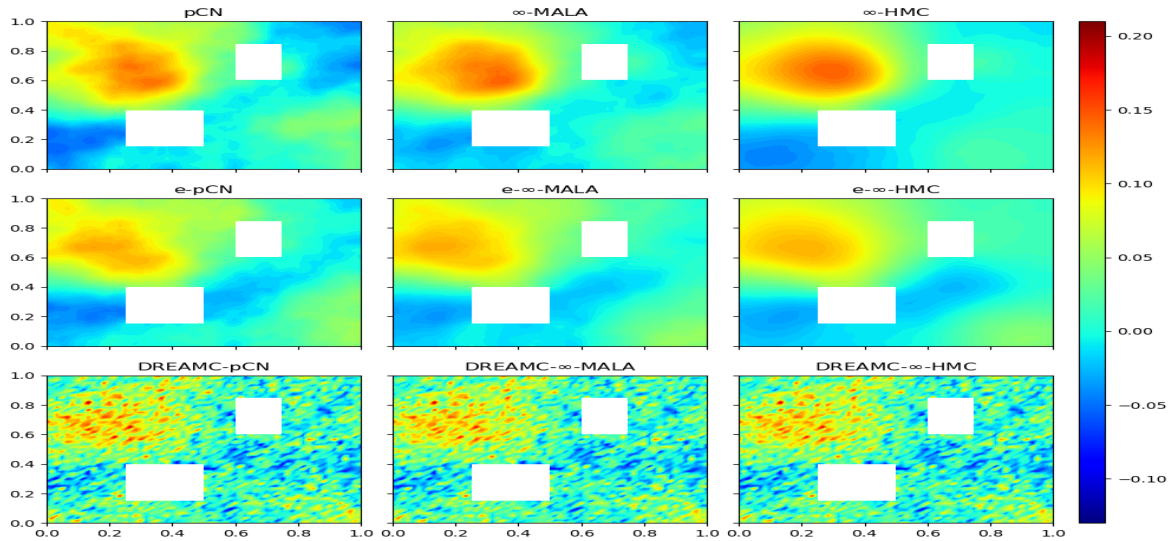
$$y = \mathcal{O}u(\mathbf{x}, t) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma_\eta^2 I_{1280})$$

In the Bayesian setting, we adopt the following GP prior with the covariance kernel $\mathcal{C}$ defined through the Laplace operator Δ:

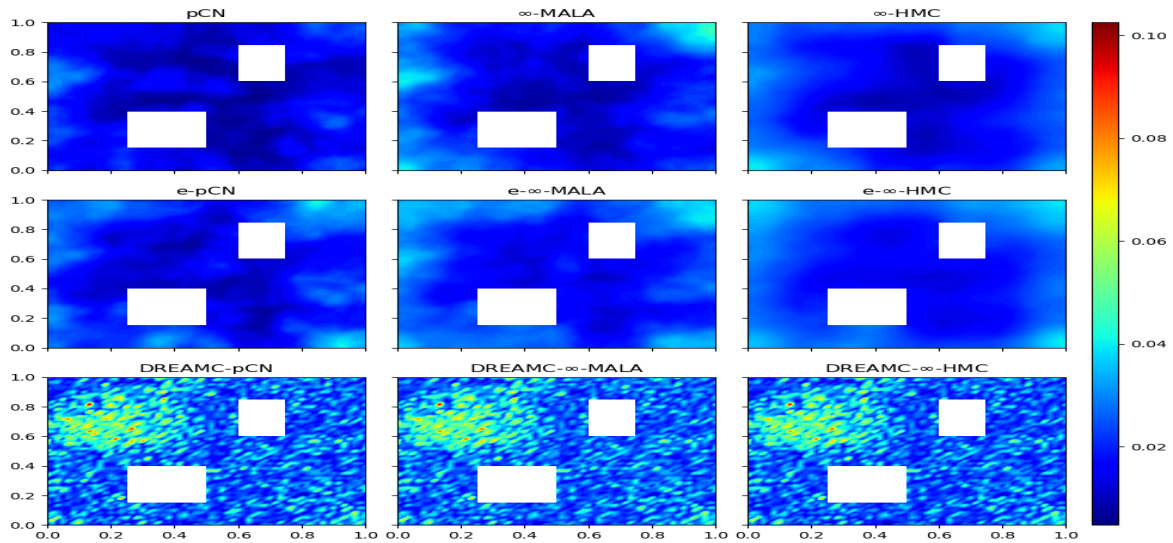
$$u \sim \mu_0 = \mathcal{N}(0, \mathcal{C}), \quad \mathcal{C} = (\delta \mathcal{I} - \gamma \Delta)^{-2}$$

where δ governs the variance of the prior and γ/δ controls the correlation length. We set $\gamma = 2$ and $\delta = 10$ in this example.

