

AN ACCELERATION STRATEGY FOR RANDOMIZE-THEN-OPTIMIZE SAMPLING VIA DEEP NEURAL NETWORKS*

Liang Yan

*School of Mathematics, Southeast University, Nanjing, 210096, China;
Nanjing Center for Applied Mathematics, Nanjing, 211135, China*

Email: yanliang@seu.edu.cn

Tao Zhou¹⁾

*LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of
Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*

Email: tzhou@lsec.cc.ac.cn

Abstract

Randomize-then-optimize (RTO) is widely used for sampling from posterior distributions in Bayesian inverse problems. However, RTO can be computationally intensive for complexity problems due to repetitive evaluations of the expensive forward model and its gradient. In this work, we present a novel goal-oriented deep neural networks (DNN) surrogate approach to substantially reduce the computation burden of RTO. In particular, we propose to draw the training points for the DNN-surrogate from a local approximated posterior distribution – yielding a flexible and efficient sampling algorithm that converges to the direct RTO approach. We present a Bayesian inverse problem governed by elliptic PDEs to demonstrate the computational accuracy and efficiency of our DNN-RTO approach, which shows that DNN-RTO can significantly outperform the traditional RTO.

Mathematics subject classification: 35R30, 62F15, 65C60, 68T05.

Key words: Bayesian inverse problems, Deep neural network, Markov chain Monte Carlo.

1. Introduction

The Bayesian approach provides a systematic framework for quantifying the uncertainty in parameter estimations of inverse problems [14, 26]. In the Bayesian approach, one combine a prior knowledge of the unknown parameters and the forward model to yield a *posterior* probability distribution, from which the statistic information of the unknown parameters can be characterized. The main task of the Bayesian approach is to draw samples from the posterior distributions, and then evaluate the associate statistic information, e.g., expectation, variance, etc. Since analytical formulas of the posterior are in general not available, many numerical sampling approaches such as Markov chain Monte Carlo (MCMC) methods [7] have been developed.

Nevertheless, the MCMC sampling scheme can be computationally challenging. Firstly, each evaluation of the system output involves a forward model evaluation, and this is infeasible if the model is expensive to evaluate. Secondly, the geometry of the posterior distribution may admits complex features in the parametric space (such as local concentration). One popular

* Received March 12, 2020 / Revised version received January 27, 2021 / Accepted February 22, 2021 /
Published online September 15, 2021 /

¹⁾ Corresponding author

way for reducing the computational complexity is to use the so-called surrogate approach: one constructs a surrogate to the true model and samples from the posterior distribution induced by the surrogate. In case the surrogate is computationally less expensive, one can dramatically speed up the MCMC algorithms. Different surrogate approaches have been investigated in recent years, for instance, projection-type reduced order models [1, 9, 18], polynomial chaos (PC) based surrogates [20–22, 31, 32], and Gaussian process regression [15, 25, 27], to name a few. Although the surrogate approach can be very effective when exploring low-dimensional distributions, it can still be inefficient for high-dimensional distributions with local concentration [7]. In these cases, the effective sample size (ESS) tends to be very low. To address this challenge, several strategies that combine the geometry information of the posterior (such as the gradient and Hessian information) have been exploited to accelerate the convergence of MCMC, see, e.g. [4, 11, 13, 17, 19]. We also mention recent progresses on adaptive multi-fidelity surrogate approaches [33–35].

In this work, we propose a new approach that combines a deep neural networks (DNN) surrogate and an optimization-based sampling approach for large-scale PDE constrained Bayesian inverse problems. In particular, we focus on the randomize-then-optimize (RTO) approach [2, 3, 30] which uses repeated solutions of a randomly perturbed optimization problem to produce samples from a non-Gaussian distribution (used as a Metropolis independence proposal). Compared to the classical Metropolis-Hastings (MH) random walk algorithm, RTO admits higher acceptance probability and lower sample auto-correlation even for high-dimensional problems [2]. The main drawback of the original RTO lies in the repetitive evaluation of the forward model and its gradient. To this end, we construct a DNN-based surrogate which makes the optimization problems rather efficient to solve. Moreover, to obtain an accurate and efficient DNN-surrogate, we propose to generate the training points from a local approximated posterior distribution, and this makes the training procedure very efficient.

We next summarize the main features of our DNN-RTO approach:

- A new approach that combines RTO and a DNN-surrogate. The new approach is expected to be promising for high dimensional problems.
- With the DNN-surrogate, the gradient information can be obtained efficiently by the backward propagation, so that the associated optimization procedure can be efficiently solved.
- We choose the training points from a local approximated posterior distribution, and this makes the training procedure (for the DNN-surrogate) very efficient.
- We present numerical examples to demonstrate that the DNN-RTO outperforms the traditional RTO.

The rest of the paper is organized as follows. In the next section, we give a brief introduction to the Bayesian inverse problems. In Section 3, we introduce the RTO algorithm. Details of our new approach are presented in Section 4. In Section 5, we use two nonlinear inverse problems to demonstrate the accuracy and efficiency of the new approach. Finally, we give some concluding remarks in Section 6.

2. Bayesian Inverse Problems

We are interested in estimating an unknown parameter $u \in \mathbb{R}^n$ from indirect observations $d \in \mathbb{R}^m$ via the following forward model

$$d = F(u) + e, \tag{2.1}$$

where $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a parameter-to-observation map that maps the unknown parameter u to the measurements d , and $e \sim \mathcal{N}(0, \Gamma_{\text{obs}})$ is the mean-zero Gaussian noise with a symmetric positive definite covariance $\Gamma_{\text{obs}} \in \mathbb{R}^{m \times m}$.

In the Bayesian setting, the prior belief about the parameter u is encoded in the prior probability distribution $\pi(u)$. Moreover, we assume that u is independent of the noise e . Then one aims at inferring the distribution of u conditioned on the data d , i.e., the posterior distribution $\pi(u|d)$. By the Bayes' rule, we have

$$\pi(u|d) \propto \exp(-\eta(u; d))\pi(u). \tag{2.2}$$

Here $\exp(-\eta(u; d))$ is called likelihood, where

$$\eta(u; d) = \frac{1}{2} \|d - F(u)\|_{\Gamma_{\text{obs}}}^2 = \frac{1}{2} (y - F(u))^T \Gamma_{\text{obs}}^{-1} (y - F(u)) \tag{2.3}$$

is the so-called potential, and \propto denotes proportionality up to a scaling constant that depends on d (but not on u).

The main task of Bayesian inverse problems (BIPs) is to characterize the posterior distribution (2.2), e.g., computing certain posterior statistic moments. Notice that if the forward model F is nonlinear, then in general the potential yields a posterior distribution which cannot be written in a closed form. Consequently, sampling methods such as the Metropolis-Hastings (MH) sampler, have been extensively studied to sampling with the posterior distribution. In this work, we will focus on the recently introduced randomize-then-optimize (RTO)-MH approach [3]. To this end, we shall present a brief introduction to RTO-MH in the next section.

3. RTO-Metropolis-Hastings Algorithm

In this section, we shall give a brief introduction to RTO. For more detailed information, one can refer to [2].

