Journal of Computational Physics 394 (2019) 56–81

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data

Yinhao Zhu [a], Nicholas Zabaras [a,*], Phaedon-Stelios Koutsourelakis [b], Paris Perdikaris [c]

[a] Center for Informatics and Computational Science, 311I Cushing Hall, University of Notre Dame, Notre Dame, IN 46556, USA
[b] Continuum Mechanics Group, Technical University of Munich, Boltzmannstraße 15, 85748 Garching, Germany
[c] Department of Mechanical Engineering and Applied Mechanics, 220 South 33rd Street, 229 Towne Building, University of Pennsylvania, Philadelphia, PA 19104-6315, USA

ARTICLE INFO

Article history:
Received 17 January 2019
Received in revised form 14 May 2019
Accepted 18 May 2019
Available online 22 May 2019

Keywords:
Physics-constrained
Normalizing flow
Conditional generative model
Reverse KL divergence
Surrogate modeling
Uncertainty quantification

ABSTRACT

Surrogate modeling and uncertainty quantification tasks for PDE systems are most often considered as supervised learning problems where input and output data pairs are used for training. The construction of such emulators is by definition a small data problem which poses challenges to deep learning approaches that have been developed to operate in the big data regime. Even in cases where such models have been shown to have good predictive capability in high dimensions, they fail to address constraints in the data implied by the PDE model. This paper provides a methodology that incorporates the governing equations of the physical model in the loss/likelihood functions. The resulting physics-constrained, deep learning models are trained without any labeled data (e.g. employing only input data) and provide comparable predictive responses with data-driven models while obeying the constraints of the problem at hand. This work employs a convolutional encoder-decoder neural network approach as well as a conditional flow-based generative model for the solution of PDEs, surrogate model construction, and uncertainty quantification tasks. The methodology is posed as a minimization problem of the reverse Kullback-Leibler (KL) divergence between the model predictive density and the reference conditional density, where the later is defined as the Boltzmann-Gibbs distribution at a given inverse temperature with the underlying potential relating to the PDE system of interest. The generalization capability of these models to out-of-distribution input is considered. Quantification and interpretation of the predictive uncertainty is provided for a number of problems.

* Corresponding author.
E-mail addresses: yzhu10@nd.edu (Y. Zhu), nzabaras@gmail.com (N. Zabaras), p.s.koutsourelakis@tum.de (P.-S. Koutsourelakis), pgp@seas.upenn.edu (P. Perdikaris).
URLs: https://www.cics.nd.edu/ (Y. Zhu), http://www.contmech.mw.tum.de/index.php?id=5 (P.-S. Koutsourelakis), https://www.seas.upenn.edu/directory/profile.php?ID=237 (P. Perdikaris).

https://doi.org/10.1016/j.jcp.2019.05.024
0021-9991/

1. Introduction

Surrogate modeling is computationally attractive for problems that require repetitive yet expensive simulations, such as determinsitsic design, uncertainty propagation, optimization under uncertainty or inverse modeling. Data-efficiency, uncertainty quantification and generalization are the main challenges facing surrogate modeling, especially for problems with high-dimensional stochastic input, such as material properties [1], background potentials [2], etc.

Training surrogate models is commonly posed as a supervised learning problem, which requires simulation data as the target. Gaussian process (GP) models are widely used as emulators for physical systems [3] with built-in uncertainty quantification. The recent advances to scale GPs to high-dimensional input include Kronecker product decomposition that exploits the spatial structure [1,4,5], convolutional kernels [6] and other algorithmic and software developments [7]. However, GPs are still struggling to effectively model high-dimensional input-output maps. Deep neural networks (DNNs) are becoming the most popular surrogate models nowadays across engineering and scientific fields. As universal function approximators, DNNs excel at settings where both the input and output are high-dimensional. Applications in flow simulations include pressure projections in solving Navier-Stokes equations [8], fluid flow through random heterogeneous media [9–11], Reynolds-Averaged Navier-Stokes simulations [12–14] and others. Uncertainty quantification for DNNs is often studied under the re-emerging framework of Bayesian deep learning[1] [15], mostly using variational inference for approximate posterior of model parameters, e.g. variational dropout [16,17], Stein variational gradient descent [18,9], although other methods exist, e.g. ensemble methods [19]. Another perspective to high-dimensional problems is offered by latent variable models [20], where the latent variables encode the information bottleneck between the input and output.

Sufficient amount of training data is usually required for the surrogates to achieve accurate predictions even under restricted settings, e.g. fixed boundary conditions. For physically-grounded domains, baking in the prior knowledge can potentially overcome the challenges of data-efficiency and generalization, etc. The *inductive bias* can be built into the network architecture, e.g. spherical convolutional neural networks (CNNs) for the physical fields on unstructured grid [21], graph networks for object- and relation-centric representations of complex, dynamical systems [22], learning linear embeddings of nonlinear dynamics based on Koopman operator theory [23]. Another approach is to *embed physical laws* into the learning systems, such as approximating differential operators with convolutions [24], enforcing hard constraint of mass conservation by learning the stream function [25] whose curl is guaranteed to be divergence-free.

A more general way to incorporate physical knowledge is through *constraint learning* [26], i.e. learning the models by minimizing the violation of the physical constraints, symmetries, e.g. cycle consistency in domain translation [27], temporal coherence of consecutive frames in fluid simulation [28] and video translation [29]. One typical example in computational physics is *learning solutions of deterministic PDEs with neural networks* in space/time, which dates back at least to the early 1990s, e.g. [30–32]. The main idea is to train neural networks to approximate the solution by minimizing the violation of the governing PDEs (e.g. the residual of the PDEs) and also of the initial and boundary conditions. In [32], a one-hidden-layer fully-connected neural network (FC-NN) with spatial coordinates as input is trained to minimize the residual norm evaluated on a fixed grid. Most of the works in the recent literature parameterize the solution with FC-NNs, thus the solution is analytical and meshfree [33,34]. In other works, the loss function is derived from a variational form [35,36]. Stochastic gradient descent is used to train the network by randomly sampling mini-batches of inputs (spatial locations and/or time instances) [37,35] and deeper networks are used to break the curse of dimensionality [38] allowing for several high-dimensional PDEs to be solved with high accuracy and speed [39,40,37,41]. Several multiscale methods have been enhanced by learning the basis functions with DNNs [42,43]. Finally, several applications of DNNs to surrogate modeling and uncertainty quantification tasks have been reported [44,45,36].

Our work focuses on physics-constrained surrogate modeling for stochastic PDEs with high-dimensional spatially-varying coefficients *without* simulation data. We first show that when solving deterministic PDEs, the CNN-based parameterizations are more computationally efficient in capturing multiscale features of the solution fields than the FC-NN ones. Furthermore, we demonstrate that in comparison with image-to-image regression approaches that employ Deep NNs [9], the proposed method achieves comparable predictive performance, despite the fact that it does not make use of any output simulation data. In addition, it produces better predictions under extrapolative conditions as when out-of-distribution test input datasets are used. Finally, a flow-based conditional generative model is proposed to capture the predictive distribution with calibrated uncertainty, without compromising the predictive accuracy.

The paper is organized as follows. Section 2 provides the definition of the problems of interest including the solution of PDEs, surrogate modeling and uncertainty quantification. Section 3 provides the parametrization of the solutions with FC-NNs and CNNs, the physics-constrained learning of a deterministic surrogate and the variational learning of a probabilistic surrogate. Section 4 investigates the performance of the developed techniques with a variety of tests for various PDE systems. We conclude in Section 5 with a summary of this work and extensions to address limitations that have been identified.

[1] http://bayesiandeeplearning.org/.

2. Problem definition

Consider modeling of a physical system governed by PDEs:

$$\begin{aligned} \mathcal{N}(u(s); K(s)) &= f(s), \qquad s \in \mathcal{S}, \\ \mathcal{B}(u(s)) &= b(s), \qquad s \in \Gamma, \end{aligned} \tag{1}$$

where $\mathcal{N}$ is a general differential operator, $u(s)$ are the field variables of interest, $f(s)$ is the source field, and $K(s)$ denotes an input property field defining the system's constitutive behavior. $\mathcal{B}$ is the operator for boundary conditions defined on the boundary Γ of the domain $\mathcal{S}$. In particular, we consider the following Darcy flow problem as a motivating example throughout this paper:

$$-\nabla \cdot (K(s)\nabla u(s))) = f(s), \qquad s \in \mathcal{S}, \tag{2}$$

with boundary conditions

$$\begin{aligned} u(s) &= u_D(s), \qquad s \in \Gamma_D, \\ K\nabla u(s) \cdot \mathbf{n} &= g(s), \qquad s \in \Gamma_N, \end{aligned} \tag{3}$$

where $\mathbf{n}$ is the unit normal vector to the Neumann boundary Γ_N, Γ_D is the Dirichlet boundary.

Of particular interest are PDEs for which the field variables can be computed by appropriate minimization of a field energy functional (potential) $V(u; K)$, i.e.

$$\arg\min_u V(u; K). \tag{4}$$

Such potentials are common in many linear and nonlinear problems in physics and engineering and serve as the basis of the finite element method. For problems where such potentials cannot be found [46], one can consider V as the integral of the square of the residual norm of the PDE evaluated at different trial solutions, e.g.

$$V(u; K) = \int_{\mathcal{S}} R^2(u; K)\, ds. \tag{5}$$

In this paper, we are interested in the solution of parametric PDEs for a given set of boundary conditions.

Definition 2.1 *(Solution of a deterministic PDE system).* Given the potential $V(u; K)$, and the boundary conditions in Eq. (3), compute the solution $u(s)$ of the PDE for a given input field $K(s)$.

The input field $K(s)$ is often modeled as a random field $K(s, \omega)$ in the context of uncertainty quantification, where ω denotes a random event in the sample space Ω. In practice, discretized versions of this field are employed in computations which is denoted as the random vector $\mathbf{x}$, i.e. $\mathbf{x} = [K(s_1), \cdots, K(s_{n_s})]$. We note that when fine-scale fluctuations of the input field K are present, the dimension n_s of $\mathbf{x}$ can become very high. Let $p(\mathbf{x})$ be the associated density postulated by mathematical considerations or learned from data, e.g. CT scans of microstructures, measurement of permeability fields, etc. Suppose $\mathbf{y}$ denotes a discretized version of the PDE solution, i.e.

$$\mathbf{y} = [u(s_1), \cdots, u(s_{n_s})].$$

Note that all the discretized field variable(s) are denoted as bold, while the continuous field variable(s) are non-bold.

We are interested in developing a surrogate model that allows fast calculation of the system response $\mathbf{y}$ for any input realization $\mathbf{x} \in p(\mathbf{x})$, and potentially for various boundary conditions. This leads to the following definition:

Definition 2.2 *(Deterministic surrogate model).* Given the potential $V(u; K)$, the boundary conditions in Eq. (3), and a set of training input data $\mathcal{D}_{\text{input}} = \{\mathbf{x}^{(i)}\}_{i=1}^N, \mathbf{x}^{(i)} \sim p(\mathbf{x})$, learn a deterministic surrogate $\mathbf{y} = \hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x})$, for predicting the solution $\mathbf{y}$ for any input $\mathbf{x} \in p(\mathbf{x})$, where $\boldsymbol{\theta}$ denotes the parameters of the surrogate model.

Note that often the density $p(\mathbf{x})$ is not known and needs to be approximated from the given data $\{\mathbf{x}^{(i)}\}_{i=1}^N$. When the density $p(\mathbf{x})$ is given, the surrogate model can be defined without referring to the particular training data set. In this case, as part of the training process, one can select any dataset of size N, $\{\mathbf{x}^{(i)}\}_{i=1}^N, \mathbf{x}^{(i)} \sim p(\mathbf{x})$, including the most informative one for the surrogate task.

We note that the aforementioned problem refers to a new type of machine learning task that falls between unsupervised learning due to the absence of labeled data (i.e. the $\mathbf{y}^{(i)}$ corresponding to each $\mathbf{x}^{(i)}$ is *not* provided) and (semi-)supervised learning because the objective involves discovering the map from the input $\mathbf{x}$ to the output $\mathbf{y}$. Given the finite training data employed in practice and the inadequacies of the model postulated, $\hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x})$, it is often advantageous to obtain a distribution over the possible solutions via a probabilistic surrogate, rather than a mere point estimate for the solution.

Definition 2.3 *(Probabilistic surrogate model).* Given the potential $V(u;K)$, the boundary conditions in Eq. (3), and a set of training input data $\mathcal{D}_{\text{input}} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}, \mathbf{x}^{(i)} \sim p(\mathbf{x})$, a probabilistic surrogate model specifies a conditional density $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$, where $\boldsymbol{\theta}$ denotes the model parameters.

Finally, since the input $\mathbf{x}$ arises from an underlying probability density, one may be interested to compute the statistics of the output $\mathbf{y}$ leading to the following forward uncertainty propagation problem.

Definition 2.4 *(Forward uncertainty propagation).* Given the potential $V(u;K)$, the boundary conditions in Eq. (3), and a set of training input data $\mathcal{D}_{\text{input}} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}, \mathbf{x}^{(i)} \sim p(\mathbf{x})$, estimate moments of the response, $\mathbb{E}[\mathbf{y}], \text{Var}[\mathbf{y}], \ldots$ or more generally any aspect of the probability density of $\mathbf{y}$.