The Bayesian inverse problem involves obtaining an estimate of the initial condition u_0 and quantifying its uncertainty based on the 80×16 spatiotemporal observations $y \in \mathbb{R}^{1280}$. For the



(a) Posterior mean estimates of the initial concentration field $u(\mathbf{x})$.



(b) Posterior standard deviation estimates of the initial concentration field $u(\mathbf{x})$.

Figure 14: Advection-diffusion inverse problem: Bayesian posterior estimates of the initial concentration field $u(\mathbf{x})$ based on 5000 samples by various MCMC algorithms.

notational convenience, we still denote $u_0(\mathbf{x})$ as $u(\mathbf{x})$ when it is not confused with the general concentration field $u(\mathbf{x}, t)$. The Bayesian UQ in this example is especially challenging not only because of its large dimensionality (3413) of spatially discretized u (Lagrange degree 1) at each time t, but also due to the spatiotemporal interactions in these observations [18]. We follow the CES framework as in Section 3.3. In the calibration stage, we collect $\{u_n^{(j)}, \mathcal{G}(u_n^{(j)})\}_{j=1,n=1}^{J,N}$

by running EKS with the ensemble size $J = 500$ for $N = 10$ iterations. For the emulation, we train DNN and CNN with the same 3 : 1 splitting of these 5000 ensembles for training and testing. The DNN has 5 layers with activation function 'LeakyReLU($\alpha = 0.01$)' for the hidden layers and 'linear' activation for the output layer and 25% nodes dropped out. The structure of CNN is illustrated in Figure 2 with 4 filters in the last convolution layer, activation 'LeakyReLU($\alpha = 0.2$)' for the convolution layers, 'PReLU' activation for the latent layer (dimension 1024), and 'linear' activation for the output layer. The trained CNN has drop out rate of 50% on all its nodes. Figure 13a compares the true gradient function $D\Phi(u^{\text{MAP}})$ (left panel) and its emulations $D\Phi^e(u_{\text{MAP}})$ (middle and right panels) as in Equation (3.4) extracted from two types of NNs. As before, we can see better extracted gradient output by CNN as an approximation to the true gradient compared with DNN. Due to the large dimensionality of inputs and outputs ($\mathcal{G}^e : \mathbb{R}^{3413} \to \mathbb{R}^{1280}$) and memory requirement, GP (implemented in `GPflow`) failed to fit and output gradient extraction.

In the sampling stage, we adopt AE with the same structure as in Figure 4, the latent dimension $d_L = 417$ (degrees of freedom on submesh 21×21), and the activation functions chosen as 'elu'. Figure 13b plots the original u^{MAP} (left), the latent representation $u_r^{\text{MAP}} = \phi(u^{\text{MAP}})$ (middle) and the reconstruction $u^{\text{MAP}'} = \psi(u_r^{\text{MAP}})$ (right). Again we can see a 'faithful' reconstruction of the original function (image) by AE even though the latent representation is not very intuitive.

Method	h [a]	AP [b]	s/iter [c]	ESS(min,med,max) [d]	minESS/s [e]	spdup [f]	PDEsolns [g]
pCN	0.001	0.69	0.03	(3.16,6.37,40.7)	0.0222	1.00	6001
∞-MALA	0.005	0.68	0.06	(3.78,11.6,51.5)	0.0122	0.55	12002
∞-HMC	0.005	0.78	0.12	(31.55,83.54,240.34)	0.0507	2.29	35872
e-pCN	0.002	0.69	0.02	(3.33,7.19,58.2)	0.0324	1.46	0
e-∞-MALA	0.008	0.72	0.05	(4.28,14.3,62)	0.0157	0.71	0
e-∞-HMC	0.008	0.72	0.11	(25.41,113.11,270.79)	0.0475	2.14	0
DREAMC-pCN	0.020	0.68	0.02	(8.88,16.99,53.35)	0.0727	**3.28**	0
DREAMC-∞-MALA	0.100	0.83	0.06	(37.65,66.58,157.09)	0.1310	**5.91**	0
DREAMC-∞-HMC	0.100	0.72	0.17	(564.12,866.72,1292.11)	0.6791	**30.64**	0

[a] step size [b] acceptance probability [c] seconds per iteration [d] (minimum, median, maximum) effective sample size [e] minimal ESS per second [f] comparison of minESS/s with pCN as benchmark [g] number of PDE solutions

Table 2: Advection-diffusion inverse problem: sampling efficiency of MCMC algorithms.