Assume that the prior of the parameter u is Gaussian, i.e., $u \sim \mathcal{N}(u_{\text{pr}}, \Gamma_{\text{pr}})$. Here u_{pr} is the prior mean, and Γ_{pr} is the prior covariance matrix. Combining with Eq. (2.3), the posterior can be thus written as

$$\pi(u|d) \propto \exp\left(-\frac{1}{2} (\|d - F(u)\|_{\Gamma_{\text{obs}}}^2 + \|u - u_{\text{pr}}\|_{\Gamma_{\text{pr}}}^2)\right). \tag{3.1}$$

Using matrix factorizations of the covariances of prior and observation noise

$$S_{\text{pr}} S_{\text{pr}}^T := \Gamma_{\text{pr}}, \quad S_{\text{obs}} S_{\text{obs}}^T := \Gamma_{\text{obs}},$$

we define a new variable $v := S_{\text{pr}}^{-1}(u - u_{\text{pr}})$ which satisfies

$$0 = f(v) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, I_m), \quad v \sim \mathcal{N}(0, I_n). \tag{3.2}$$

Here $f(v) = S_{\text{obs}}^{-1}[F(S_{\text{pr}}v + u_{\text{pr}}) - d]$, I_n and I_m are identity matrices of size n and m , respectively. Then, the resulting posterior density is given by

$$\pi(v|d) = \pi_{\text{tar}}(v) \propto \exp\left(-\frac{1}{2}\|H(v)\|^2\right), \quad (3.3)$$

where $H : \mathbb{R}^n \rightarrow \mathbb{R}^{(n+m)}$ is defined as

$$H(v) = \begin{bmatrix} v \\ f(v) \end{bmatrix}. \quad (3.4)$$

Notice that if we have a sample v from the target density $\pi_{\text{tar}}(v)$, we can easily obtain the corresponding posterior sample of u by applying the transformation $u = S_{\text{pr}}v + u_{\text{pr}}$. Now we outline how to use the RTO-MH to sample from a posterior of the form (3.3):

- We first choose a linearization point v_{ref} from the following optimization problem

$$v_{\text{ref}} = \arg \min_v \frac{1}{2}\|H(v)\|^2. \quad (3.5)$$

Then we can compute a matrix $Q \in \mathbb{R}^{(n+m) \times n}$ with orthonormal columns from a thin QR factorization of $\nabla H(v_{\text{ref}})$.

- Next, we draw independent samples $\xi^{(i)}$ from an n -dimensional standard Gaussian, and for each sample $\xi^{(i)}$ we generate proposal points $v_{\text{prop}}^{(i)}$ by solving the following optimization problem

$$v_{\text{prop}}^{(i)} = \arg \min_v \frac{1}{2}\|\bar{Q}^T H(v) - \xi^{(i)}\|^2, \quad (3.6)$$

The above equation (3.6) is called randomize-then-optimize (RTO), and under certain conditions (such as those in [3]), the points $v_{\text{prop}}^{(i)}$ are distributed according to the following proposal density,

$$\pi_{\text{RTO}}(v) = (2\pi)^{-\frac{n}{2}} |\det(Q^T \nabla H(v))| \exp\left(-\frac{1}{2}\|Q^T H(v)\|^2\right). \quad (3.7)$$

- Finally, we use (3.6) and (3.7) as an independence proposal within the MH algorithm for sampling from $\pi_{\text{tar}}(v)$. Given a previous sample $v^{(i-1)}$ and a proposed RTO sample $v_{\text{prop}}^{(i)}$, the acceptance ratio for the MH method is given by [3]

$$\frac{\pi_{\text{tar}}(v_{\text{prop}}^{(i)})\pi_{\text{RTO}}(v^{(i-1)})}{\pi_{\text{tar}}(v^{(i-1)})\pi_{\text{RTO}}(v_{\text{prop}}^{(i)})} = \frac{w(v_{\text{prop}}^{(i)})}{w(v^{(i-1)})}, \quad (3.8)$$

where

$$w(v) = |\det(Q^T \nabla H(v))|^{-1} \exp\left(-\frac{1}{2}\|H(v)\|^2 + \frac{1}{2}\|Q^T H(v)\|^2\right). \quad (3.9)$$

When the dimension of the parameter vector v is very high, solving the optimization problem (3.6) and computing the RTO probability density (3.7) can be computationally costly. In order to overcome this challenge, Bardsley et.al. [2] introduce a new subspace acceleration strategy to make the computational complexity of RTO scale linearly with the parametric dimension. The main idea is to use the singular value decomposition (SVD) of the linearized forward model $\nabla f(v_{\text{ref}})$ instead of computing the QR of the matrix $\nabla H(v_{\text{ref}})$. Similar to the original RTO, the scalable implementation of RTO also includes three steps:

Step 1. Compute the reduced SVD of the $\nabla f(v_{\text{ref}})$, which has rank r , as

$$\nabla f(v_{\text{ref}}) = \Psi \Lambda \Phi^T. \quad (3.10)$$

Step 2. Define

$$v_r = \Phi^T v, \quad \text{and} \quad v = \Phi v_r + v_{\perp}, \quad (3.11)$$

where v_{\perp} is an element in the orthogonal complement of $\text{range}(\Phi)$. For each realization of an n -dimensional standard Gaussian random vector ξ , one can compute

$$v_{\perp} = (I_n - \Phi \Phi^T) \xi \quad (3.12)$$

and solve the following r -dimensional optimization problem with ξ and v_{\perp}

$$v_r = \arg \min_z \|(\Lambda^2 + I_r)^{-\frac{1}{2}} (z + \Lambda \Psi^T f(v_{\perp} + \Phi z) - \Phi^T \xi)\|^2. \quad (3.13)$$

Notice that the original n -dimensional optimization problem in (3.6) are replaced by Eqs. (3.12) and (3.13).

Step 3. Compute the weighting function $w(v)$ in (3.9) as

$$w(v) = |\det(\tilde{Q}^T \nabla H(v))|^{-1} \exp\left(-\frac{1}{2} (\|f(v)\|^2 + \|\Phi^T v\|^2 - \|(\Lambda^2 + I_r)^{-\frac{1}{2}} (\Phi^T v + \Lambda \Psi^T f(v))\|^2)\right), \quad (3.14)$$

where the determinant takes the following simplified form

$$|\det(\tilde{Q}^T \nabla H(v))| = |\det(\Lambda^2 + I_r)^{-\frac{1}{2}}| |\det(I_r + \Lambda \Psi^T \nabla f(v) \Phi)|. \quad (3.15)$$

The resulting MCMC method, which is called scalable implementation of RTO-MH, is summarized in Algorithm 3.1.

Notice that the main bottleneck for using the RTO-MH approach is the repetitive evaluations of forward model and its gradient in the optimization procedure (3.13) and the weighting function (3.14) that involves the observation that may involve a complicated model. Thus it is desirable to construct effective approximation of these quantities to provides a good balance between accuracy and computation cost. The motivates our DNN-based surrogate model approach which will be introduced in Section 4.