3. Methodology

3.1. Differentiable parameterizations of solutions

We only consider the parameterizations of solutions using neural networks, primarily FC-NNs and CNNs. Given one input $\mathbf{x} = [K(s_1), \cdots, K(s_{n_s})]$, most previous works [32,39,33,37] use FC-NNs to represent the solution as

$$u(s) = \hat{u}_{\boldsymbol{\phi}}(s), \tag{6}$$

where the input to the network is coordinate s, the output is the predicted solution at s, and $\hat{u}_{\boldsymbol{\phi}}$ denotes a FC-NN with parameters $\boldsymbol{\phi}$. The spatial gradients can be evaluated exactly by automatic differentiation. This approach yields a smooth representation of the solution that can be evaluated at any input location. Even though the outputs in this model at two different locations are correlated (as they both depend on the shared parameters $\boldsymbol{\phi}$ of the NN), FC-NNs do not have the inductive bias as in CNNs, e.g. translation invariance, parameter sharing, etc. Despite promising results in a series of canonical problems [47], the trainability and predictive performance of FC-NNs deteriorates as the complexity of the underlying solution increases. This drawback is confirmed by our numerical studies presented in Section 4.1 involving solution fields with multiscale features.

An alternative parametrization of the solution is through a convolutional decoder network

$$\mathbf{y} = \hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{z}), \tag{7}$$

where $\mathbf{y} = [u(s_1), \cdots, u(s_{n_s})]$ denotes the solution on pre-defined fixed grids $s_1, \cdots, s_{n_s}$ that is generated by one pass of the latent variable $\mathbf{z}$ through the decoder, similarly as in [48]. Note that $\mathbf{z}$ is usually much lower-dimensional than n_s and initialized arbitrarily. The spatial gradients can be approximated efficiently with Sobel filter,[2] which amounts to one convolution layer with fixed kernel, see Appendix A for details. In contrast to FC-NNs, convolutional architectures can directly capture complex spatial correlations and return a multi-resolution representation of the underlying solution field.

Remark 1. The dimensionality n_s of the input $\mathbf{x}$ is not required to be the same as that of the output $\mathbf{y}$. Since our CNN approach would involve operations between images including pixel-wise multiplication of input and output images (see Section 3.2.1), we select herein the same dimensionality for both inputs and outputs. Upsampling/downsampling can always be used to accommodate different dimensionalities n_{sx} and n_{sy} of the input and output images, respectively.

To solve the deterministic PDE for a given input, we can train the FC-NN solution as in Eq. (6) by minimizing the residual loss where the exact derivatives are calculated with automatic differentiation [32,39,33,37]. The loss functions for FC-NNs and CNNs to solve PDEs are detailed in Appendix B.1.

3.2. Physics-constrained learning of deterministic surrogates without labeled data

We are particularly interested in surrogate modeling with high-dimensional input and output, i.e. $\dim(\mathbf{x}), \dim(\mathbf{y}) \gg 1$. Surrogate modeling is an extension of the solution networks in the previous section by adding the realizations of stochastic input $\mathbf{x}$ as the input, e.g. $u(s,\mathbf{x}) = \hat{u}_{\boldsymbol{\phi}}(s,\mathbf{x})$ in the FC-NN case [36], or $\mathbf{y} = \hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x})$ in the CNN case [45].

Here, we adopt the *image-to-image regression* approach [9] to deal with the problem arising in practice where the realizations of the random input field are image-like data instead of being computed from an analytical formula. More specifically, the surrogate model $\mathbf{y} = \hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x})$ is an extension of the decoder network in Eq. (7) by prepending an encoder network to transform the high-dimensional input $\mathbf{x}$ to the latent variable $\mathbf{z}$, i.e. $\mathbf{y} = \text{decoder} \circ \text{encoder}(\mathbf{x})$.

In contrast to existing convolutional encoder-decoder network structures [9], the surrogate model studied here is trained *without* labeled data i.e. without computing the solution of the PDE. Instead, it is trained by learning to solve the PDE with given boundary conditions, using the following loss function

[2] https://www.researchgate.net/publication/239398674_An_Isotropic_3x3_Image_Gradient_Operator.

$$L(\boldsymbol{\theta}; \{\mathbf{x}^{(i)}\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^{N}\Big[V(\hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), \mathbf{x}^{(i)}) + \lambda B(\hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}))\Big], \tag{8}$$

where $\hat{\mathbf{y}}^{(i)} = \hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$ is the prediction of the surrogate for $\mathbf{x}^{(i)} \in \mathcal{D}_{\text{input}}$, $V(\hat{\mathbf{y}}^{(i)}, \mathbf{x}^{(i)})$ is the equation loss, either in the form of the *residual norm* [32] or the *variational functional* [35] of the PDE, $B(\hat{\mathbf{y}}^{(i)})$ is the boundary loss of the prediction $\hat{\mathbf{y}}^{(i)}$, and λ is the weight (Lagrange multiplier) to softly enforce the boundary conditions. Both $V(\hat{\mathbf{y}}^{(i)}, \mathbf{x}^{(i)})$ and $B(\hat{\mathbf{y}}^{(i)})$ may involve integration and differentiation with respect to the spatial coordinates, which are approximated with highly efficient discrete operations, detailed below for the Darcy flow problem. The surrogate trained with the loss function in Eq. (8) is called *physics-constrained surrogate* (PCS).

In contrast to the physically motivated loss function advocated above, a typical data-driven surrogate employs a loss function of the form

$$L_{\text{MLE}}(\boldsymbol{\theta}; \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{y}^{(i)} - \hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right\|_2^2, \tag{9}$$

where $\mathbf{y}^{(i)}$ is the output data for the input $\mathbf{x}^{(i)}$ which must be computed in advance. We refer to the surrogate trained with loss function in Eq. (9) as *data-driven surrogate* (DDS).

3.2.1. Loss function for Darcy flow

There are at least four variations of loss functions for a second-order elliptic PDE problem, depending on whether the field variables refer to the primal variable (pressure) or to mixed variables (pressure and fluxes), and whether the loss is expressed in strong form or variational form. Specifically, for the Darcy flow problem defined in Eq. (2), we can consider:

Primal residual loss. The residual norm for the primal variable is

$$V(u; K) = \int_{\mathcal{S}} \Big(\nabla \cdot (K\nabla u) + f\Big)^2 ds. \tag{10}$$

Primal variational loss. The energy functional is

$$V(u; K) = \int_{\mathcal{S}} \Big(\frac{1}{2}K\nabla u \cdot \nabla u - fu\Big) ds - \int_{\Gamma_N} gu\, ds. \tag{11}$$

Mixed formulation introduces an additional (vector) variable, namely flux τ, which turns Eq. (2) into a systems of equations

$$\begin{aligned} \tau &= -K\nabla u, \qquad \text{in } \mathcal{S}, \\ \nabla \cdot \tau &= f, \qquad \text{in } \mathcal{S}, \end{aligned} \tag{12}$$

with the same boundary conditions as in Eq. (3). $\tau(s) = [\tau_1(s), \tau_2(s)]$ are the flux field components along the horizontal and vertical directions, respectively.

Mixed variational loss. Following the Hellinger-Reissner principle [49], the mixed variational principle states that the solution (τ^*, u^*) of the Darcy flow problem is the unique critical point of the functional

$$V(\tau, u; K) = \int_{\mathcal{S}} \Big(\frac{1}{2}K^{-1}\tau \cdot \tau - u\nabla \cdot \tau + fu\Big) ds + \int_{\Gamma_D} u_D \tau \cdot \mathbf{n} ds, \tag{13}$$

over the space of vector fields $\tau \in \mathcal{H}(\text{div})$ satisfying the Neumann boundary condition and all the fields $u \in \mathcal{L}^2$. It should be highlighted that the solution (τ^*, u^*) is not an extreme point of the functional in Eq. (13), but a *saddle point*, i.e.

$$V(\tau^*, u) \le V(\tau^*, u^*) \le V(\tau, u^*).$$

Mixed residual loss. The residual norm for the mixed variables is

$$V(\tau, u; K) = \int_{\mathcal{S}} \Big[\Big(\tau + K\nabla u\Big)^2 + \Big(\nabla \cdot \tau - f\Big)^2\Big] ds. \tag{14}$$

Both the variational and mixed formulations have the advantage of lowering the order of differentiation which is approximated numerically in our implementation by a Sobel filter, as detailed in Appendix A. For example by employing the discretized representation $\mathbf{x}$ for K where the domain is $\mathcal{S} = [0, 1] \times [0, 1]$, the mixed residual loss is evaluated as

$$V(\boldsymbol{\tau}, \mathbf{u}; \mathbf{x}) \approx \frac{1}{n_s}\Big(\|\boldsymbol{\tau} + \mathbf{x} \odot \nabla \mathbf{u}\|_2^2 + \|\nabla \cdot \boldsymbol{\tau} - \mathbf{f}\|_2^2 \Big), \tag{15}$$

where n_s is the number of uniform grids, $\nabla \mathbf{u} = [\mathbf{u}_h, \mathbf{u}_v]$, $\mathbf{u}_h, \mathbf{u}_v$ are two gradient images along the horizontal and vertical directions estimated by Sobel filter, similarly for $\nabla \cdot \boldsymbol{\tau} = (\boldsymbol{\tau}_1)_h + (\boldsymbol{\tau}_2)_v$, and $\odot$ denotes the element-wise product.

3.3. Probabilistic surrogates with reverse KL formulation

While a deterministic surrogate provides fast predictions to new input realizations, it does not model the predictive uncertainty which is important in practice especially when the surrogate is tested on unseen (during training) inputs. Moreover, many PDEs in physics have multiple solutions [50] which cannot be captured with a deterministic model. Thus building probabilistic surrogates that can model the distribution over possible solutions given the input is of great importance.

A probabilistic surrogate models the conditional density of the predicted solution given the input, i.e. $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$. Instead of learning this conditional density with labeled data [51–53], we distill it from a reference density $p_\beta(\mathbf{y}|\mathbf{x})$. The reference density is a Boltzmann-Gibbs distribution

$$p_\beta(\mathbf{y}|\mathbf{x}) = \frac{\exp(-\beta L(\mathbf{y}, \mathbf{x}))}{Z_\beta(\mathbf{x})}, \tag{16}$$

where $L(\mathbf{y}, \mathbf{x}) = V(\mathbf{y}, \mathbf{x}) + \lambda B(\mathbf{y})$ is the loss function (Eq. (8)) for the deterministic surrogate that penalizes the violation of the PDE and boundary conditions, and the parameter β that plays a role similar to the inverse temperature in statistical mechanics is used here to control the overall variance of the reference density. This energy-based model is obtained solely from the PDE and boundary conditions, without having access to labeled output data [54]. However, this PDE-constrained model provides similar information as the labeled data allowing us to learn a probabilistic surrogate.

Since sampling from the probabilistic surrogate $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ is usually fast and evaluating the (unnormalized) reference density $p_\beta(\mathbf{y}|\mathbf{x})$ is often cheap, we choose to minimize the following reverse KL divergence:

$$\begin{aligned} D_{\mathrm{KL}}(p(\mathbf{x})\, p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \,\|\, p(\mathbf{x})\, p_\beta(\mathbf{y}|\mathbf{x})) &= \mathbb{E}_{p(\mathbf{x})}\Big[-\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}[\log p_\beta(\mathbf{y}|\mathbf{x})] + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})]\Big] \\ &= \beta \mathbb{E}_{p(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}[L(\mathbf{y}, \mathbf{x})] - \mathbb{H}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) + \mathbb{E}_{p(\mathbf{x})}[\log Z_\beta(\mathbf{x})]. \end{aligned} \tag{17}$$

The first term is the expectation of the loss function $L(\mathbf{y}, \mathbf{x})$ w.r.t. the joint density $p(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$, which enforces the satisfaction of PDEs and boundary conditions. The second term is the negative conditional entropy of $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ which promotes the diversity of model predictions. It also helps to stabilize the training of the flow-based conditional generative model introduced in Section 3.3.1. The third term is the expected negative free energy, which is constant when optimizing $\boldsymbol{\theta}$. For the models with intractable log-likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$, one can derive a lower bound for the conditional entropy $\mathbb{H}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ that helps to regularize training and avoid mode collapse as in [55]. In this work, the log-likelihood can be exactly evaluated for the model introduced in Section 3.3.1.

This idea is similar to probability density distillation [56] to learn generative models for real-time speech synthesis, neural renormalization group [57] to accelerate sampling for Ising models, and Boltzmann generators [58] to efficiently sample equilibrium states of many-body systems.