We compare the performance of ∞-MCMC algorithms (pCN, ∞-MALA, ∞-HMC), their emulative versions, and the corresponding DREAMC algorithms. For each algorithm, we run 6000 iterations and burn in the first 1000. For HMC algorithms, we set $I = 5$. We tune the step sizes for each algorithm so that they have similar acceptance rates around $60 \sim 70\%$. Figure 14a compares their posterior mean estimates and Figure 14b compares their estimates of posterior standard deviation. We can see that emulative MCMC algorithms generate similar results as the original MCMC methods. DREAMC algorithms yield estimates close enough to those by the original MCMC. Although there are some deviations in the uncertainty estimates, the results by DREAMC algorithms are significantly better than those by ensemble Kalman methods, which severely underestimate the posterior standard deviations (See Figure **??**).

Table 2 compares the sampling efficiency of various MCMC algorithms measured by minESS/s. Three most efficient sampling algorithms are all DREAMC algorithms. DREAMC

∞-HMC attains up to 30 times speed up compared to the benchmark pCN. Considering the complexity of this inverse problem with spatiotemporal observations, this is a significant achievement. Again, we exclude the training time of CNN and AE from the comparison since it is rather negligible compared with the overall sampling time.

In Appendix **??**, Figure **??** verifies DREAMC ∞-HMC is the most efficient MCMC algorithm that has the smallest autocorrelation shown on the right panel. It is followed by other HMC algorithms and DREAMC ∞-MAMA which is even better than ∞-HMC. Figure **??** plots the KL divergence between the posterior and the prior in terms of iteration (upper) and time (lower) respectively. As we can see, ∞-HMC converges the fastest.

6. Conclusion. In this paper, we have proposed a new framework to scale up Bayesian UQ for inverse problems. More specifically, we use CNN – a regularized neural network, which is a powerful tool for image recognition and amenable to inverse problems if we treat the discretized parameter function as an input image. This way, CNN is capable of learning spatial features. In addition, the resulting algorithm has low computational complexity and is robust: as seen in Figure 3, the performance of CNN as an emulator is relatively stable across different training sizes. If larger training size is required for certain problems, we could train CNN adaptively as more samples are collected from the parameter space [88]. We have adopted AE to further reduce the dimension of the parameter space and speed up the sampling process. Overall, by utilizing different techniques based on neural networks, we have been able to scale up Bayesian UQ up to thousands of dimensions.

In the current framework, we rely on regular grid mesh to facilitate the CNN training – a discretized function over a 2d mesh needs to be converted to a matrix of image pixels. Such a limitation can be alleviated by using mesh CNN [37], which could train CNN directly on irregular mesh; for example, triangular mesh has been extensively used for solving PDE. This will extend our methodology and further enhances its utility.

The standard AE used in our proposed framework and the corresponding latent projection by dense layers might not be the optimal choices. Alternatively, we could use convolutional AE (CAE) [36], which generates more recognizable latent representation as illustrated in Figure **??**. In this case, the latent parameter can be interpreted as a representation of the original function on a coarser mesh. What is more, we could modify the loss function to adaptively learn the dimensionality of intrinsic latent space.

There are spatiotemporal data in some inverse problems (e.g., advection-diffusion equation). In such cases, we could model the temporal pattern of observations in the emulation using some recurrent neural networks (RNN) [81], e.g., long short-term memory (LSTM) [42]. We can then build a 'CNN-RNN' emulator with convolutional layer for function (image) inputs and RNN layer for multivariate time series outputs. While we have obtained promising preliminary results (See Figure **??**), we intend to pursue this idea in a follow-up paper. Lastly, future research could involve replacing the MCMC algorithms in the sampling step with some recently proposed information gradient flow methods [97, 96] and comparing the performance with the current DREAMC approach.

Acknowledgments. Shahbaba is supported by NSF grants DMS-1936833 and DMS-1763272, and NIH grant R01-MH115697.

REFERENCES

[1] N. G. O. D. S. R. A. C. . V. B. Aanonsen, S. I., *The ensemble kalman filter in reservoir engineering–a review*, Society of Petroleum Engineers, 14 (2009).

[2] A. R. Barron, *Neural net approximation*, in 7th Yale Workshop on Adaptive and Learning Systems, Yale University, May 1992.

[3] Y. Bengio, *Learning deep architectures for AI*, Foundations and Trends® in Machine Learning, 2 (2009), pp. 1–127, https://doi.org/10.1561/2200000006, https://doi.org/10.1561%2F2200000006.

[4] A. Beskos, *A stable manifold MCMC method for high dimensions*, Statistics & Probability Letters, 90 (2014), pp. 46–52.

[5] A. Beskos, M. Girolami, S. Lan, P. E. Farrell, and A. M. Stuart, *Geometric mcmc for infinite-dimensional inverse problems*, Journal of Computational Physics, 335 (2017), pp. 327 – 351, http://www.sciencedirect.com/science/article/pii/S0021999116307033.