Algorithm 3.1. Scalable implementation of RTO-MH [2]

1. Find v_{ref} using (3.5) .
2. Determine the Jacobian matrix of the forward model, $\nabla f(v_{\text{ref}})$.
3. Compute the SVD of $\nabla f(v_{\text{ref}})$
4. **for** $j = 1, \dots, n_{\text{samps}}$ **do** in parallel
5. Samples $\xi^{(i)}$ from a standard n -dimensional Gaussian distribution.
6. Solve for a proposal sample $v_{\text{prop}}^{(i)} = v_{\perp} + \Phi v_r$ using (3.12) and (3.13).
7. Compute $w(v_{\text{prop}}^{(i)})$ from (3.14) using the determinant from (3.15).
8. **end for**
9. Set $v^0 = v_{\text{ref}}$
10. **for** $j = 1, \dots, n_{\text{samps}}$ **do** in series
11. Sample t from a uniform distribution on $[0, 1]$.
12. Accept/reject $v_{\text{prop}}^{(i)}$ according $t < w(v_{\text{prop}}^{(i)})/w(v^{(i-1)})$
13. **end for**

4. DNN Surrogate for RTO-MH

In this section, we shall present a DNN-based surrogate modeling to accelerate the RTO-MH approach.

4.1. Feedforward DNN-based surrogate modeling

The basic idea of deep neural networks (DNN) for surrogate modeling is that one can approximate an input-output map $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ through a hierarchical abstract layers of latent variables [12]. A typical example is the feedforward neural network, which is also called multi-layer perception (MLP). It consists of a collection of layers that include an input layer, an output layer, and a number of hidden layers. Specifically, in the k th hidden layer, d_k number of neurons are included. Each hidden layer of the network receives an output $z^{(k-1)} \in \mathbb{R}^{d_{k-1}}$ from the previous layer where an affine transformation of the form

$$\mathcal{F}_k(z^{(k-1)}) = \mathbf{W}^{(k)}z^{(k-1)} + \mathbf{b}^{(k)}, \tag{4.1}$$

is performed. Here $\mathbf{W}^{(k)} \in \mathbb{R}^{d_k \times d_{k-1}}$, $\mathbf{b}^{(k)} \in \mathbb{R}^{d_k}$ are the weights and biases of the k th layer. The nonlinear activation function σ is applied to each component of the transformed vector before sending it as an input to the next layer. The activation function is an identity function after an output layer. Thus, the final neural network representation is given by the composition

$$\mathcal{NN}(v) = (\mathcal{F}_L \circ \sigma \circ \mathcal{F}_{L-1} \circ \dots \circ \sigma \circ \mathcal{F}_1)(v), \tag{4.2}$$

where the operator \circ is the composition operator, $v = z^{(0)}$ is the input. A typical neural network architecture can be founded in Fig. 4.1.

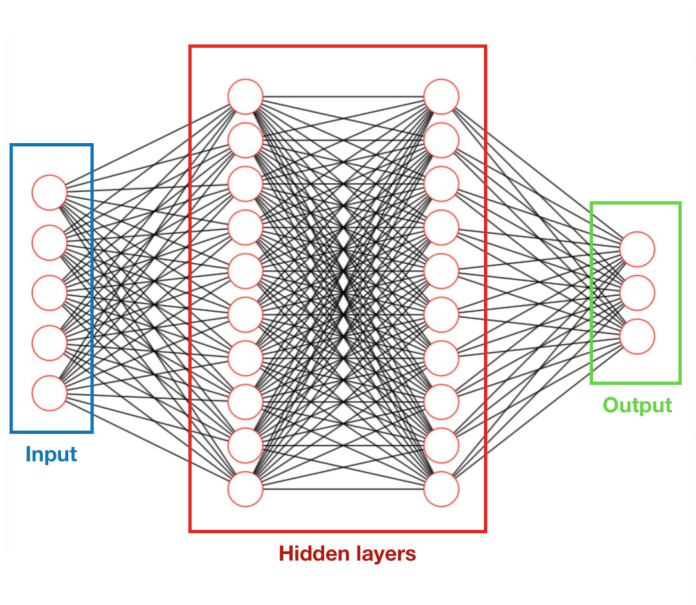


Fig. 4.1. The structure of a two-hidden-layer neural network.

Some popular choices for the activation function include sigmoid, hyperbolic tangent, rectified linear unit (ReLU), to name a few [12, 24]. In the current work, we shall use `Swish` as the

activation function [24, 29]:

$$\sigma(z) = \frac{z}{1 + \exp(-z)}.$$

Once the network architecture is defined, one can resort to optimization tools to find the unknown parameters $\theta = \{\mathbf{W}^{(k)}, \mathbf{b}^{(k)}\}$ based on the training data. Precisely, let $\mathcal{D} := \{(v_i, y_i)\}_{i=1}^N$ be a set of training data, we can define the following minimization problem:

$$\arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \|y_i - \mathcal{N}\mathcal{N}(v_i; \theta)\|^2, \quad (4.3)$$

where $\mathcal{J}(\theta; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N \|y_i - \mathcal{N}\mathcal{N}(v_i; \theta)\|^2$ is the so called loss function. Solving this problem is generally achieved by the stochastic gradient descent (SGD) algorithm [6]. SGD simply minimizes the function by taking a negative step along an estimate of the gradient $\nabla_{\theta} \mathcal{J}(\theta; \mathcal{D})$ at iteration k . The gradients are usually computed through backpropagation. At each iteration, SGD updates the solution by

$$\theta_{k+1} = \theta_k - \lambda \nabla_{\theta} \mathcal{J}(\theta; \mathcal{D}),$$

where λ is the learning rate. Recent algorithms that offer adaptive learning rates are available, such as Ada-Grad [36], RMSProp [28] and Adam [16], ect. The present work adopts Adam optimization algorithm, and we shall construct a DNN as a surrogate model for the true forward model in the BIPs.

4.2. Choosing effective training points

Notice that the accuracy of the neural network $\mathcal{N}\mathcal{N}(v; \theta)$ approximation are clearly influenced by the choice of the training points. A naive way to choose the training data is to generate enough data over the whole prior distribution, however, this may loss the gain computational efficiency [33, 35]. Notice that our concern in BIPs is only the posterior distribution which may be located in a small region. Consequently, we only need to make sure that $\mathcal{N}\mathcal{N}(v; \theta)$ is accurate enough in the posterior density region where no need to ensure its accuracy elsewhere. Nevertheless, the high-probability density region of the posterior hard to be identified until data are available. In the following, we propose a goal-oriented technique based on the observation data and the gradient information to choose the local training points.