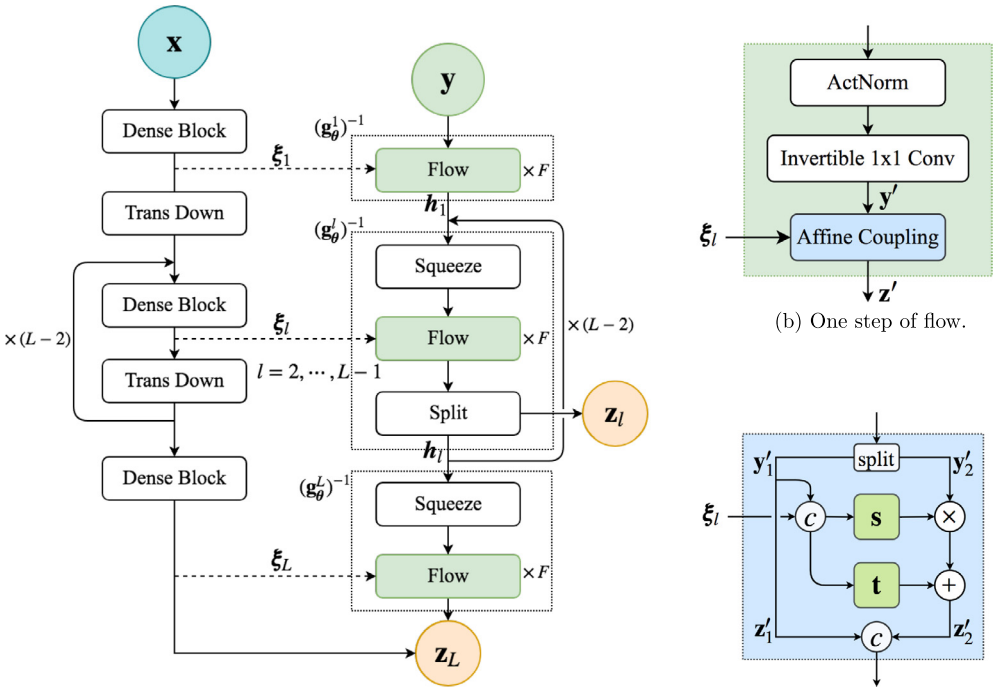
The reverse KL divergence itself is not enough to guarantee that the predictive uncertainty is well-calibrated. Even if this divergence is optimized to zero, i.e. $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) = p_\beta(\mathbf{y}|\mathbf{x})$, the predictive uncertainty is still controlled by β. Thus we add an uncertainty calibration constraint to the optimization problem, i.e.

$$\begin{aligned} &\min_{\beta, \boldsymbol{\theta}} \quad D_{\mathrm{KL}}(p(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \,\|\, p(\mathbf{x})p_\beta(\mathbf{y}|\mathbf{x})), \\ &s.t. \quad p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \text{ is calibrated on validation data.} \end{aligned} \tag{18}$$

Here, the predictive uncertainty is calibrated using the reliability diagram [59]. The naive approach to select β is through grid search, i.e. train the probabilistic surrogate with different values of β, and select the one under which the trained surrogate is well-calibrated w.r.t. validation data, which includes input-output data pairs.

Remark 2. Instead of tuning β with grid search, we can also re-calibrate the trained model *post-hoc* [60,61] by learning an auxiliary regression model. For a small amount of miscalibration, sampling latent variables with different temperature (Section 6 in [62]) can also change the variance of the output with a slight drop in predictive accuracy.

Remark 3. Similar to our approach, Probabilistic Numerical Methods (PNMs) [63–65] take a statistical point of view of classical numerical methods (e.g. a finite element solver) that treat the output as a point estimate of the true solution. Given finite information (e.g. finite number of evaluations of the PDE operator and boundary conditions) and prior belief about the solution, PNMs output the posterior distribution of the solution. PNM focuses on inference of the solution for one input, instead of amortized inference as what the probabilistic surrogate does.



(a) Mutliscale conditioning. (b) One step of flow. (c) Affine coupling layer.

Fig. 1. Multiscale conditional Glow. (a) Multiscale features extracted with the encoder network (left) are used as conditions to generate output with the Glow model (right). $\times F$, $\times(L-2)$ denotes repeating for F times and $L-2$ times respectively. (b) One step of flow, i.e. `Flow` block in (a), and (c) Affine coupling layer following the structure of Glow (Fig. 2 in [62]) except conditioning on the input features. The figure shows the forward path from $\{\mathbf{y};\mathbf{x}\}$ to $\mathbf{z}=\{\mathbf{z}_2,\cdots,\mathbf{z}_L\}$. The reverse (sampling) path from $\{\mathbf{z};\mathbf{x}\}$ to $\mathbf{y}$ is used during training, where $\mathbf{z}$ are sampled from diagonal Gaussians, see Algorithm 1. See Appendix C for the details of all modules in the model.

3.3.1. *Conditional flow-based generative models*

This section presents flow-based generative models [66] as our probabilistic surrogates. This family of models offers several advantages over other generative models [67,68], such as exact inference and exact log-likelihood evaluation that is particularly attractive for learning the conditional distribution with the reverse KL divergence as in Eq. (17). The generative model $\mathbf{y}=\mathbf{g}_\theta(\mathbf{z})$ consists of a sequence of *invertible* layers (also called normalizing flows [69]) that transforms a simple distribution $p(\mathbf{z})$ to a target distribution $p(\mathbf{y})$, i.e.

$$\mathbf{y}:=\mathbf{h}_0 \overset{\mathbf{g}_\theta^1}{\longleftrightarrow} \mathbf{h}_1 \overset{\mathbf{g}_\theta^2}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{\mathbf{g}_\theta^L}{\longleftrightarrow} \mathbf{h}_L:=\mathbf{z},$$

where $\mathbf{g}_\theta=\mathbf{g}_\theta^1\circ\mathbf{g}_\theta^2\circ\cdots\circ\mathbf{g}_\theta^L$. By the change of variables formula, the log-likelihood of the model given $\mathbf{y}$ can be calculated as

$$\log p_\theta(\mathbf{y})=\log p_\theta(\mathbf{z})+\sum_{l=1}^{L}\log|\det(d\mathbf{h}_l/d\mathbf{h}_{l-1}))|,$$

where the log-determinant of the absolute value of the Jacobian term $\log|\det(d\mathbf{h}_l/d\mathbf{h}_{l-1}))|$ for each transform $(\mathbf{g}_\theta^l)^{-1}$ can be easily computed for certain design of invertible layers [69,66] similar to the Feistel cipher. Given training data of $\mathbf{y}$, the model can be optimized stably with maximum likelihood estimation.

A recently developed generative flow model called Glow [62] proposed to learn invertible 1×1 convolution to replace the fixed permutation and synthesize large photo-realistic images using the log-likelihood objective. We extend Glow to condition on high-dimensional input $\mathbf{x}$, e.g. images, as shown in Fig. 1. The conditional model consists of two components (Fig. 1a): an encoder network which extracts multiscale features $\{\boldsymbol{\xi}_l\}_{l=1}^L$ from the input $\mathbf{x}$ through a cascade of alternating dense blocks and downsampling layers, and a Glow model (with multiscale structure) which transforms the latent variables $\mathbf{z}=\{\mathbf{z}_2,\cdots,\mathbf{z}_L\}$ distributed at different scales to the output $\mathbf{y}$ conditioned on $\{\boldsymbol{\xi}_l\}_{l=1}^L$ through skip connections (dashed lines in Fig. 1a, as in Unet [70]) between the encoder and the Glow model. The multiscale nature of the model arises since the intermediate features $\{\boldsymbol{\xi}_l\}_{l=1}^L$, and latent variables $\{\mathbf{z}_l\}_{l=2}^L$ are of different spatial sizes.

More specifically, the input features $\boldsymbol{\xi}_l$ enter the Glow model as the condition for the affine coupling layers at the same scale, as shown in Fig. 1b, whose input and output are denoted as $\mathbf{y}'$ and $\mathbf{z}'$ in the forward path. As shown in Fig. 1c, the

input features $\boldsymbol{\xi}_l$ are concatenated Ⓒ with half of the flow features $\mathbf{y}'_1$ before passing to scale $\boxed{\mathbf{s}}$ and shift $\boxed{\mathbf{t}}$ networks which specify arbitrarily nonlinear transforms that need not to be invertible. Given $\mathbf{z}' = [\mathbf{z}'_1, \mathbf{z}'_2]$ and $\boldsymbol{\xi}_l$, $\mathbf{y}' = [\mathbf{y}'_1, \mathbf{y}'_2]$ can be recovered exactly by reversing the shift and scaling operations, as detailed in Table C.1. Note that $\boldsymbol{\xi}_l$ is the condition for all F steps of flow at scale $l = 1, \cdots, L$, where L denotes the number of scales (or levels). More details of the model including Dense Blocks, Transition Down layers, split, squeeze, activation normalization, invertible 1×1 convolution and affine coupling layers are given in Appendix C.

In a data-driven scenario, the conditional Glow is trained by passing data $\mathbf{y}$ through the model to compute the latent $\mathbf{z}$ and maximizing the evaluated log-likelihood of data given $\mathbf{x}$. But to train with the loss in Eq. (17), we need to sample the output $\hat{\mathbf{y}}$ from the conditional density $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ given $\mathbf{x}$, which goes in the opposite direction of the data-driven case. Algorithm 1 shows the details of training conditional Glow. The sampling/generation process is shown within the outer for-loop before computing the loss. Note that for one input sample only one output sample is used to approximate the expectation over $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ during training. To obtain multiple output samples for an input e.g. so as to compute the predictive mean and variance during prediction, we only need to sample the noise variables $\{\boldsymbol{\epsilon}_l\}_{l=2}^{L}$ multiple times for one given input. The conditional log-likelihood $p_{\boldsymbol{\theta}}(\hat{\mathbf{y}}|\mathbf{x})$ can be exactly evaluated as the following:

$$\log p_{\boldsymbol{\theta}}(\hat{\mathbf{y}}|\mathbf{x}) = \log p_{\boldsymbol{\theta}}(\mathbf{z}) + \log|\det(d\mathbf{z}/d\hat{\mathbf{y}})|, \tag{19}$$

where both the latent $\mathbf{z}$ and $\log|\det(d\mathbf{z}/d\hat{\mathbf{y}})|$ depend on $\mathbf{x}$ and realizations of the noise $\{\boldsymbol{\epsilon}_l\}_{l=2}^{L}$. The density of the latent $p_{\boldsymbol{\theta}}(\mathbf{z})$ is usually a simple distribution, e.g. diagonal Gaussian, which is computed with the second (for $\{\mathbf{z}_l\}_{l=2}^{L-1}$) and third (for $\mathbf{z}_L$) terms within the bracket of the reverse KL divergence loss in Algorithm 1. Also $\log|\det(d\mathbf{z}/d\mathbf{y})|$ is computed with the fourth term. Appendix C provides the formula to compute $\log|\det(d\mathbf{h}_l^{(i)}/d\mathbf{h}_{l-1}^{(i)})|$, which is the sum of the log-determinant of the Jacobian for `ActNorm`, the `Invertible 1×1 Conv` and the `Affine Coupling` layer. Notably, the log-determinant of the Jacobian for the affine coupling layer is just $\mathtt{sum}(\log|\mathbf{s}|)$, where $\mathbf{s}$ is the output of the scaling network. Thus the conditional density $p_{\boldsymbol{\theta}}(\hat{\mathbf{y}}|\mathbf{x})$ can be evaluated exactly and efficiently, enabling us to directly approximate the entropy term in Eq. (17), e.g. via Monte Carlo approximation.

The model parameters $\boldsymbol{\theta}$ include the parameters in the encoder (dense blocks and transition down layers) as well as the parameters in the Glow model (the scale and shift networks in affine coupling layers, scale and bias parameters in ActNorm, kernel matrix in 1×1 convolution, and convolution layer for the diagonal Gaussian for the latent variables $\mathbf{z}_l$).

Remark 4. The training process does not require output data. However, validation data with input-output pairs are necessary to calibrate the predictive uncertainty of the trained model. Careful initialization of the model is important to stabilize the training process. In this work, we initialize the `ActNorm` to be the identity transform, the weight matrix of `Invertible` 1×1 `Convolution` to be a random rotation matrix, and the `Affine Coupling` layer to be close to the identity transform ($\hat{\mathbf{s}} = \mathbf{0}$ and $\mathbf{t} = \mathbf{0}$ in Table C.1). We can also use data-dependent initialization to speed up the training process. More specifically, one mini-batch $\mathcal{D}_{\text{init}} = \{(\mathbf{x}^{(j)}, \mathbf{r}^{(j)})\}_{j=1}^{M'}$ (e.g. $M' = 32$) of input-output data pairs can be passed forward from $\{\mathbf{y}; \mathbf{x}\}$ to $\mathbf{z}$ to initialize the parameters of `ActNorm` such that the post-`ActNorm` activations per-channel have zero mean and unit variance given $\mathcal{D}_{\text{init}}$ [62]. The reference output $\mathbf{r}$ can be the solution from standard deterministic PDE solvers or more appropriately here from the methods presented in Sections 3.1 and 4.1.

Algorithm 1 Training conditional Glow.