[6] A. Beskos, F. J. Pinski, J. M. Sanz-Serna, and A. M. Stuart, *Hybrid Monte-Carlo on Hilbert spaces*, Stochastic Processes and their Applications, 121 (2011), pp. 2201–2230.

[7] A. Beskos, G. Roberts, and A. Stuart, *Optimal scalings for local metropolis–hastings chains on nonproduct targets in high dimensions*, The Annals of Applied Probability, 19 (2009), pp. 863–898, https://doi.org/10.1214/08-aap563, https://doi.org/10.1214%2F08-aap563.

[8] A. Beskos, G. Roberts, A. Stuart, and J. Voss, *MCMC methods for diffusion bridges*, Stochastics and Dynamics, 8 (2008), pp. 319–350.

[9] L. Bottou, *On-line learning and stochastic approximations*, in On-Line Learning in Neural Networks, Cambridge University Press, jan 1999, pp. 9–42, https://doi.org/10.1017/cbo9780511569920.003, https://doi.org/10.1017%2Fcbo9780511569920.003.

[10] L. Bottou and Y. Cun, *Large scale online learning*, in Advances in Neural Information Processing Systems 16 (NIPS 2003), S. Thrun, L. Saul, and B. Schölkopf, eds., vol. 16, MIT Press, 2004, https://proceedings.neurips.cc/paper/2003/file/9fb7b048c96d44a0337f049e0a61ff06-Paper.pdf.

[11] N. K. Chada, A. M. Stuart, and X. T. Tong, *Tikhonov regularization within ensemble kalman inversion*, 2019, https://arxiv.org/abs/1901.10382.

[12] Y. Chen and D. S. Oliver, *Ensemble randomized maximum likelihood method as an iterative ensemble smoother*, Mathematical Geosciences, 44 (2011), pp. 1–26, https://doi.org/10.1007/s11004-011-9376-z, http://dx.doi.org/10.1007/s11004-011-9376-z.

[13] D. Chicco, P. Sadowski, and P. Baldi, *Deep autoencoder neural networks for gene ontology annotation predictions*, in Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, ACM, sep 2014, https://doi.org/10.1145/2649387.2649442, https://doi.org/10.1145%2F2649387.2649442.

[14] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, *Multi-column deep neural network for traffic sign classification*, Neural Networks, 32 (2012), pp. 333 – 338, https://doi.org/https://doi.org/10.1016/j.neunet.2012.02.023, http://www.sciencedirect.com/science/article/pii/S0893608012000524. Selected Papers from IJCNN 2011.

[15] E. Cleary, A. Garbuno-Inigo, S. Lan, T. Schneider, and A. M. Stuart, *Calibrate, emulate, sample*, 2020, https://arxiv.org/abs/2001.03689.

[16] P. Constantine, C. Kent, and T. Bui-Thanh, *Accelerating markov chain monte carlo with active subspaces*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2779–A2805, https://doi.org/10.1137/15M1042127, https://doi.org/10.1137/15M1042127.

[17] S. L. Cotter, G. O. Roberts, A. Stuart, and D. White, *MCMC methods for functions: modifying old algorithms to make them faster*, Statistical Science, 28 (2013), pp. 424–446.

[18] N. Cressie and C. Wikle, *Statistics for Spatio-Temporal Data*, CourseSmart Series, Wiley, 2011, https://books.google.com/books?id=-kOC6D0DiNYC.

[19] T. Cui, K. J. Law, and Y. M. Marzouk, *Dimension-independent likelihood-informed MCMC*, Journal of Computational Physics, 304 (2016), pp. 109 – 137.

[20] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals, and Systems, 2 (1989), pp. 303–314, https://doi.org/10.1007/bf02551274, https://doi.org/10.1007%2Fbf02551274.

[21] M. Dashti and A. M. Stuart, *The Bayesian Approach to Inverse Problems*, Springer International

Publishing, Cham, 2017, pp. 311–428, https://doi.org/10.1007/978-3-319-12385-1_7, https://doi.org/10.1007/978-3-319-12385-1_7.

[22] J. de Wiljes, S. Reich, and W. Stannat, *Long-time stability and accuracy of the ensemble kalman–bucy filter for fully observed processes and small measurement noise*, SIAM Journal on Applied Dynamical Systems, 17 (2018), pp. 1152–1181, https://doi.org/10.1137/17M1119056, https://doi.org/10.1137/17M1119056, https://arxiv.org/abs/https://doi.org/10.1137/17M1119056.

[23] A. A. Emerick and A. C. Reynolds, *Investigation of the sampling performance of ensemble-based methods with a simple reservoir model*, Computational Geosciences, 17 (2013), pp. 325–350, https://doi.org/10.1007/s10596-012-9333-z, http://dx.doi.org/10.1007/s10596-012-9333-z.