Consider the first-order Taylor approximation of the forward model $f(v)$ at v_{ref} , we have

$$f(v) \approx f(v_{\text{ref}}) + \nabla f(v_{\text{ref}})(v - v_{\text{ref}}) := Av - b, \quad (4.4)$$

where $A := \nabla f(v_{\text{ref}})$ and $b = Av_{\text{ref}} - f(v_{\text{ref}})$. Using this linearization approximation, we can define the *approximation* posterior of $\pi_{\text{tar}}(v)$ as

$$\pi_{\text{tar}}(v) \approx \tilde{\pi}_{\text{tar}}(v) \propto \exp\left(-\frac{1}{2} \tilde{H}(v)\right), \quad (4.5)$$

where $\tilde{H} : \mathbb{R}^n \rightarrow \mathbb{R}^{(n+m)}$ is defined as

$$\tilde{H}(v) = \begin{bmatrix} v \\ Av - b \end{bmatrix}. \quad (4.6)$$

It is easy to verify that $\tilde{\pi}_{\text{tar}} \sim \mathcal{N}(\tilde{\mu}_{\text{pos}}, \tilde{\Gamma}_{\text{pos}})$ with

$$\tilde{\Gamma}_{\text{pos}} = (A^T A + I)^{-1}, \quad \text{and} \quad \tilde{\mu}_{\text{pos}} = \tilde{\Gamma}_{\text{pos}} A^T b. \quad (4.7)$$

This means that the first-order Taylor approximation provides us with a *local* Gaussian measure approximating π_{var} . This Gaussian measure allows for direct sampling if the reference point v_{ref} and the gradient of the forward model are given. Notice that the covariance is the inverse matrix of the $A^T A + I$, however, with a simple algebraic manipulation by the reduced SVD $A = \nabla f(v_{\text{ref}}) = \Psi \Lambda \Phi^T$, we can write it as

$$\tilde{\Gamma}_{\text{pos}} = \Phi(\Lambda^2 + I_r)^{-1}\Phi^T + (I_n - \Phi\Phi^T). \quad (4.8)$$

The above discussion motivates us to choose the training points from the local Gaussian approximation of the posterior with covariance $\tilde{\Gamma}_{\text{pos}}$ defined in (4.8). In Section 5, we will perform the comparison between this strategy and the prior-based strategy (choose the data in the whole prior domain) by numerical examples.

4.3. Summary of the DNN-based RTO-MH

As mentioned in the previous sections, we can alleviate the complexity issue with a DNN surrogate for the forward model $f(v)$. It is clear that after obtaining the parameters θ , we have an explicit functional $\mathcal{NN}(v; \theta)$ and can compute its gradient $\nabla_v \mathcal{NN}(v; \theta)$ easily via the back propagation [5]. These approximations can be then substituted into the computation procedure of the RTO-MH framework, and obtain the DNN-based RTO-MH algorithm. When the original forward model is computationally costly, simulating the surrogate $\mathcal{NN}(v; \theta)$ provides a more efficient mechanism for the RTO-MH sampler. The detail of the scheme is summarized in Algorithm 4.1. Our proposed method provides a natural framework to incorporate DNN surrogate in RTO-MH. Moreover, it can be easily extended to other optimization-based sampling, e.g., the random-map implicit sampling [23].

Algorithm 4.1. The offline and online stages for the NN-RTO method

1. **Offline stage:**
2. Choose N training points $\{v_i\}_{i=1}^N$ randomly from the approximate posterior $\tilde{\pi}_{\text{var}}(v)$
3. Compute the corresponding full-order snapshots $\{f(v_i)\}_{i=1}^N$;
4. Prepare the training set $\mathcal{D} = \{v_i, f(v_i)\}$;
5. Train the DNN model $\mathcal{NN}(v; \theta)$ by using the training set \mathcal{D} .
6. **Online stage:**
7. **for** $j = 1, \dots, n_{\text{samps}}$ **do** in parallel
8. Samples $\xi^{(i)}$ from a standard n -dimensional Gaussian distribution.
9. Compute a proposal sample $v_{\text{prop}}^{(i)}$, and $w(v_{\text{prop}}^{(i)})$ using RTO algorithm with the trained neural network $\mathcal{NN}(v; \theta)$.
10. **end for**
11. Set $v^0 = v_{\text{ref}}$
12. **for** $j = 1, \dots, n_{\text{samps}}$ **do** in series
13. Sample t from a uniform distribution on $[0, 1]$.
14. Accept/reject $v_{\text{prop}}^{(i)}$ according $t < w(v_{\text{prop}}^{(i)})/w(v^{(i-1)})$
15. **end for**

5. Numerical Examples

In this section, we present a benchmark elliptic PDE inverse problem to illustrate the accuracy and efficiency of the DNN-RTO-MH approach. To better present the results, we shall perform the following three-types of approaches:

- The conventional “RTO” (or the direct RTO) which is based on the true forward model evaluations.
- The “NN-RTO-pr” that use a DNN surrogate with training data that are generated with respect to the prior distribution.
- The “NN-RTO” that use a DNN surrogate with training data that are generated with respect to an approximation posterior distribution in Section 4.2, i.e., the suggested algorithm in this work.

In our all numerical tests, the computations were performed using MATLAB 2018a on an Intel-i7 desktop computer. For solving the optimization problem (3.13) we use the built-in nonlinear least squares solver lsqnonlin of MATLAB, which implements the trust region reflective Newton method. The Adam optimizer is used to train the DNN as mentioned before. The learning rate is set to be $\lambda = 5 \times 10^{-4}$, and the hyper-parameter values of Adam are chosen based on default recommendations as suggested in [16]. In order to compare our proposed method to standard RTO-MH in terms of sampling efficiency, we consider the effective sample size (ESS) adjusted by CPU time. Given n_{samps} posterior samples, the ESS for each parameters is defined as

$$ESS = \frac{n_{\text{samps}}}{1 + 2 \sum_{k=1}^K \rho(k)} \quad (5.1)$$

where $\sum_{k=1}^K \rho(k)$ is the sum of K monotone sample autocorrelations [10]. We use the minimum ESS over all parameters normalized by the CPU time, s (in seconds), as the overall measure of efficiency: $\min(ESS)/s$. In this work, we will use $n_{\text{samps}} = 5000$ and $K = n_{\text{samps}} - 1$ to obtain the results.

5.1. Problem setup

We consider the problem of inferring subsurface permeability from a finite number of noisy pressure head measurements [8, 35]. The forward model is given by the solution of an elliptic PDE in two spatial dimensions

$$-\nabla \cdot (\kappa(x) \nabla p(x)) = f(x), \quad (5.2)$$

where $x = (x_1, x_2) \in [0, 1]^2$ is the spatial coordinate. The boundary conditions are

$$\begin{aligned} p(x)|_{x_1=0} &= 1, & p(x)|_{x_1=1} &= 0, \\ \frac{\partial p(x)}{\partial x_2}|_{x_2=0} &= x_1, & \frac{\partial p(x)}{\partial x_2}|_{x_2=1} &= 1 - x_1. \end{aligned}$$

The data d is given by a finite set of p , perturbed by noise, and the problem is to recover the permeability $\kappa(x)$ from these measurements. In what follows, we choose the source $f(x) = 100 \sin(\pi x_1) \sin(\pi x_2)$, and solve Eq. (5.2) using a standard Galerkin finite element method with bilinear basis function on a uniform 40-by-40 grid.