Input: Inverse temperature β, input samples $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$. Mini-batch size M, number of steps F of `Flow` in each scale, number of scales L.
Output: Model parameters $\boldsymbol{\theta}$.
for *number of training iterations* **do**
  Sample a mini-batch of input $\{\mathbf{x}^{(i)}\}_{i=1}^{M}$, pass it through the encoder to compute the multiscale input features $\{\boldsymbol{\xi}_l^{(i)}\}_{l=1,i=1}^{L,M}$;
  Sample the latent $\mathbf{z}_L^{(i)} = \boldsymbol{\mu}_{\boldsymbol{\theta}}^L(\boldsymbol{\xi}_L^{(i)}) + \boldsymbol{\sigma}_{\boldsymbol{\theta}}^L(\boldsymbol{\xi}_L^{(i)}) \odot \boldsymbol{\epsilon}_L^{(i)}, \boldsymbol{\epsilon}_L^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  Compute flow feature $\mathbf{h}_{L-1}^{(i)} = \mathbf{g}_{\boldsymbol{\theta}}^L(\mathbf{h}_L^{(i)}; \boldsymbol{\xi}_L^{(i)})$; ▷ $\mathbf{h}_L = \mathbf{z}_L$, $\mathbf{g}_{\boldsymbol{\theta}}^L$ includes the reverse path of (`Sequeeze`, F steps of `Flow`)
  for $l = L-1 : 2$ **do**
    Sample the split latent variable at level l $\mathbf{z}_l^{(i)} = \boldsymbol{\mu}_{\boldsymbol{\theta}}^l(\mathbf{h}_l^{(i)}) + \boldsymbol{\sigma}_{\boldsymbol{\theta}}^l(\mathbf{h}_l^{(i)}) \odot \boldsymbol{\epsilon}_l^{(i)}, \boldsymbol{\epsilon}_l^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), i = 1, \cdots, M$;
    Compute flow feature $\mathbf{h}_{l-1}^{(i)} = \mathbf{g}_{\boldsymbol{\theta}}^l(\mathbf{h}_l^{(i)}, \mathbf{z}_l^{(i)}; \boldsymbol{\xi}_l^{(i)})$; ▷ $\mathbf{g}_{\boldsymbol{\theta}}^l$ includes the reverse path of (`Sequeeze`, F steps of `Flow` and `Split`)
  end
  Compute output $\hat{\mathbf{y}}^{(i)} = \mathbf{g}_{\boldsymbol{\theta}}^1(\mathbf{h}_1^{(i)}; \boldsymbol{\xi}_1^{(i)})$; ▷ $\mathbf{g}_{\boldsymbol{\theta}}^1$ includes the reverse path of F steps of `Flow`
  Minimize the *reverse KL* divergence in Eq. (17) with Adam optimizer w.r.t. $\boldsymbol{\theta}$,
  $\frac{1}{M}\sum_{i=1}^{M}\Big[\beta L(\hat{\mathbf{y}}^{(i)}, \mathbf{x}^{(i)}) + \sum_{l=2}^{L-1}\log\mathcal{N}(\mathbf{z}_l^{(i)}|\boldsymbol{\mu}_{\boldsymbol{\theta}}^l(\mathbf{h}_l^{(i)}), (\boldsymbol{\sigma}_{\boldsymbol{\theta}}^l(\mathbf{h}_l^{(i)}))^2) + \log\mathcal{N}(\mathbf{z}_L^{(i)}|\boldsymbol{\mu}_{\boldsymbol{\theta}}^L(\boldsymbol{\xi}_L^{(i)}), (\boldsymbol{\sigma}_{\boldsymbol{\theta}}^L(\boldsymbol{\xi}_L^{(i)}))^2) + \sum_{l=1}^{L}\log|\det(d\mathbf{h}_l^{(i)}/d\mathbf{h}_{l-1}^{(i)})|\Big]$.
  ▷ $\mathbf{h}_0 = \hat{\mathbf{y}}$.
end

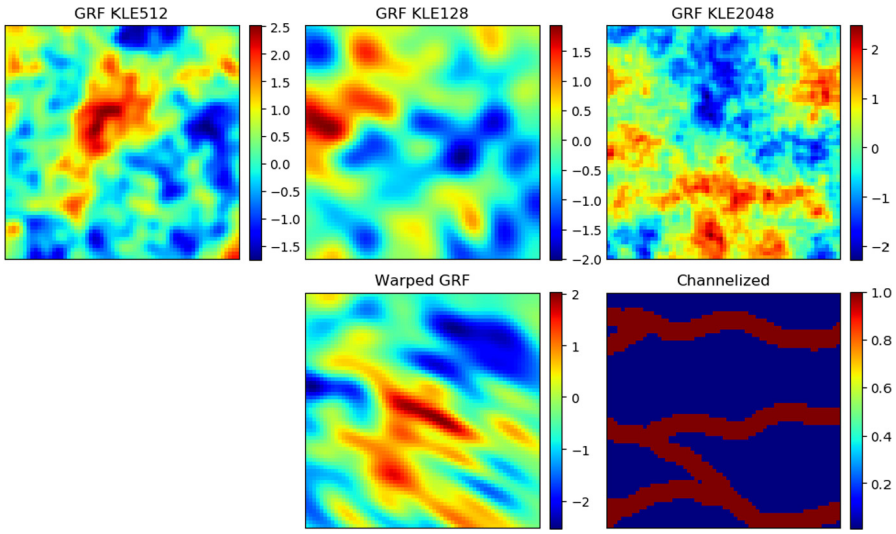


Fig. 2. Samples from 5 test input distributions over a 64×64 uniform grid, i.e. GRF KLE512, GRF KLE128, GRF KLE2048, warped GRF, channelized field. Log permeability samples are shown except the last channelized field that is defined with binary values 0.01 and 1.0. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4. Numerical experiments

Model problem. Steady-state flow in random heterogeneous media is studied as the model problem throughout the experiments, as in Eqs. (2), (12), (3). We consider the domain $\mathcal{S} = [0, 1] \times [0, 1]$, the left and right boundaries are Dirichlet, with pressure values 1 and 0, respectively. The upper and lower boundaries are Neumann, with zero flux. The source field is zero.

Dataset. Only input samples are needed to train the physics-constrained surrogates (PCSs). Additional simulated output data for training data-driven surrogates (DDSs) and evaluating surrogate performance are obtained with FEniCS [71]. Here, we mainly introduce three types of input datasets, which are Gaussian random field (GRF), warped GRF, and channelized field.

The first input dataset is the exponential of a GRF, i.e. $K(s) = \exp(G(s))$, $G(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot))$, where $k(s, s') = \exp(-\left\|s - s'\right\|_2 / l)$, l is the length scale. The field realization is generated with Karhunen-Loève expansion (KLE) with the leading N terms, paired with Latin hypercube sampling. See Section 4.1 in [9] for more details. This type of dataset is called `GRF KLEN`. For the deterministic surrogate experiments in Section 4.2, the training input GRF KLE512 is generated with length scale $l = 0.25$, $N = 512$ leading terms, discretized over a 64×64 uniform grid, which accumulates 95.04% energy. For the probabilistic surrogate in Section 4.3, the parameters for the training input GRF KLE100 are $N = 100$, $l = 0.2$, over 32×32 uniform grid. The test set may have other KLE truncations, but with the same length scale in each case, i.e. $l = 0.25$ for 64×64, and $l = 0.2$ for 32×32. The dataset for uncertainty propagation consists of 10,000 input-output data pairs unseen during training.

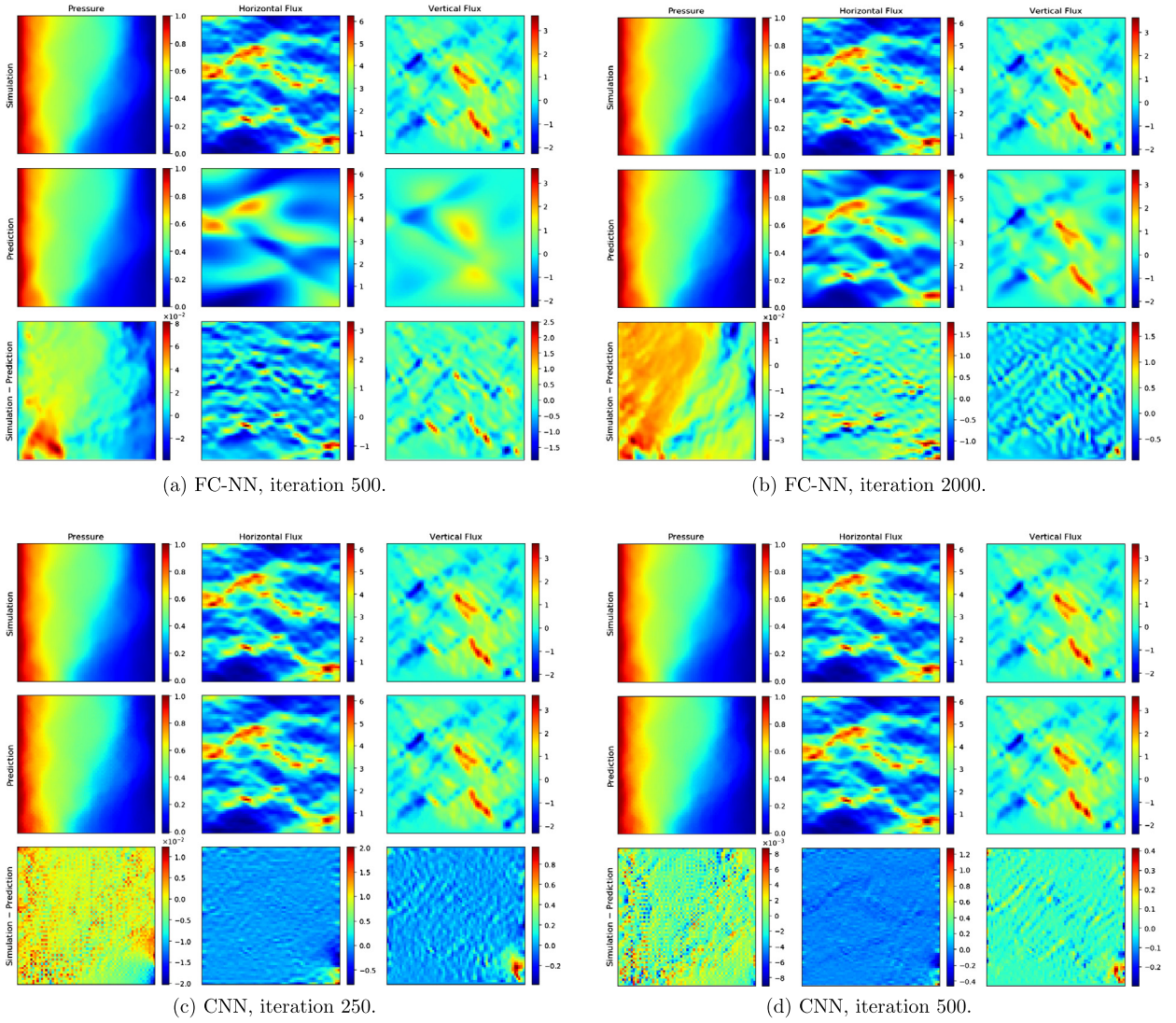
A slightly different test input is `warped GRF` [5], where there are two Gaussian fields and the output of the first GRF is the input to the second GRF. The kernel for both GRFs is squared exponential kernel, the length scale and KLE terms are $2, 16$ for the first GRF and $0.1, 128$ for the second GRF.

The last type of input field considered is a `channelized` field. Samples are obtained by cropping 64×64 patches from one large training image [72] of size 2500×2500, or 32×32 patches from the resized 1250×1250 image (resized with nearest neighborhood). Typical samples of the input datasets considered are shown in Fig. 2.

We begin our experiments by solving deterministic PDEs with spatially-varying coefficient (input) with convolutional decoder networks, and compare with FC-NNs. Then we show experiments for surrogate modeling for solving random PDEs, and compare with the data-driven approach. The last part is on experiments of using the conditional Glow as our probabilistic surrogate for uncertainty quantification tasks. The code and datasets for this work are available at https://github.com/cics-nd/pde-surrogate.

4.1. Solving Deterministic PDEs

In this section, we explore the relative merit of using CNNs and FC-NNs to parameterize the solutions of deterministic PDEs with image-like input field, including both linear and nonlinear PDEs. Since our focus is on surrogate modeling, the results below are mostly qualitative. The network architectures and training details are described in Appendix B.2.



(a) FC-NN, iteration 500.

(b) FC-NN, iteration 2000.

(c) CNN, iteration 250.

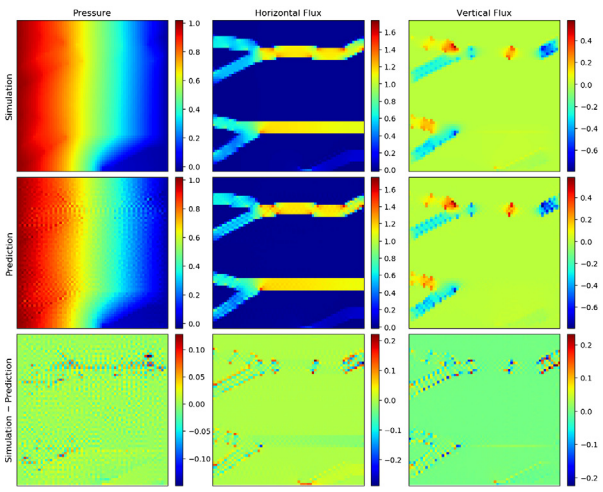
(d) CNN, iteration 500.

Fig. 3. Solving Darcy flow for one sample from GRF KLE1024 under mixed residual loss. The FC-NN takes much longer to resolve the fine details of flux fields, while the pressure field does not improve much. The CNN obtains more accurate solutions in fewer iterations.