[24] G. Evensen, *Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics*, Journal of Geophysical Research, 99 (1994), p. 10143, https://doi.org/10.1029/94jc00572, http://dx.doi.org/10.1029/94JC00572.

[25] G. Evensen, *The ensemble Kalman filter: theoretical formulation and practical implementation*, Ocean Dyn., 53 (2003), pp. 343–367, https://doi.org/10.1007/s10236-003-0036-9.

[26] G. Evensen, *Data Assimilation: The Ensemble Kalman Filter*, Springer-Verlag Berlin Heidelberg, 2 ed., 2009.

[27] G. Evensen, *Analysis of iterative ensemble smoothers for solving inverse problems*, Computational Geosciences, 22 (2018), pp. 885–908, https://doi.org/10.1007/s10596-018-9731-y, http://dx.doi.org/10.1007/s10596-018-9731-y.

[28] G. Evensen and P. J. van Leeuwen, *Assimilation of geosat altimeter data for the agulhas current using the ensemble kalman filter with a quasi-geostrophic model*, 1996.

[29] G. Evensen and P. J. van Leeuwen, *Assimilation of geosat altimeter data for the agulhas current using the ensemble kalman filter with a quasigeostrophic model*, Monthly Weather Review, 124 (1996), pp. 85–96, https://doi.org/10.1175/1520-0493(1996)124⟨0085:aogadf⟩2.0.co;2, https://doi.org/10.1175%2F1520-0493%281996%29124%3C0085%3Aaogadf%3E2.0.co%3B2.

[30] K. Fukushima, *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*, Biological Cybernetics, 36 (1980), pp. 193–202, https://doi.org/10.1007/BF00344251, https://doi.org/10.1007/BF00344251.

[31] A. Garbuno-Inigo, F. Hoffmann, W. Li, and A. M. Stuart, *Interacting langevin diffusions: Gradient structure and ensemble kalman sampler*, SIAM Journal on Applied Dynamical Systems, 19 (2020), pp. 412–441, https://doi.org/10.1137/19M1251655, https://doi.org/10.1137/19M1251655, https://arxiv.org/abs/https://doi.org/10.1137/19M1251655.

[32] A. Garbuno-Inigo, N. Nüsken, and S. Reich, *Affine invariant interacting langevin dynamics for bayesian inference*, SIAM Journal on Applied Dynamical Systems, 19 (2020), pp. 1633–1658, https://doi.org/10.1137/19m1304891, https://doi.org/10.1137%2F19m1304891.

[33] F. Gers and E. Schmidhuber, *LSTM recurrent networks learn simple context-free and context-sensitive languages*, IEEE Transactions on Neural Networks, 12 (2001), pp. 1333–1340, https://doi.org/10.1109/72.963769, https://doi.org/10.1109%2F72.963769.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. http://www.deeplearningbook.org.

[35] A. Graves, D. Eck, N. Beringer, and J. Schmidhuber, *Biologically plausible speech recognition with LSTM neural nets*, in Biologically Inspired Approaches to Advanced Information Technology, Springer Berlin Heidelberg, 2004, pp. 127–136, https://doi.org/10.1007/978-3-540-27835-1_10, https://doi.org/10.1007%2F978-3-540-27835-1_10.

[36] X. Guo, X. Liu, E. Zhu, and J. Yin, *Deep clustering with convolutional autoencoders*, in Neural Information Processing, Springer International Publishing, 2017, pp. 373–382, https://doi.org/10.1007/978-3-319-70096-0_39, https://doi.org/10.1007%2F978-3-319-70096-0_39.

[37] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, *Meshcnn: A network with an edge*, ACM Trans. Graph., 38 (2019), https://doi.org/10.1145/3306346.3322959, https://doi.org/10.1145/3306346.3322959.

[38] J. Hensman, A. Matthews, and Z. Ghahramani, *Scalable Variational Gaussian Process Classification*, in Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, G. Lebanon and S. V. N. Vishwanathan, eds., vol. 38 of Proceedings of Machine Learning Research, San Diego, California, USA, 09–12 May 2015, PMLR, pp. 351–360,

https://proceedings.mlr.press/v38/hensman15.html.

[39] D. Higdon, M. Kennedy, J. C. Cavendish, J. A. Cafeo, and R. D. Ryne, *Combining field data and computer simulations for calibration and prediction*, SIAM Journal on Scientific Computing, 26 (2004), pp. 448–466, https://doi.org/10.1137/S1064827503426693, https://doi.org/10.1137/S1064827503426693, https://arxiv.org/abs/https://doi.org/10.1137/S1064827503426693.

[40] G. Hinton and R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science, 313 (2006), pp. 504–507.