5.2. Example 1: a nine-dimensional inverse problem

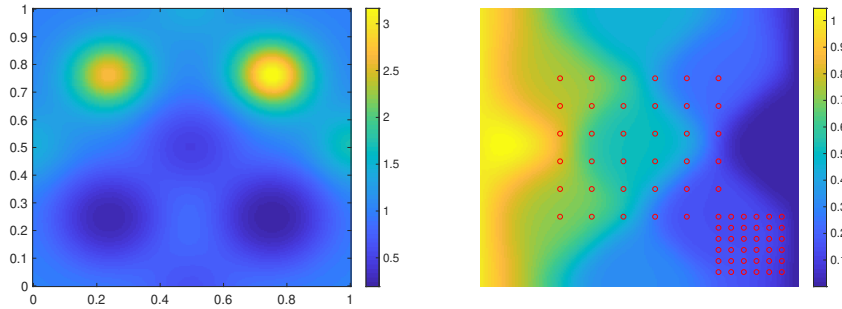


Fig. 5.1. Example 1: Setup of the test case for Example 1. Left: the true permeability used for generating the synthetic data sets. Right: the model outputs of the true permeability. The red circles indicate the measurement sensors.

In the first example, the permeability field $\kappa(x)$ is defined by

$$\kappa(x) = \sum_{i=1}^9 \kappa_i \exp\left(-\frac{\|x - x_{0,i}\|^2}{2 \times 0.1^2}\right),$$

where $\{x_{0,i}\}_{i=1}^9$ are the centers of the radial basis function, and the weights $\{\kappa_i\}_{i=1}^9$ are parameters in the Bayesian inverse problem.

This example is a typical benchmark problem considered in Refs. [8, 33]. We first choose a realization of κ_i from a uniform distribution as the true solution. The true permeability field used to generate the test data and the corresponding pressure head are shown in Fig.5.1. The prior distributions on each of the weights $\kappa_i, i = 1, \dots, 9$ are independent and log-normal; that is, $\log(\kappa_i) := v^i \sim N(0, 1)$. Partial observations of the pressure field are collected 71 measurement sensors as shown by the red circles in Fig. 5.1. This yields observed data $d \in \mathbb{R}^{71}$ as

$$d_j = p(x_j) + \max_j\{|p(x_j)|\}\delta\xi_j,$$

where δ dictates the relative noise level and ξ_j is a Gaussian random variable with zero mean and unit standard deviation. In the following, we set $\delta = 0.05$.

5.2.1. Computational efficiency

We first investigate the sampling accuracy of the DNN-RTO algorithm. To this end, we shall construct the DNN surrogate using $N = \{50, 100\}$ training points with 3 hidden layers and 40 neurons per layer. Notice that the training points are chosen with the algorithm in Section 4.2. In Fig. 5.2, we present the marginal distributions of each component of the parameters, and the contours of the marginal distributions of each pair of components. The black lines represent the results generated by the direct RTO approach based on the true forward model evaluations (the reference solution), the red and blue lines represent results of the DNN-RTO with $N = 50$ and 100, respectively. It is clearly seen in Fig. 5.2 that the DNN-RTO algorithm results in a good approximation to the reference solution. The posterior mean and posterior standard deviation obtained by the DNN-RTO approach and the direct RTO approach are presented in Fig. 5.3. Good agreements between the two algorithms are observed.

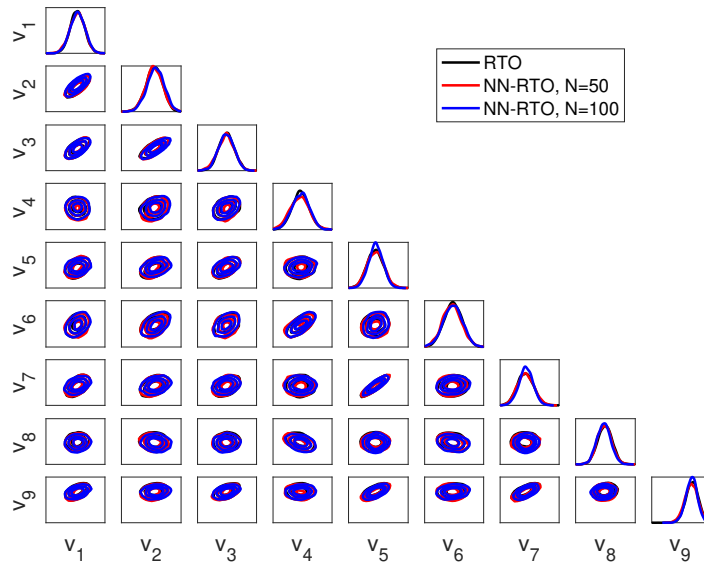


Fig. 5.2. One- and two-dimensional posterior marginals of the two parameters for Example 1 using NN-RTO approach.

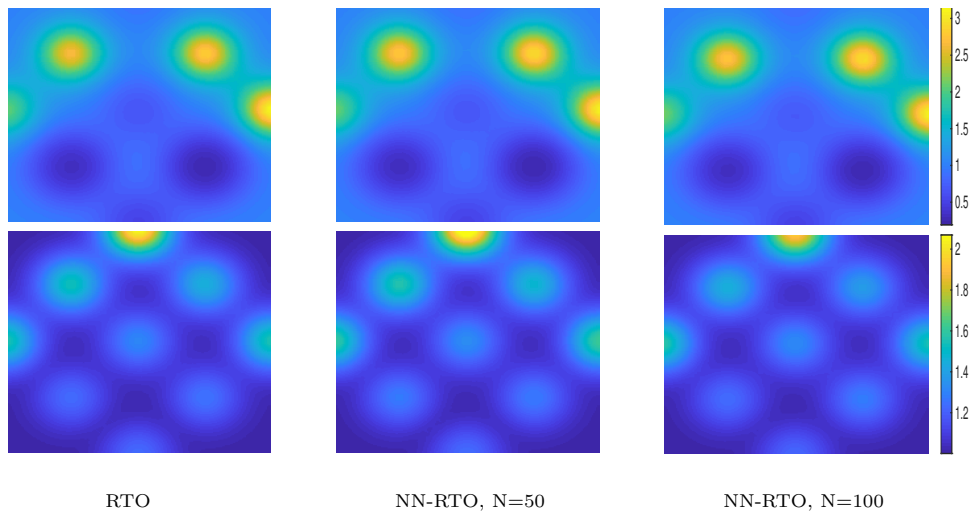


Fig. 5.3. Posterior mean (top) and posterior standard deviation (bottom) of $p(x)$ arising from direct RTO, NN-RTO approach ($N = 50, 100$), respectively.

Next, we compare the accuracy of our proposed algorithm to that of DNN-RTO-pr (with training points generated according to the prior distribution). Again, we use $N = \{50, 100\}$ training points that are generated by the prior information (rather than the algorithm in Section 4.2) to train the DNN. The corresponding results are reported in Figs. 5.4 and 5.5. It is shown in the figures that the results using the prior-based DNN approach admits a larger error. By comparing Figs. 5.2 and 5.3, we can conclude that the approximation results using DNN-RTO are much more accurate than that of the prior-based "DNN-RTO-pr" approach.