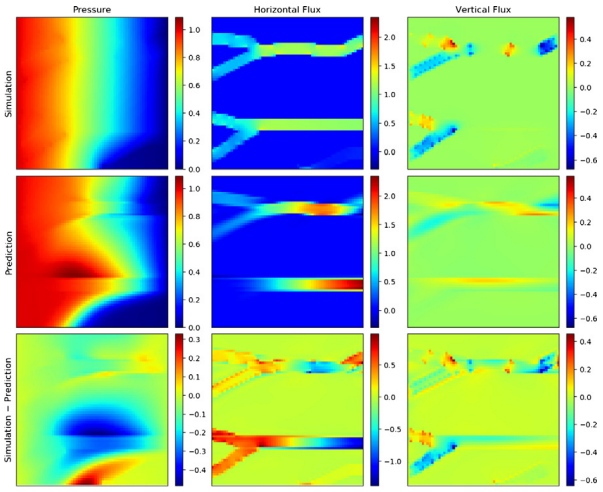
Comparison of CNNs and FC-NNs to solve Darcy flow. We compare convolutional decoder networks and fully-connected networks presented in Section 3.1 to solve the PDE system in Eq. (2). The input permeability field is sampled from GRF KLE1024 over a 64×64 uniform grid. We optimize the CNN and the FC-NN with mixed residual loss using L-BFGS optimizer for 500 and 2000 iterations, respectively. The results are shown in Fig. 3. The solution learned with the CNN in iteration 250 is even better than the solution learned with the FC-NN in iteration 2000, in terms of accuracy and retaining multiscale features of the flux fields. The same phenomenon is observed for input GRFs with other intrinsic dimensionalities. We further experiment on input sampled from the channelized field, as shown in Fig. 4. For this case, however, we observe that the FC-NN fails to converge to a small enough error in contrast to the CNN.

The experiments on solving deterministic PDEs show that CNNs can capture the multiscale features of the solution much more effectively than the FC-NNs, as reflected by the resolved flux fields. This is mostly because of the difference in their parameterizations of a field solution and the ways to obtain spatial gradients. FC-NNs tend to generate images that look like light-paintings,[3] but not rugged fields. More broadly, this type of parameterization is intensively explored named compositional pattern producing networks [73]. CNNs can represent images with multiscale features quite efficiently as is evident in our experiments and the rapid advances in image generation applications. Due to the discretization of spatial gradients with Sobel filters, the error of the learned solution is mainly on the boundaries, and the checkerboard

[3] https://distill.pub/2018/differentiable-parameterizations/# section-xy2rgb.



(a) CNN, iteration 500.



(b) FC-NN, iteration 2000.

Fig. 4. Solving Darcy flow for one sample of the channelized field. The same network and training setup as in Fig. 3 are used. The FC-NN parameterization fails to converge.

artifact becomes more severe in the pressure field as the flux fields becomes more rugged, as shown in Fig. 5 for GRF KLE4096. Again the CNN can capture the flux field much faster and better than the FC-NN, but in this case the pressure field begins to show severe checkerboard artifact that the largest error being larger than that of the pressure solution of the FC-NN.

Nonlinear flow in porous media. Darcy's law $\tau = -K\nabla u$ is a well established linear constitutive relationship for flow through porous media when the Reynolds number Re approaches zero. It has been shown both theoretically [74] and experimentally [75,76] that the constitutive relation undergoes a cubic transitional regime at low Re, and then a quadratic Forchheimer [77] when $Re \sim O(1)$. To show that our approach also works for nonlinear PDEs, we look at the nonlinear correction of Darcy's law as the following

$$-\nabla u = \frac{1}{K}\tau + \frac{\alpha_1}{K^{\frac{1}{2}}}\tau^2 + \alpha_2\tau^3, \tag{20}$$

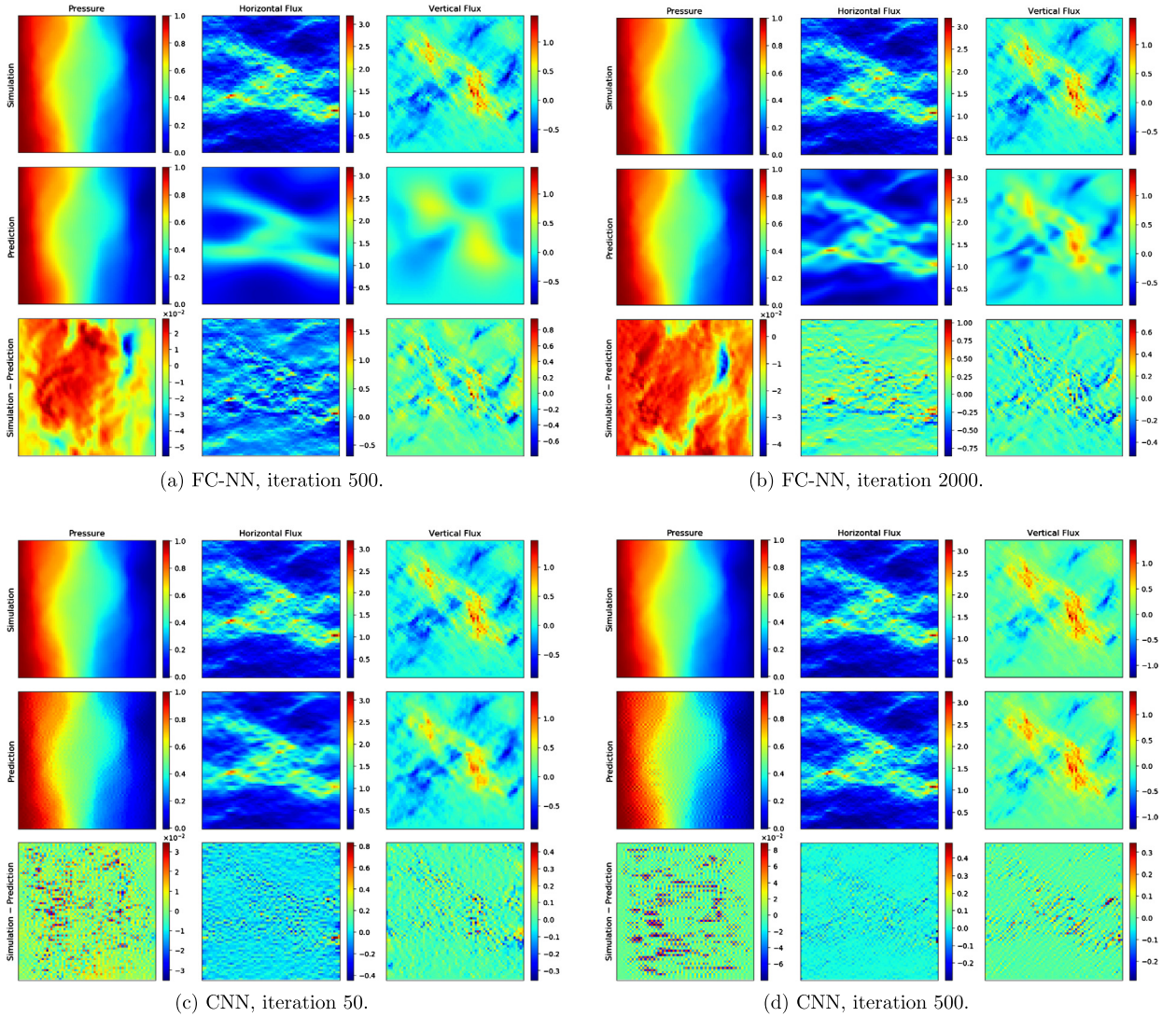
where α_1, α_2 are usually obtained by fitting to experiment data. We use CNNs to solve this nonlinear flow with the constitutive Eq. (20), the continuity equation $\nabla \cdot \tau = 0$ and the same boundary condition with the linear Darcy case. The reference solution is obtained with FEniCS (dual mixed formulation with Newton solver that converges in $5 \sim 6$ iterations with relative tolerance below 10^{-6}). We experiment on input fields from GRF KLE1024 and the channelized field, with $\alpha_1 = 0.1$ and $\alpha_2 = 0.1$ in the first case, and $\alpha_1 = 1.0$ and $\alpha_2 = 1.0$ in the second case. The convolutional decoder network is the same as in the previous section, and is trained with mixed residual loss. The results are shown in Fig. 6.

For GRF KLE1024, the effect of the cubic constitutive relation is actually smoothing out the flux field in comparison to the linear case in Fig. 3 using the same input field. The nonlinearity of PDEs does not seem to increase the burden for the CNN training except for a few more steps of forward and backpropagation due to the nonlinear operations in the constitutive equation. This is a negligible cost w.r.t. the computations in the decoder network itself. However, note that solving nonlinear PDEs with the Newton solver requires N iterations, thus increasing the computation by N times. For surrogate modeling, the mapping that the CNN learns from K to u is nonlinear even when the PDE to solve is linear. We expect it will be easier to learn a surrogate in the nonlinear case due to the smoother output fields. We leave further investigation of surrogate modeling and uncertainty quantification for nonlinear stochastic PDEs for our future work.

4.2. Deterministic surrogate

The experiments in solving deterministic PDEs lead us to choose CNNs over FC-NNs for surrogate modeling, with less training time and comparable accuracy, especially for high-dimensional input. We train both the physics-constrained surrogates and data-driven surrogates, and compare their accuracy and generalizability.

Network. Dense convolutional encoder-decoder network [9] is used as the surrogate model, with one input channel $\mathbf{x}$ and three output channels $[\mathbf{u}, \boldsymbol{\tau}_1, \boldsymbol{\tau}_2]$, as shown in Fig. 7. The upsampling method in the decoding layers in the current implementation is nearest upsampling followed by convolution, different from transposed convolution used in the data-driven



(a) FC-NN, iteration 500.

(b) FC-NN, iteration 2000.

(c) CNN, iteration 50.

(d) CNN, iteration 500.

Fig. 5. Solving Darcy flow for one sample from GRF KLE4096 under mixed residual loss.

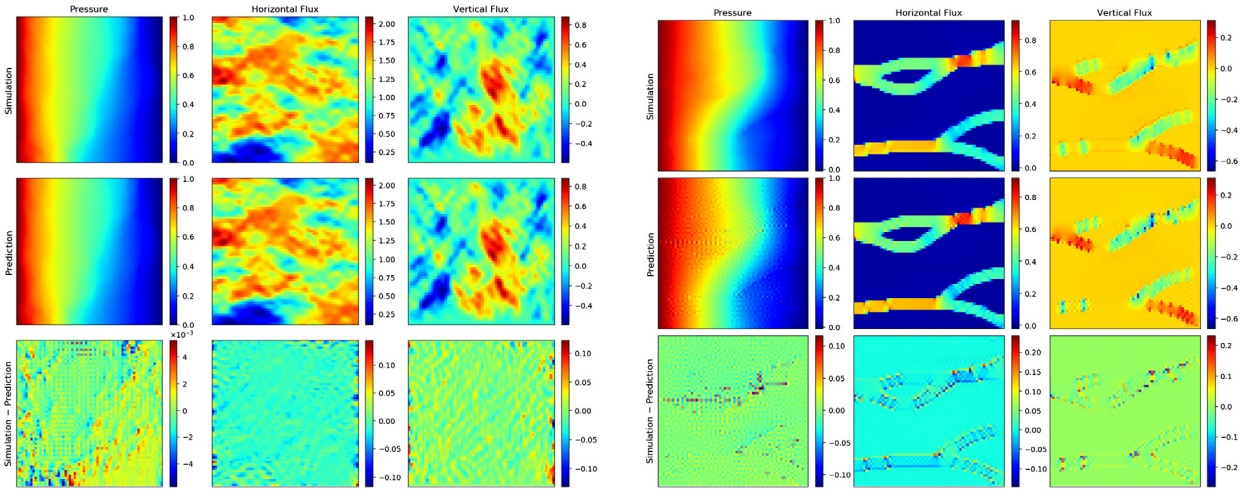
case. This is essential to avoid the checkerboard effect,[4] partially severed by Sobel filter besides the natural tendency of transposed convolution. The resolution of the input fields is reduced by 4 times through the encoding path, from 64×64 to 16×16, then increased to the size of the output fields, 64×64. The number of layers in the three dense blocks are $6, 8, 6$, with growth rate 16. There are 48 initial feature maps after the first convolution layer.

Training. We train the PCS with mixed residual loss as in Eq. (15) with only input data, and compare it with the DDS with the same network architecture but trained with additional output data. The number of training data, mini-batch size and the category of test distributions vary in different experiments, but all with $T = 512$ test data and employing the Adam [78] optimizer paired with one cycle policy[5] (learning rate scheduler) where the maximum learning rate is 0.001. The mini-batch size ranges from 8 to 32 depending on the number of training data. The weight coefficient for the boundary conditions is $\lambda = 10$. The evaluation metrics for prediction are relative L_2 error and R^2 score,

$$\epsilon_j = \frac{1}{T}\sum_{i=1}^{T}\frac{\left\|\hat{\mathbf{y}}_j^{(i)} - \mathbf{y}_j^{(i)}\right\|_2}{\left\|\mathbf{y}_j^{(i)}\right\|_2}, \qquad R_j^2 = 1 - \frac{\sum_{i=1}^{T}\left\|\hat{\mathbf{y}}_j^{(i)} - \mathbf{y}_j^{(i)}\right\|_2^2}{\sum_{i=1}^{T}\left\|\mathbf{y}_j^{(i)} - \bar{\mathbf{y}}_j\right\|_2^2}, \tag{21}$$

[4] https://distill.pub/2016/deconv-checkerboard/.