[41] G. E. Hinton, S. Osindero, and Y.-W. Teh, *A fast learning algorithm for deep belief nets*, Neural Computation, 18 (2006), pp. 1527–1554, https://doi.org/10.1162/neco.2006.18.7.1527, https://doi.org/10.1162/neco.2006.18.7.1527, https://arxiv.org/abs/https://doi.org/10.1162/neco.2006.18.7.1527. PMID: 16764513.

[42] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Comput., 9 (1997), pp. 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735.

[43] P. L. Houtekamer and H. L. Mitchell, *A sequential ensemble kalman filter for atmospheric data assimilation*, Monthly Weather Review, 129 (2001), pp. 123–137, https://doi.org/10.1175/1520-0493(2001)129⟨0123:asekff⟩2.0.co;2, http://dx.doi.org/10.1175/1520-0493(2001)129⟨0123:ASEKFF⟩2.0.CO;2.

[44] P. L. Houtekamer and F. Zhang, *Review of the ensemble kalman filter for atmospheric data assimilation*, Monthly Weather Review, 144 (2016), pp. 4489–4532, https://doi.org/10.1175/mwr-d-15-0440.1, http://dx.doi.org/10.1175/MWR-D-15-0440.1.

[45] M. A. Iglesias, *A regularizing iterative ensemble kalman method for pde-constrained inverse problems*, Inverse Problems, 32 (2016), p. 025002, https://doi.org/10.1088/0266-5611/32/2/025002, http://dx.doi.org/10.1088/0266-5611/32/2/025002.

[46] M. A. Iglesias, K. J. H. Law, and A. M. Stuart, *Ensemble kalman methods for inverse problems*, Inverse Problems, 29 (2013), p. 045001, https://doi.org/10.1088/0266-5611/29/4/045001, http://dx.doi.org/10.1088/0266-5611/29/4/045001.

[47] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, *Deep convolutional neural network for inverse problems in imaging*, IEEE Transactions on Image Processing, 26 (2017), pp. 4509–4522, https://doi.org/10.1109/TIP.2017.2713099.

[48] M. Jurek and M. Katzfuss, *Multi-resolution filters for massive spatio-temporal data*, Journal of Computational and Graphical Statistics, (2021), pp. 1–16, https://doi.org/10.1080/10618600.2021.1886938, https://doi.org/10.1080%2F10618600.2021.1886938.

[49] R. E. Kalman, *A new approach to linear filtering and prediction problems*, Journal of Basic Engineering, 82 (1960), p. 35, https://doi.org/10.1115/1.3662552, http://dx.doi.org/10.1115/1.3662552.

[50] E. Kalnay, *Atmospheric modeling, data assimilation and predictability*, (2002), https://doi.org/10.1017/cbo9780511802270, http://dx.doi.org/10.1017/CBO9780511802270.

[51] M. Katzfuss, J. R. Stroud, and C. K. Wikle, *Ensemble kalman methods for high-dimensional hierarchical dynamic space-time models*, Journal of the American Statistical Association, 115 (2019), pp. 866–885, https://doi.org/10.1080/01621459.2019.1592753, https://doi.org/10.1080%2F01621459.2019.1592753.

[52] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, Journal of the Royal Statistical Society, Series B, Methodological, 63 (2001), pp. 425–464.

[53] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, in 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014, https://arxiv.org/abs/http://arxiv.org/abs/1312.6114v10.

[54] D. P. Kingma and M. Welling, *An introduction to variational autoencoders*, Foundations and TrendsÂ® in Machine Learning, 12 (2019), pp. 307–392, https://doi.org/10.1561/2200000056, http://dx.doi.org/10.1561/2200000056.

[55] K. Z. Kody Law, Andrew Stuart, *Data Assimilation: A Mathematical Introduction*, vol. 62 of Texts in Applied Mathematics, Springer International Publishing, 1 ed., 2015.

[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105, https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[57] S. Lan, *Adaptive dimension reduction to accelerate infinite-dimensional geometric markov chain monte carlo*, Journal of Computational Physics, 392 (2019), pp. 71 – 95, https://doi.org/https://doi.org/10.1016/j.jcp.2019.04.043, http://www.sciencedirect.com/science/article/pii/S002199911930289X.

[58] S. Lan, T. Bui-Thanh, M. Christie, and M. Girolami, *Emulation of higher-order tensors in manifold Monte Carlo methods for Bayesian inverse problems*, Journal of Computational Physics, 308 (2016), pp. 81–101.

[59] S. Lan, V. Stathopoulos, B. Shahbaba, and M. Girolami, *Markov Chain Monte Carlo from Lagrangian Dynamics*, Journal of Computational and Graphical Statistics, 24 (2015), pp. 357–378.