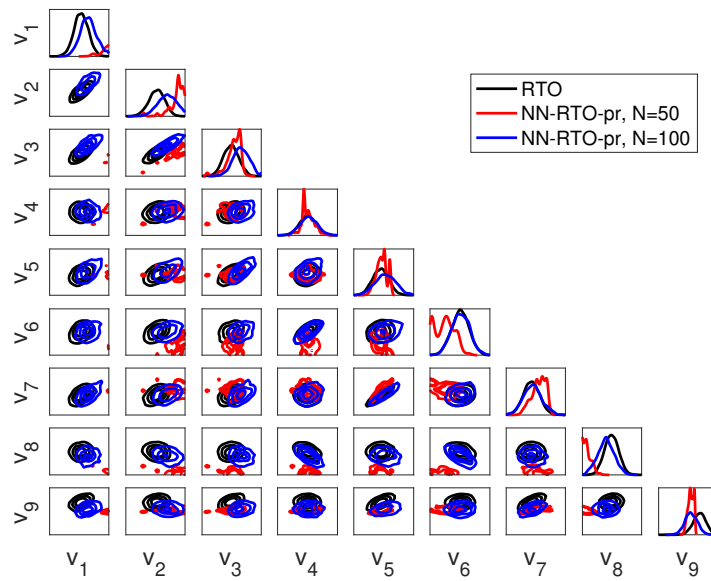


Fig. 5.4. One- and two-dimensional posterior marginals of the two parameters for Example 1 using NN-RTO-pr approach.

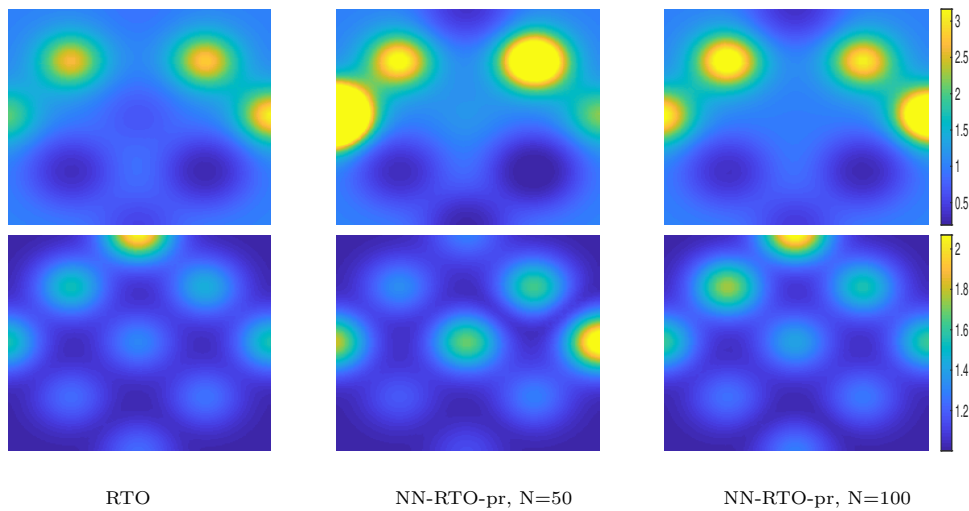


Fig. 5.5. Posterior mean (top) and posterior standard deviation (bottom) of $p(x)$ arising from direct RTO, NN-RTO-pr approach ($N = 50, 100$), respectively.

In Table 5.1, we summarize the acceptance probability, the CPU time, the ESS (min., med., max.), the time-normalized minESS and the speedup factor, for comparing the direct RTO and the DNN-RTO approach. It can be seen that RTO makes 5000 draws in 9806 seconds and obtains an acceptance probability around 0.93. While for DNN-RTO with $N = 50$, only 156 seconds are required to get an acceptance probability around 0.82. DNN-RTO yields 15.4 effective samples per second while RTO is only 0.41 effective draws per second. It is noticed that even when with $N = 50$ training points, the DNN-RTO approach can speed up the number

Table 5.1: Comparison of the computational efficiency in Example 1. AP is the acceptance probability, s is the CPU times (second), ESS (min., med., max.), minESS/ s is the time-normalized ESS and spdup is the speed up of sampling efficiency measured by minESS/ s with RTO as the baseline. MCMC chain length is 5000 steps.

Method	AP	s	ESS	minESS/ s	spdup
RTO	0.93	9806	(4030, 4342, 4460)	0.41	1
DNN-RTO, $N = 50$	0.82	156	(2409, 2719, 3428)	15.4	37.6
DNN-RTO, $N = 100$	0.80	170	(2612, 2895, 3606)	15.4	37.6
DNN-RTO-pr, $N = 50$	0.08	85	(3, 5, 77)	0.04	0.1
DNN-RTO-pr, $N = 100$	0.54	169	(529, 958, 1245)	3.1	7.6

of effectively independent samples generated per second by a factor of 37. One can also learn that NN-RTO is much more efficient than DNN-RTO-pr.

5.2.2. Sensitivity to the neural network architecture

We now investigate the sensitivity of our algorithm with respect to the architecture of the neural networks. We consider the error in estimating mean REM and covariance REC of parameters as

$$REM = \frac{\|\bar{\kappa} - \kappa^\dagger\|_\infty}{\|\kappa^\dagger\|_\infty}, \quad REC = \frac{\|cov(\kappa) - cov(\kappa^\dagger)\|_F}{\|cov(\kappa^\dagger)\|_F},$$

where κ^\dagger and $cov(\kappa^\dagger)$ are the “true” posterior mean and covariance arising from direct RTO, and $\|\cdot\|_F$ denotes the Frobenius norm.

To verify the sensitivity of the proposed method with respect to the structures of the \mathcal{NN} , we test several constant values choosing with different depth $L \in \{1, 2, 3, 4\}$ and width $d_k \in \{20, 40, 60, 80\}$. With these settings, we run Algorithm 4.1 using the $N = 100$ training points. The corresponding numerical results are presented in Table 5.2, in which the corresponding CPU times are also presented. As shown in this table, the computational results for DNN-RTO with different depth L and width d_k are almost the same. Table 5.3 shows the estimate errors REM , REC and the CPU times s with respect to the number of training points obtained with various values of d_k and $L = 3$ hidden layers. Overall, we observe that the numerical results remain robust for all neural network architectures considered.

Table 5.2: Example 1: The estimate errors REM , REC and the CPU times s obtained using NN-RTO approach with various values of L , d_k and $N = 100$ training points.

$L \backslash d_k$	20	40	60	80
1	(0.019,0.102,94)	(0.017,0.046,121)	(0.022,0.052,128)	(0.017,0.033,137)
2	(0.026,0.510,150)	(0.022,0.095,166)	(0.010,0.057,195)	(0.019,0.038,202)
3	(0.029,0.949,139)	(0.029,0.068,210)	(0.018,0.062,246)	(0.016,0.077,270)
4	(0.402,0.918,175)	(0.029,0.074,241)	(0.055,0.100,286)	(0.049,0.066,293)

Table 5.3: The estimate errors REM, REC and the CPU times s obtained using NN-RTO approach with various values of N, d_k and $L = 3$ hidden layers.