[5] https://github.com/fastai/fastai/blob/master/fastai/callbacks/one_cycle.py.



(a) GRF KLE1024. (b) Channelized field.

Fig. 6. Simulation (FEniCS) and learned solution (prediction) with CNNs for the nonlinear flow for (a) GRF KLE1024 with $\alpha_1 = 0.1$ and $\alpha_2 = 0.1$, and (b) channelized field with $\alpha_1 = 1.0$ and $\alpha_2 = 1.0$.

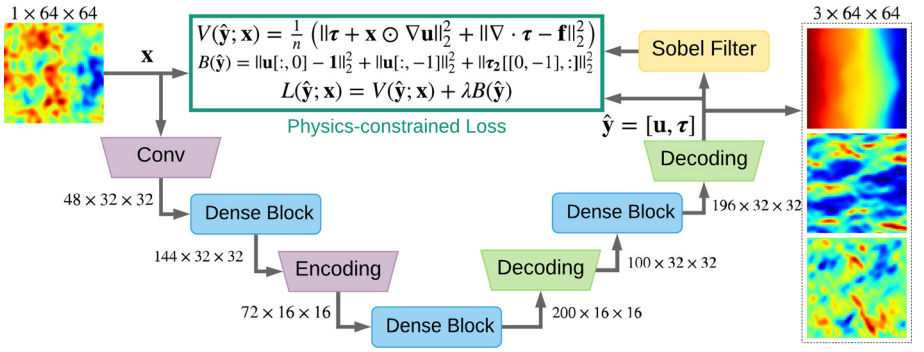


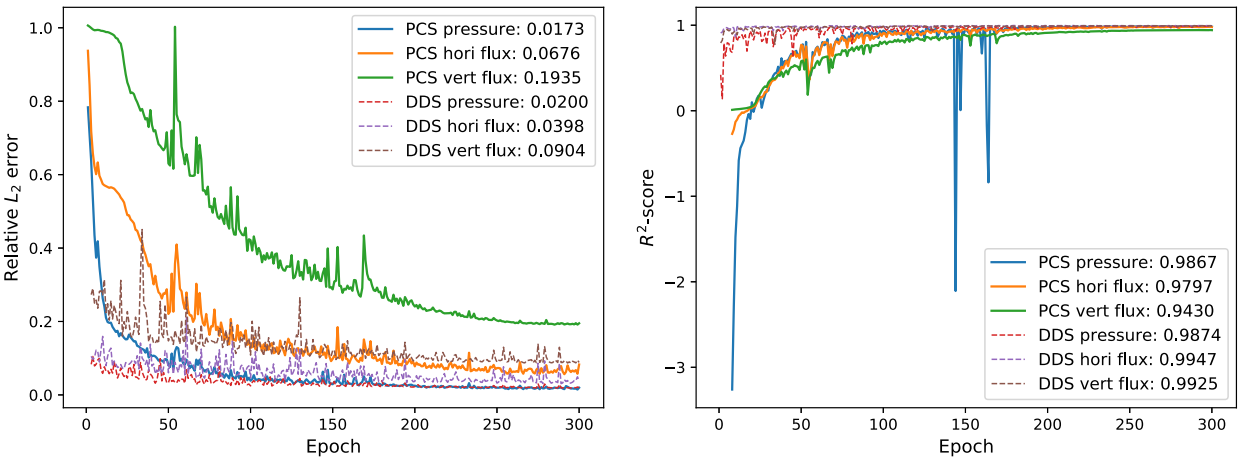
Fig. 7. Dense convolutional encoder-decoder network as the deterministic surrogate. The model's input is the realization of a random field, the model's output is the prediction for each input field including 3 output fields, i.e. pressure and two flux fields. The model is trained with physics-constrained loss without target data.

where $\hat{\mathbf{y}}_j^{(i)}$ is the surrogate prediction of the j-th output channel/field ($j = 1, 2, 3$ for pressure, horizontal flux and vertical flux field respectively), $\mathbf{y}_j^{(i)}$ is the corresponding simulator output, $\bar{\mathbf{y}}_j = \frac{1}{T}\sum_{i=1}^{T}\mathbf{y}_j^{(i)}$, T is the total number of test inputs, $\|\cdot\|_2$ is the L_2 norm. We mainly use relative l_2 error as evaluation metric. The PCS is trained for 300 epochs and the DDS is trained for 200 epochs, since DDS is faster to converge than the PCS in general, as shown in Fig. 8.

Prediction. To show that the physics-constrained approach to learn surrogate works well, we train the PCS on two datasets, i.e. GRF KLE512 (8192 samples) and channelized fields (4096 samples), respectively. The prediction examples of the PCS for test GRFs and channelized fields are shown in Fig. 9.

We show the test relative L_2 error and R^2 score during training in Fig. 8. Overall the PCS takes longer to converge than the DDS, which is reasonable since the PCS has to solve the PDE and learn the surrogate mapping at the same time. Compared with the DDS, the accuracy of the PCS' predictions of the pressure field are similar when trained with the same number of data, but the PCS' predictions of the flux fields are worse. For the latter case, the evaluation metric is dominated by the error on the boundary which is induced by the approximation of spatial derivatives. However, the predictions within the boundary are accurate, as shown in Fig. 9. Also the relative L_2 error is more sensitive than R^2 when the error is small, which can be seen by comparing Figs. 8a and 8b.

Remark 5. The quantitative results are mainly for the pressure field, not the flux fields even through we use the mixed formulation loss to train the model. Using the loss functions in Eqs. (8) and (9), we observe that the DDS focuses more on the flux fields than the pressure field, but the PCS has better predictability on the pressure field, which is often desirable. For the PCS trained with the mixed formulation, we can either output the pressure and flux fields directly, or re-compute the flux field with the predicted pressure field using the constitutive equation. The other reason for using the mixed residual loss over the primal variational loss is the better predictive accuracy of the pressure field.



(a) Test relative L_2 error. (b) Test R^2 score.

Fig. 8. Test relative L_2 error and R^2 score during training. The solid lines shows the error for the PCSs and the dashed lines for the DDSs. Both surrogates are trained on 8192 samples of GRF KLE512 and tested on the same 512 samples of GRF KLE512.

Varying the number of training inputs. We train the PCS with different number of samples from GRF KLE512, and compare its predictive performance against the DDS in Fig. 10. From the figure, the relative L_2 error decreases as the PCS is trained with more input data. While this is not surprising, it shows the convergence behavior of physics-constraint learning approach. Moreover, the PCS achieves similar relative L_2 error of predicted pressure field with the DDS when there are enough training input samples, and even lower when the number of training input samples is 8192.

The common requirement for data-driven modeling of physical systems is data efficiency, since we need *expensive* simulated output data to supervise the training. Taking [9] for an example, the number of training data is often less than 1024. The comparison here is not really appropriate. The DDS does not require physics while the PCS does not require output data. Overall, Fig. 10 suggests that with physical knowledge, we can achieve *comparable* predictive performance with the state-of-the-art DDS *without* any simulation output (but only samples from the random input).

Generalization. Apart from computational time, the PCS can 'generalize' to *any* input by directly solving the governing equations, i.e. minimizing the loss function in Eq. (8) over this particular input, as shown to work properly in Section 4.1. Thus generalization here evaluates how accurate the model's prediction is when we need to predict fast, e.g. pass the input through the surrogate, or fine-tuning the surrogate for few steps.

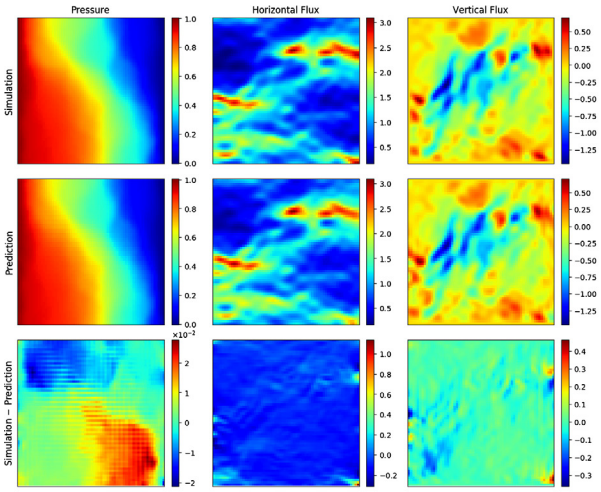
Fig. 10 shows the surrogates' interpolation performance for the test input from the same distribution as the training input, i.e. GRF KLE512. Here, we further examine the surrogates' *extrapolation* to out-of-distribution input. We select two other GRFs with different KLE terms, in particular we take KLE128 which is smoother than KLE512 and KLE2048 that leads to higher-variability than KLE512. The third test input is warped GRF which is two layers of Gaussian processes. The fourth test input is the channelized field. The samples from those test distributions are shown in Fig. 2.

We take the surrogates trained on GRF KLE512 as in the previous experiment, and test them on the four new input distributions. The relative L_2 error of predicted pressure field is shown in Fig. 11 for the surrogates trained with 8192 data. The figure shows both PCSs and DDSs generalize well to other test GRF input, including the warped one, but less so when it comes to the channelized field, which is completely different from the training input. Notably, the PCS has better generalization than the DDS when tested on warped GRF and channelized fields, which are further away from the training input distribution than the other two GRFs. This is highlighted in Fig. 12a. This holds as well for surrogates trained on $512, 1024, 2048, 4096$ samples. Fig. 12b shows the generalization performance when the training sample size is 4096.

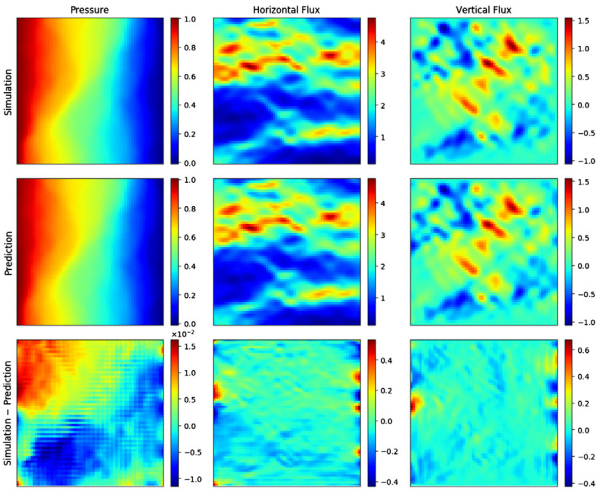
4.3. Probabilistic surrogate

This section presents the experiments on using the conditional Glow model shown in Fig. 1 as the probabilistic surrogate. We are interested in how the conditional Glow captures predictive uncertainty, uncertainty calibration and its generalization performance to unseen test input. We choose to work on 32×32 discretization instead of 64×64 with the input GRF KLE100 because of the large model size of the current implementation of the model.

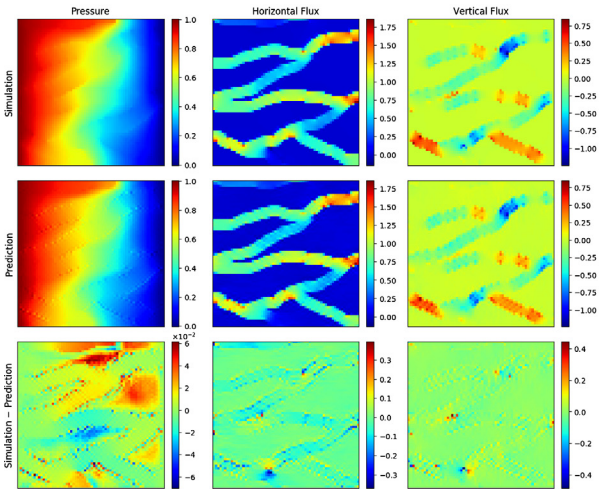
Network. In our experiment, we use $L = 3$ levels, each of which contains $F = 6$ steps of flow. Both the dense blocks and coupling networks $\boxed{\mathbf{s}}$ and $\boxed{\mathbf{t}}$ in affine coupling layers use DenseNet [79] as the building block. The number of dense layers within each dense block in the encoder is $3, 4, 4$ (from the input to the latent direction). The coupling networks `CouplingNN` as in Table C.1 for scaling and shift have 3 dense layers, followed by a 3×3 convolution layer with zero initialization to reduce the number of output features to be the same as its input features. The model has $1,535,549$



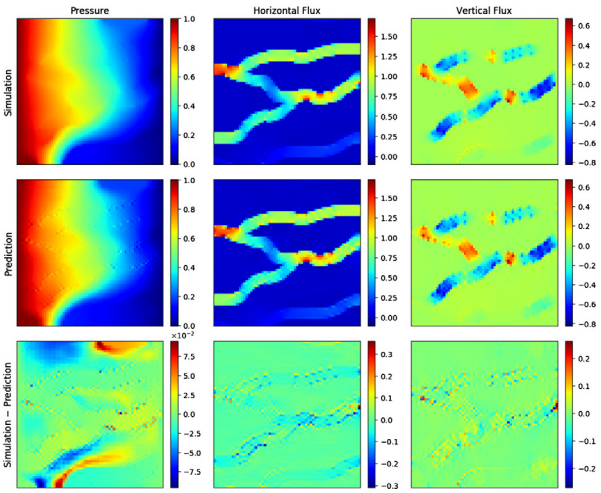
(a) GRF KLE512, test 1.