[60] S. Lan, B. Zhou, and B. Shahbaba, *Spherical hamiltonian monte carlo for constrained target distributions*, in Proceedings of The 31st International Conference on Machine Learning, Beijing, China, 2014, pp. 629–637.

[61] K. Law, *Proposals which speed up function-space MCMC*, Journal of Computational and Applied Mathematics, 262 (2014), pp. 127–138.

[62] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444, https://doi.org/10.1038/nature14539, https://doi.org/10.1038%2Fnature14539.

[63] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural Computation, 1 (1989), pp. 541–551, https://doi.org/10.1162/neco.1989.1.4.541.

[64] J. Lee, J. Sohl-dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri, *Deep neural networks as gaussian processes*, in International Conference on Learning Representations, 2018, https://openreview.net/forum?id=B1EA-M-0Z.

[65] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, *The expressive power of neural networks: A view from the width*, in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., vol. 30, Curran Associates, Inc., 2017, pp. 6231–6239, https://proceedings.neurips.cc/paper/2017/file/32cbf687880eb1674a07bf717761dd3a-Paper.pdf.

[66] M. T. McCann, K. H. Jin, and M. Unser, *Convolutional neural networks for inverse problems in imaging: A review*, IEEE Signal Processing Magazine, 34 (2017), pp. 85–95, https://doi.org/10.1109/MSP.2017.2739299.

[67] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig, *Using recurrent neural networks for slot filling in spoken language understanding*, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 23 (2015), pp. 530–539, https://doi.org/10.1109/TASLP.2014.2383614.

[68] E. T. Nalisnick, *On Priors for Bayesian Neural Networks*, PhD thesis, University of California-Irvine, 2018.

[69] R. M. Neal, *Bayesian Learning for Neural Networks*, Springer New York, 1996, https://doi.org/10.1007/978-1-4612-0745-0, https://doi.org/10.1007%2F978-1-4612-0745-0.

[70] R. M. Neal, *MCMC using Hamiltonian dynamics*, in Handbook of Markov Chain Monte Carlo, S. Brooks, A. Gelman, G. Jones, and X. L. Meng, eds., Chapman and Hall/CRC, 2010.

[71] J. Oakley and A. O'Hagan, *Bayesian inference for the uncertainty distribution of computer model outputs*, Biometrika, 89 (2002), pp. 769–784, https://doi.org/10.1093/biomet/89.4.769, https://doi.org/10.1093/biomet/89.4.769, https://arxiv.org/abs/http://oup.prod.sis.lan/biomet/article-pdf/89/4/769/667202/890769.pdf.

[72] J. E. Oakley and A. O'Hagan, *Probabilistic sensitivity analysis of complex models: A bayesian approach*, Journal of the Royal Statistical Society. Series B (Statistical Methodology), 66 (2004), pp. 751–769, http://www.jstor.org/stable/3647504.

[73] A. O'Hagan, *Bayesian analysis of computer code outputs: A tutorial*, Reliability Engineering & System Safety, 91 (2006), pp. 1290 – 1300, https://doi.org/https://doi.org/10.1016/j.ress.2005.11.025, http://www.sciencedirect.com/science/article/pii/S0951832005002383. The Fourth International Conference on Sensitivity Analysis of Model Output (SAMO 2004).

[74] D. S. Oliver, A. C. Reynolds, and N. Liu, *Inverse theory for petroleum reservoir characterization and history matching*, (2008), https://doi.org/10.1017/cbo9780511535642, http://dx.doi.org/10.1017/CBO9780511535642.

[75] M. Opper and C. Archambeau, *The variational gaussian approximation revisited*, Neural Compu-

tation, 21 (2009), pp. 786–792, https://doi.org/10.1162/neco.2008.08-07-592, https://doi.org/10.1162%2Fneco.2008.08-07-592.

[76] N. Petra and G. Stadler, *Model variational inverse problems governed by partial differential equations*, tech. report, The Institute for Computational Engineering and Sciences, The University of Texas at Austin., 2011.

[77] A. Pinkus, *Approximation theory of the MLP model in neural networks*, Acta Numerica, 8 (1999), pp. 143–195, https://doi.org/10.1017/s0962492900002919, https://doi.org/10.1017%2Fs0962492900002919.

[78] M. Ribeiro, A. E. Lazzaretti, and H. S. Lopes, *A study of deep convolutional auto-encoders for anomaly detection in videos*, Pattern Recognition Letters, 105 (2018), pp. 13 – 22, https://doi.org/https://doi.org/10.1016/j.patrec.2017.07.016, http://www.sciencedirect.com/science/article/pii/S0167865517302489. Machine Learning and Applications in Artificial Intelligence.

[79] G. O. Roberts, A. Gelman, and W. R. Gilks, *Weak convergence and optimal scaling of random walk metropolis algorithms*, The Annals of Applied Probability, 7 (1997), pp. 110–120.