$N \backslash d_k$	20	40	60	80
50	(0.069,1.150,155)	(0.058,0.097,191)	(0.019,0.072,228)	(0.022,0.033,261)
100	(0.029,0.949,139)	(0.029,0.068,210)	(0.018,0.062,246)	(0.016,0.077,270)
150	(0.018,0.136,193)	(0.016,0.034,209)	(0.015,0.054,227)	(0.019,0.038,255)
200	(0.018,0.134,117)	(0.015,0.032,210)	(0.019,0.063,225)	(0.018,0.050,256)

Table 5.4: Comparison of the computational efficiency in Example 2. AP is the acceptance probability, s is the CPU times (second), ESS(min., med., max.), minESS/ s is the time-normalized ESS and spdup is the speed up of sampling efficiency measured by minESS/ s with RTO as the baseline. MCMC chain is 5000 steps.

Method	AP	s	ESS	minESS/ s	spdup
RTO	0.54	2271	(1005, 1249, 1668)	0.44	1
DNN-RTO, $N = 1000$	0.35	96	(1050, 1983, 2259)	10.9	25
DNN-RTO, $N = 1500$	0.35	88	(1036, 1264, 1470)	11.8	27

5.3. Example 2: a high dimensional inverse problem

In the second example, we consider a log-diffusivity field $\log \kappa(x)$ that is endowed with a Gaussian process prior, with zero mean and an isotropic kernel:

$$C(x_1, x_2) = \sigma^2 \exp\left(-\frac{\|x_1 - x_2\|^2}{2l^2}\right).$$

Here we set variance $\sigma^2 = 1$ and $l = 0.1$. This prior allows the field to be easily parameterized with a Karhunen-Loeve expansion:

$$p(x; v) \approx \sum_{i=1}^n v^i \sqrt{\lambda_i} \phi_i(x), \tag{5.3}$$

where λ_i and $\phi_i(x)$ are the eigenvalues and eigenfunctions, respectively, of the integral operator on $[0, 1]^2$ defined by the kernel C , and the parameter $v = (v^1, \dots, v^n)$ are endowed with independent standard normal priors, $v^i \sim N(0, 1)$. These parameters then become the targets of inference. In particular, we truncate the Karhunen-Loeve expansion with $n = 120$ modes. The true solution $\kappa(x)$ used to generate the test data are shown in Fig. 5.6. The measurement sensors of p are evenly distributed over $[0.1, 0.9]^2$ with grid spacing 0.1, i.e., $d \in \mathbb{R}^{81}$. The observational errors are taken to be additive and Gaussian:

$$d_j = p(x_j; v) + \xi_j, \quad j = 1, \dots, 81,$$

with $\xi_j \sim N(0, 0.05^2)$. In this example, four hidden layers and 80 neurons per layer are used in \mathcal{NN} .

We ran DNN-RTO with $N = \{1000, 1500\}$ training points and generated an MCMC chain of length 5000. As shown in Fig. 5.7, the posterior mean and posterior standard deviation estimated by DNN-RTO are closer to the reference solution (that is obtained by direct RTO approach). We compare the sampling efficiency of different algorithms, and the result is summarized in Table 5.4. Again, we observe a similar result in the raw ESS when comparing

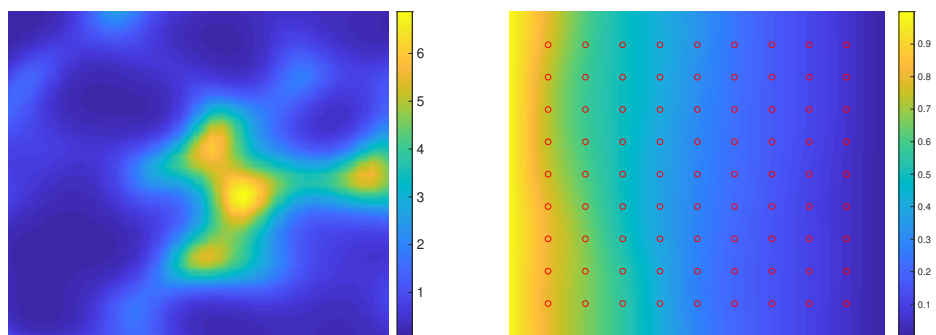


Fig. 5.6. The true permeability $\kappa(x)$ (left) and the model outputs (right) for elliptic PDE inverse problems in example 2.

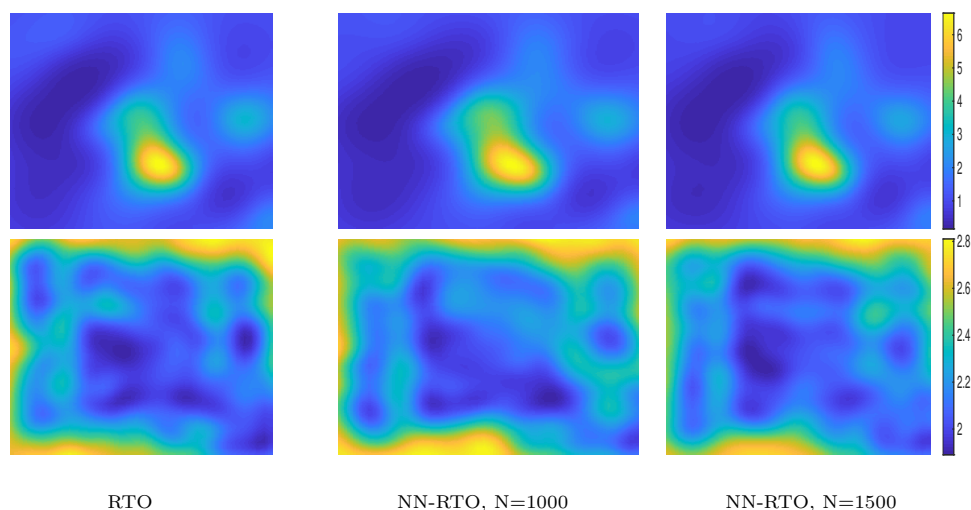


Fig. 5.7. Posterior mean (top) and posterior standard deviation (bottom) arising from direct RTO, NN-RTO approach ($N = 1000$) and prior-based NN-RTO ($N = 1500$), respectively.

DNN-RTO algorithms with the direct RTO, but an increase in efficiency due to the computational time cut by DNN surrogate. The computation used 96 seconds to produce a minESS of about 10^3 for DNN-RTO with $N = 1000$. Notice that more than an order of magnitude of improvement is observed for DNN-RTO compared to RTO. All the above discussion confirm that DNN-RTO is advantageous in sampling efficiency.