(b) GRF KLE512, test 2.



(c) Channelized, test 1.



(d) Channelized, test 2.

Fig. 9. Prediction examples of the PCS under the mixed residual loss. (a) and (b) are 2 test results for the PCS trained with 8192 samples of GRF KLE512; (c) and (d) are 2 test results for the PCS trained with 4096 samples of channelized fields.

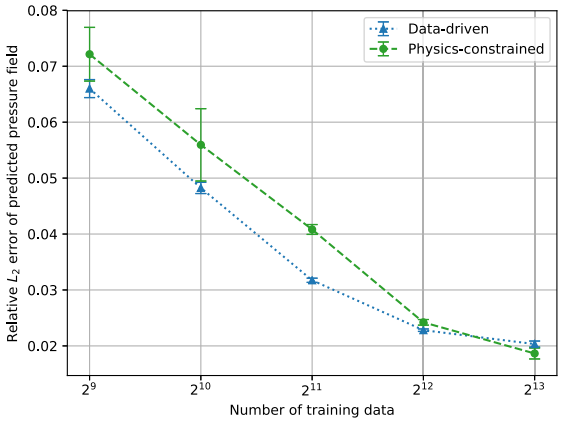


Fig. 10. The relative L_2 error of the predicted pressure field of physics-constrained and data-driven surrogates trained with 512, 1024, 2048, 4096, 8192 GRF KLE512 data, each with 5 runs to obtain the error bars. The test set contains 512 samples from GRF KLE512 as well. We emphasize that training the DDS requires an equal number of output data i.e. solutions of the governing PDE. The reference to compute relative L_2 error is simulated with FEniCS.

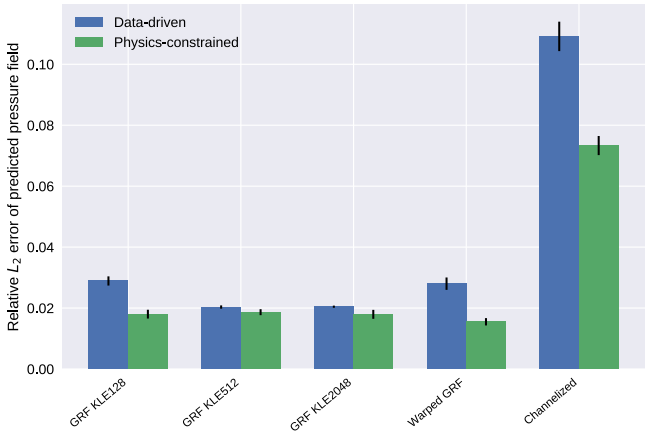


Fig. 11. Generalization to new input distributions, which are GRF KLE128, KLE512 (interpolation), KLE2048, warped GRF, and channelized fields. The surrogates are trained with 8192 samples from GRF KLE512. Each test set contains 512 samples.

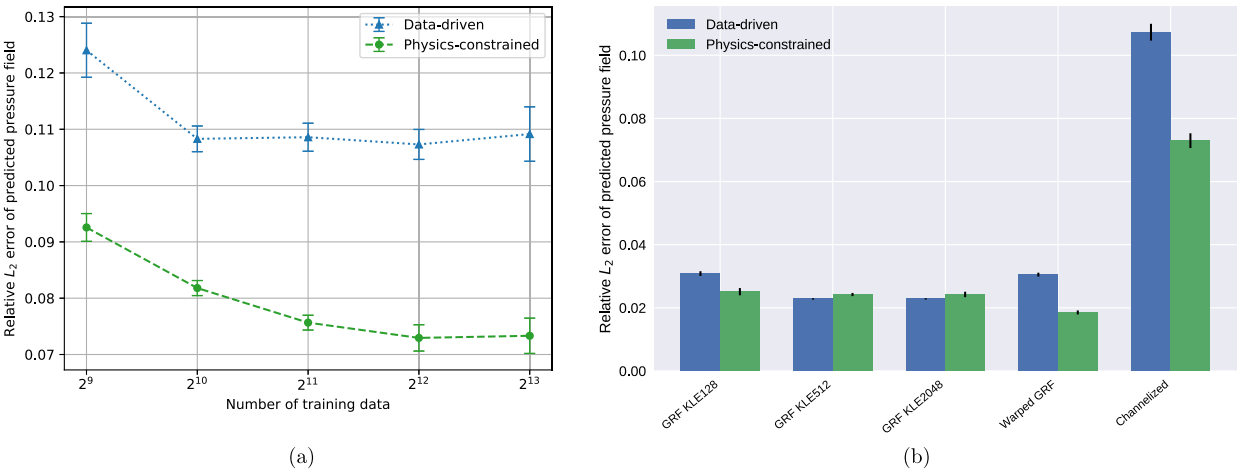


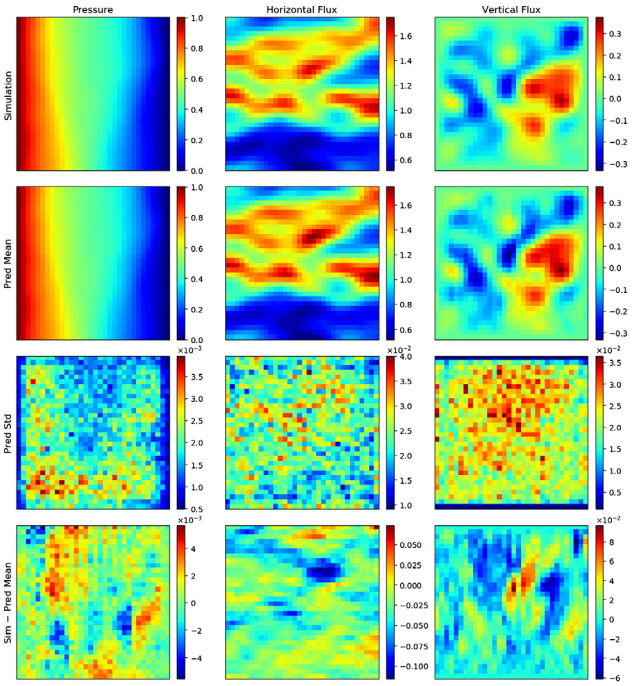
Fig. 12. (a) The relative L_2 error of predicted pressure field with PCSs and DDSs trained with $512, 1024, 2048, 4096, 8192$ GRF KLE512 data (the same surrogates as Fig. 10), each with 5 runs. The test set contains 512 samples from channelized field, with completely different distribution from the training GRF. (b) Generalization across new test input distributions for surrogates trained with 4096 samples from GRF KLE512.

parameters, including 179 convolution layers. For other hyperparameters of the model, please refer to our open-source code.

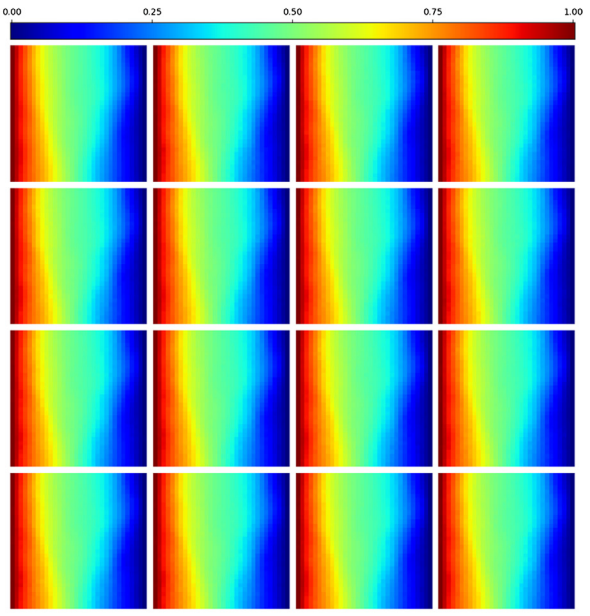
Training. The model is trained with 4096 input samples from GRF KLE100 over 32×32 grid for 400 epochs with mini-batch size 32. No output data is needed for training. We use the Adam optimizer with initial learning rate 0.0015, and one-cycle learning rate scheduler. The weight for boundary conditions λ is 50. The inverse temperature β is prefixed to certain values. Training the model with the above setting on a single NVIDIA GeForce GTX 1080 Ti GPU card takes about 3 hours.

Predictive distribution. Fig. 13 shows the prediction for a test input from GRF KLE100, where in Fig. 13a the predictive mean and variance are estimated pixel-wise with 20 samples from the conditional density by sampling 20 realizations of noise $\{\boldsymbol{\epsilon}_l^{(i)}\}_{l=1,i=1}^{L,20}$ as in Algorithm 1. The test relative L_2 error for the pressure field (comparing predictive mean against simulated output) achieves 0.0038, which is comparable to the relative L_2 error of the deterministic surrogate (0.0035). The predictive variance of the pressure and vertical flux fields reflect correctly the boundary conditions, which are close to zero on the left-right boundaries and top-bottom boundaries, respectively. We also draw 15 samples from the predictive distribution for each output field, which are shown in Figs. 13b, 13c, 13d. The predictive output samples are still diverse despite the predictive mean being highly accurate. Mode collapse is a well-known problem for conditional GANs [80,81] and VAEs [82], which seems not much of a concern for flow-based generative models as demonstrated with the diversity of samples.

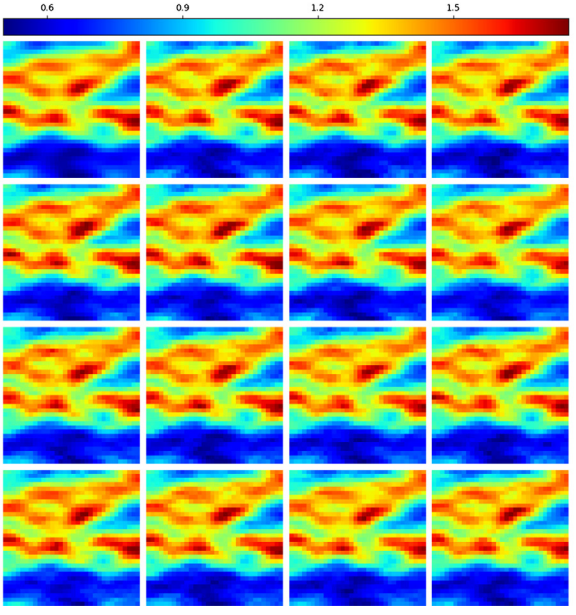
Uncertainty propagation. We use the trained conditional Glow as a surrogate to quickly predict the output for $10,000$ input samples from GRF KLE100, then compute the mean and variance of the estimated output mean, and output variance, then compare against the Monte Carlo estimate using the corresponding $10,000$ simulated output. We generate 20 samples for



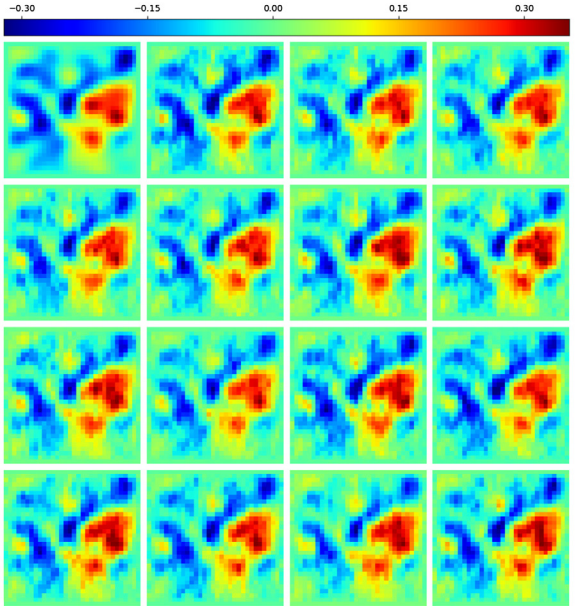
(a) Predictive mean and variance.



(b) Samples of predicted pressure.