[80] G. O. Roberts and J. S. Rosenthal, *Optimal scaling of discrete approximations to langevin diffusions*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 60 (1998), pp. 255–268, https://doi.org/https://doi.org/10.1111/1467-9868.00123, https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00123, https://arxiv.org/abs/https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00123.

[81] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature, 323 (1986), pp. 533–536, https://doi.org/10.1038/323533a0, https://doi.org/10.1038%2F323533a0.

[82] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statistical Science, 4 (1989), pp. 409–423, https://doi.org/10.1214/ss/1177012413, http://dx.doi.org/10.1214/ss/1177012413.

[83] T. J. Santner, B. J. Williams, and W. Notz, *The design and analysis of computer experiments*, Springer, 2003.

[84] C. Schillings and A. Stuart, *Analysis of the ensemble kalman filter for inverse problems*, SIAM Journal on Numerical Analysis, 55 (2017), pp. 1264–1290, https://doi.org/10.1137/16M105959X, https://doi.org/10.1137/16M105959X, https://arxiv.org/abs/https://doi.org/10.1137/16M105959X.

[85] C. Schillings and A. M. Stuart, *Convergence analysis of ensemble kalman inversion: the linear, noisy case*, Applicable Analysis, 97 (2017), pp. 107–123, https://doi.org/10.1080/00036811.2017.1386784, http://dx.doi.org/10.1080/00036811.2017.1386784.

[86] J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks, 61 (2015), pp. 85 – 117, https://doi.org/https://doi.org/10.1016/j.neunet.2014.09.003, http://www.sciencedirect.com/science/article/pii/S0893608014002135.

[87] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, *Improved protein structure prediction using potentials from deep learning*, Nature, 577 (2020), pp. 706–710, https://doi.org/10.1038/s41586-019-1923-7, https://doi.org/10.1038%2Fs41586-019-1923-7.

[88] B. Shahbaba, L. M. Lomeli, T. Chen, and S. Lan, *Deep markov chain monte carlo*. arXiv:1910.05692, 2019, https://arxiv.org/abs/1910.05692.

[89] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, *Mastering the game of go with deep neural networks and tree search*, Nature, 529 (2016), pp. 484–489, https://doi.org/10.1038/nature16961, https://doi.org/10.1038%2Fnature16961.

[90] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, *Mastering the game of go without human knowledge*, Nature, 550 (2017), pp. 354–359, https://doi.org/10.1038/nature24270, https://doi.org/10.1038%2Fnature24270.

[91] G. Stephenson, *Using derivative information in the statistical analysis of computer models*, PhD thesis, University of Southampton, 2010.
[92] A. M. Stuart, *Inverse problems: a bayesian perspective*, Acta Numerica, 19 (2010), pp. 451–559.
[93] L. Verlet, *Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules*, Phys. Rev., 159 (1967), pp. 98–103.
[94] U. Villa, N. Petra, and O. Ghattas, *hippylib: An extensible software framework for large-scale inverse problems governed by pdes; part i: Deterministic inversion and linearized bayesian inference*, 2020, https://arxiv.org/abs/1909.03948.
[95] F. Wang, A. Eljarrat, J. Müller, T. R. Henninen, R. Erni, and C. T. Koch, *Multi-resolution convolutional neural networks for inverse problems*, Scientific Reports, 10 (2020), https://doi.org/10.1038/s41598-020-62484-z, https://doi.org/10.1038%2Fs41598-020-62484-z.
[96] Y. Wang and W. Li, *Information newton's flow: second-order optimization method in probability space*, 2020, https://arxiv.org/abs/2001.04341.
[97] Y. Wang and W. Li, *Accelerated information gradient flow*, Journal of Scientific Computing, 90 (2021), https://doi.org/10.1007/s10915-021-01709-3, https://doi.org/10.1007%2Fs10915-021-01709-3.
[98] M. Welling and Y. W. Teh, *Bayesian learning via stochastic gradient Langevin dynamics*, in Proceedings of the International Conference on Machine Learning, 2011.
[99] A. Zammit-Mangion and C. K. Wikle, *Deep integro-difference equation models for spatio-temporal forecasting*, Spatial Statistics, 37 (2020), p. 100408, https://doi.org/https://doi.org/10.1016/j.spasta.2020.100408, https://www.sciencedirect.com/science/article/pii/S2211675320300026. Frontiers in Spatial and Spatio-temporal Research.
[100] D.-X. Zhou, *Universality of deep convolutional neural networks*, Applied and Computational Harmonic Analysis, 48 (2020), pp. 787 – 794, https://doi.org/https://doi.org/10.1016/j.acha.2019.06.004, http://www.sciencedirect.com/science/article/pii/S1063520318302045.