6. Summary

In this paper, we present a new strategy, namely the DNN-RTO algorithm, to accelerate the original RTO-MH algorithm. One of the key components of our DNN-RTO algorithm is a goal-oriented strategy for choosing the training points from a local Gaussian measurement. Our DNN-surrogate requires very few training points to achieve the same level of accuracy compared with a prior-based DNN-surrogate. To demonstrate the accuracy and efficiency of the proposed algorithm, a benchmark example to infer the permeability field for elliptic PDEs with synthetic

data is tested. The numerical results show that the DNN-RTO is able to accelerate RTO sampling by up to several orders of magnitude. We believe the approach in this work will be promising in dealing with high dimensional BIPs and/or BIPs with limited regularity.

Acknowledgments. LY's work was supported by the NSF of China (No. 11771081), the science challenge project, China (No. TZ2018001) and Zhishan Young Scholar Program of SEU, China. TZ's work was supported by the National Key R&D Program of China (No. 2020YFA0712000), the NSF of China (under grant numbers 11822111, 11688101 and 11731006), the science challenge project (No. TZ2018001), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA25000404) and youth innovation promotion association (CAS), China.

References

- [1] S.R. Arridge, J.P. Kaipio, V. Kolehmainen, M. Schweiger, E. Somersalo, T. Tarvainen, and M. Vauhkonen, Approximation errors and model reduction with an application in optical diffusion tomography, *Inverse Problems*, **22**:1 (2006), 175–195.
- [2] J.M. Bardsley, T. Cui, Y. Marzouk, and Z. Wang, Scalable optimization-based sampling on function space, *arXiv preprint arXiv:1903.00870*, 2019.
- [3] J.M. Bardsley, A. Solonen, H. Haario, and M. Laine, Randomize-then-optimize: A method for sampling from posterior distributions in nonlinear inverse problems, *SIAM Journal on Scientific Computing*, **36**:4 (2014), A1895–A1910.
- [4] 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), 327–351.
- [5] Christopher M Bishop, *Pattern recognition and machine learning*, springer, 2006.
- [6] L. Bottou, Large-scale machine learning with stochastic gradient descent, In *Proceedings of COMPSTAT'2010*, Springer (2010), 177–186.
- [7] S. Brooks, A. Gelman, G.L. Jones, and X.L. Meng, editors, *Handbook of Markov chain Monte Carlo*, Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, Boca Raton, FL, 2011.
- [8] T. Cui, Y.M. Marzouk, and K. Willcox, Data-driven model reduction for the Bayesian solution of inverse problems, *International Journal for Numerical Methods in Engineering*, **102**:5 (2015), 966–990.
- [9] D. Galbally, K. Fidkowski, K. Willcox, and O. Ghattas, Non-linear model reduction for uncertainty quantification in large-scale inverse problems, *International Journal for Numerical Methods in Engineering*, **81**:12 (2010), 1581–1608.
- [10] C.J. Geyer, Practical markov chain monte carlo, *Statistical science*, (1992), 473–483.
- [11] M. Girolami and B. Calderhead, Riemann manifold langevin and hamiltonian monte carlo methods, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **73**:2 (2011), 123–214.
- [12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.
- [13] M.D. Hoffman and A. Gelman, The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo, *Journal of Machine Learning Research*, **15**:1 (2014), 1593–1623.
- [14] J.P. Kaipio and E. Somersalo, *Statistical and Computational Inverse Problems*, volume 160, Springer, 2005.
- [15] M.C. Kennedy and A. O'Hagan, Bayesian calibration of computer models, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**:3 (2001), 425–464.
- [16] D. Kingma and J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*, 2014.

- [17] 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), 81–101.
- [18] C. Lieberman, K. Willcox, and O. Ghattas, Parameter and state model reduction for large-scale statistical inverse problems, *SIAM Journal on Scientific Computing*, **32**:5 (2010), 2523–2542.
- [19] J. Martin, L.C. Wilcox, C. Burstedde, and O. Ghattas, A stochastic newton MCMC method for large-scale statistical inverse problems with application to seismic inversion, *SIAM Journal on Scientific Computing*, **34**:3 (2012), A1460–A1487.
- [20] Y.M. Marzouk and H.N. Najm, Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems, *Journal of Computational Physics*, **228**:6 (2009), 1862–1902.
- [21] Y.M. Marzouk, H.N. Najm, and L.A. Rahn, Stochastic spectral methods for efficient Bayesian solution of inverse problems, *Journal of Computational Physics*, **224**:2 (2007), 560–586.
- [22] Y.M. Marzouk and D. Xiu, A stochastic collocation approach to Bayesian inference in inverse problems, *Communications in Computational Physics*, **6** (2009), 826–847.
- [23] M. Morzfeld, X. Tu, E. Atkins, and A.J. Chorin, A random map implementation of implicit filters, *Journal of Computational Physics*, **231**:4 (2012), 2049–2066.
- [24] P. Ramachandran, B. Zoph, and Q. Le, Searching for activation functions, *arXiv preprint arXiv:1710.05941*, 2017.
- [25] Carl Edward Rasmussen, J.M. Bernardo, M.J. Bayarri, J.O. Berger, A.P. Dawid, D. Heckerman, A.F.M. Smith, and M. West, Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals, In *Bayesian Statistics 7*, (2003), 651–659, 2003.
- [26] A.M. Stuart, Inverse problems: a Bayesian perspective, *Acta Numerica*, **19**:1 (2010), 451–559.
- [27] A.M. Stuart and A. Teckentrup, Posterior consistency for Gaussian process approximations of Bayesian posterior distributions, *Mathematics of Computation*, **87**:310 (2018), 721–753.
- [28] T. Tieleman and G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, *COURSERA: Neural networks for machine learning*, **4**:2 (2012), 26–31.
- [29] R.K. Tripathy and I. Bilionis, Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification, *Journal of Computational Physics*, **375** (2018), 565–588.
- [30] Z. Wang, J.M. Bardsley, A. Solonen, T. Cui, and Y.M. Marzouk, Bayesian inverse problems with l_1 priors: A randomize-then-optimize approach, *SIAM Journal on Scientific Computing*, **39**:5 (2017), S140–S166.
- [31] L. Yan and L. Guo, Stochastic collocation algorithms using l_1 -minimization for Bayesian solution of inverse problems. *SIAM Journal on Scientific Computing*, **37**:3 (2015), A1410–A1435.
- [32] L. Yan and Y.X. Zhang, Convergence analysis of surrogate-based methods for Bayesian inverse problems, *Inverse Problems*, **33**:12 (2017), 125001.
- [33] L. Yan and T. Zhou, Adaptive multi-fidelity polynomial chaos approach to Bayesian inference in inverse problems, *Journal of Computational Physics*, **381** (2019), 110–128.
- [34] L. Yan and T. Zhou, An adaptive multifidelity pc-based ensemble kalman inversion for inverse problems, *International Journal for Uncertainty Quantification*, **9**:3 (2019), 205–220.
- [35] L. Yan and T. Zhou, An adaptive surrogate modeling based on deep neural networks for large-scale bayesian inverse problems, *Communications in Computational Physics*, **28**:5 (2020), 2180–2205.
- [36] M. Zeiler, Adadelta: an adaptive learning rate method, *arXiv preprint arXiv:1212.5701*, 2012.