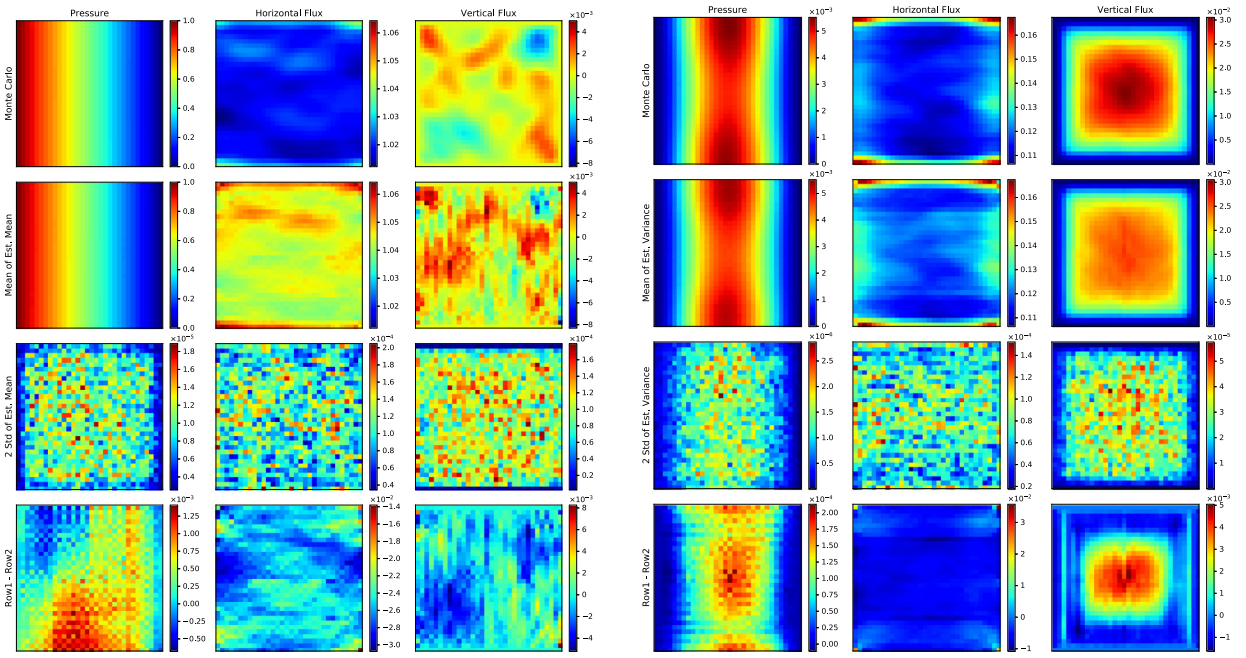


(c) Samples of predicted horizontal flux.



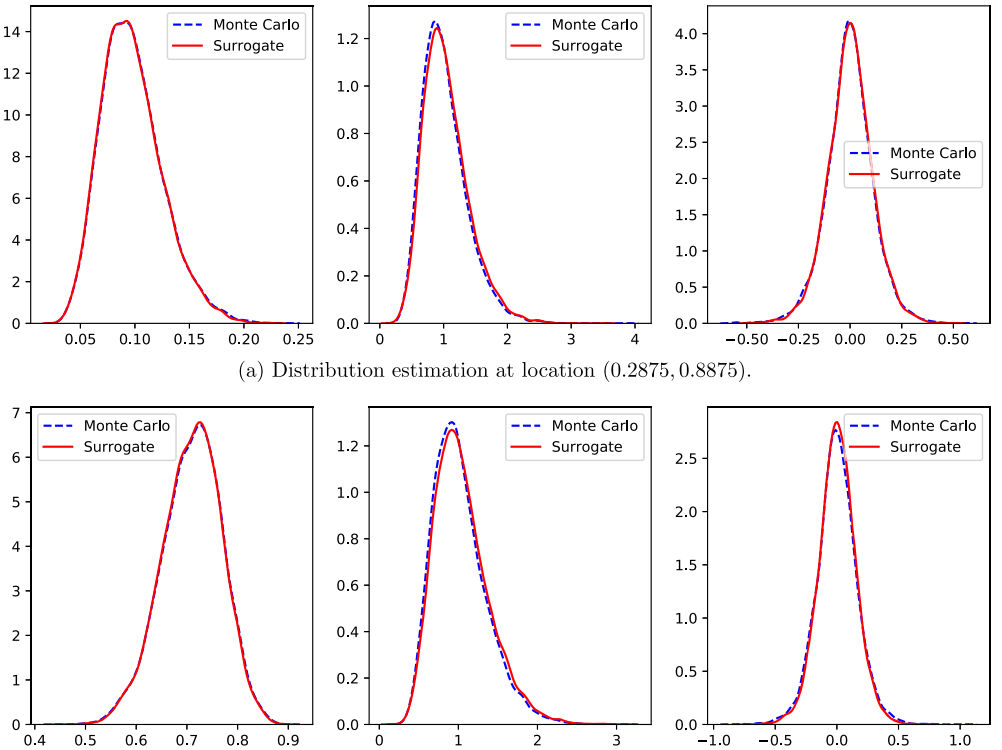
(d) Samples of predicted vertical flux.

Fig. 13. Prediction of the multiscale conditional Glow ($\beta = 150$) for a test input which is sampled from GRF KLE100 over 32×32 grid. (a) The predictive mean (2nd row) and one standard derivation (3rd row) are obtained with 20 output samples. The first row show three simulated output fields, the 4-th row shows the error between the reference and the predictive mean. In (b), (c), (d), the top left corner shows the simulated output, the rest 15 are samples from the conditional predictive density $p_\theta(\mathbf{y}|\mathbf{x})$. The relative L_2 error for the predicted pressure field is 0.0038 when tested on 512 samples from GRF KLE100.



(a) Estimate of output mean. (b) Estimate of output variance.

Fig. 14. Uncertainty propagation with multiscale conditional Glow, $\beta = 150$. (a) The first row shows the sample mean of 10,000 simulated output, the second and third rows show the sample mean and two standard deviation of the estimate mean of 10,000 predicted output with the probabilistic surrogate, and the fourth row shows the error between the first two rows. (b) The results for output variance.



(a) Distribution estimation at location $(0.2875, 0.8875)$.

(b) Distribution estimation at location $(0.4625, 0.2875)$.

Fig. 15. Distribution estimate with conditional Glow, $\beta = 150$. From left to right shows the density estimate of pressure, horizontal flux and vertical flux at certain locations of the domain $[0, 1]^2$.

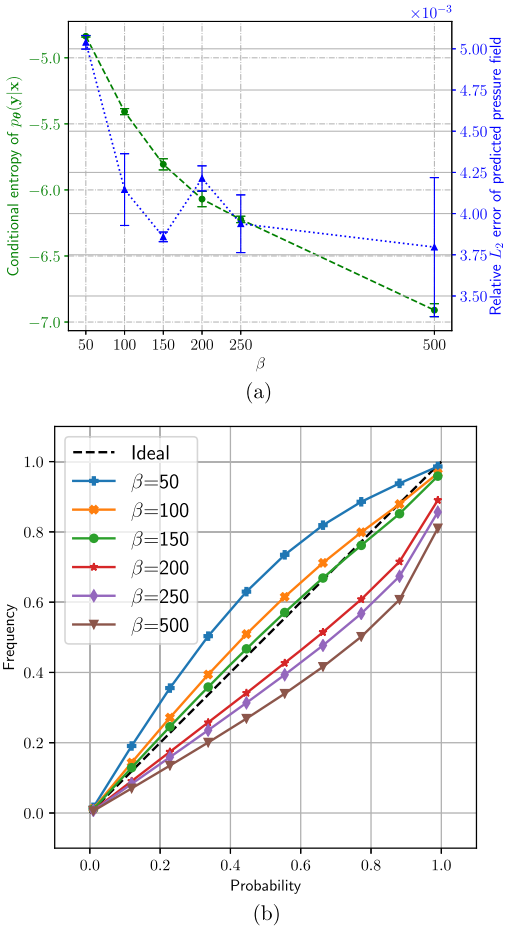


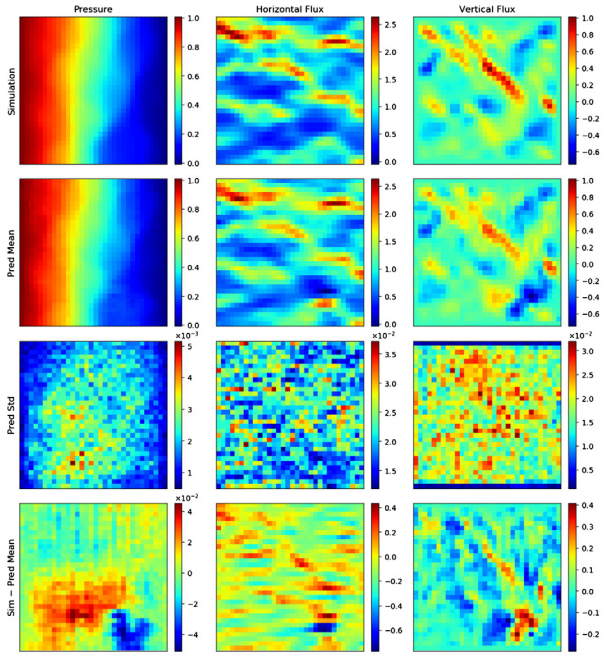
Fig. 16. (a) Conditional entropy of $p_\theta(\mathbf{y}|\mathbf{x})$ and relative L_2 error of predicted pressure field w.r.t. β. Conditional entropy is evaluated in bits per dimension. The surrogate is tested on 512 input samples from GRF KLE100. The error bar is obtained with 3 independent runs. (b) Reliability diagram of predicted pressure field with conditional Glow trained with different β, which is evaluated with 10,000 input-output data pairs. The closer the diagram is to the diagonal, the better the probabilistic surrogate is calibrated.

each input with the trained surrogate, then estimate the mean and variance of the output with the law of total expectation and the law of total variance. By repeating this process for 10 times, we obtain 10 estimates of the mean and variance for the output. Then the sample mean and variance of the 10 estimate means and estimate variances can be computed, which are shown in the second and third row of Fig. 14. The statistics of the surrogate output matches that of the simulation output very well, especially for the output variance which are typically underestimated when using surrogates. Note that there is only small error (around 3% relative error) between the estimated mean of the horizontal flux field despite the noticeable difference in color as in Fig. 14a.

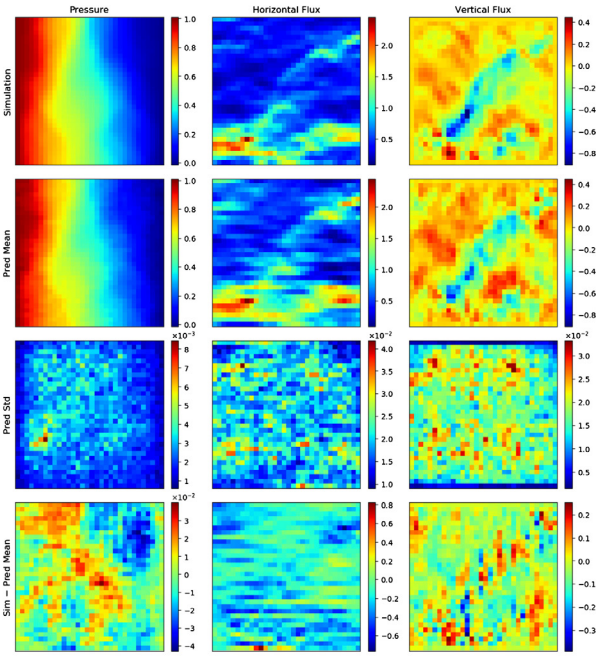
Distribution estimate. We show in Fig. 15 the kernel density estimation for the values of three output fields at random locations in the domain using the 10,000 output samples from simulation and the ones propagated with the trained conditional Glow.

Uncertainty calibration by tuning β. Given the PDEs and boundary conditions, the prediction of the surrogate can be evaluated directly with the loss $L(\mathbf{y}, \mathbf{x})$, without requiring the reference solution (e.g. simulation output). However, this loss cannot be readily translated to the uncertainty of the solution, e.g. the upper and lower bound of the solution at every grid point in the domain. The probabilistic surrogate trained under the reverse KL divergence can provide the uncertainty estimate, but may be at the expense of the accuracy of the mean prediction. The precision parameter β controls the overall variance of the reference density, which is reflected from the conditional entropy of the model density $p_\theta(\mathbf{y}|\mathbf{x})$ in Fig. 16a. The influence of β on the accuracy and the entropy of the model can be seen from the two competing terms in the reverse KL divergence as well.

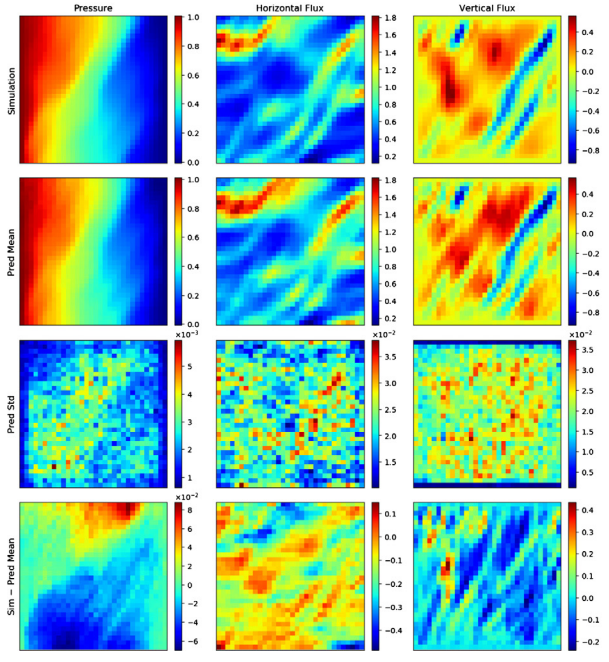
The reliability diagram shown in Fig. 16b is used as an uncertainty calibration tool to measure the discrepancy between the model forecasts and the (empirical) long-run frequencies [19]. More concretely, we first compute the p% prediction interval (`probability` in the horizontal axis of the plot) for each test input based on Gaussian quantiles using the pre-



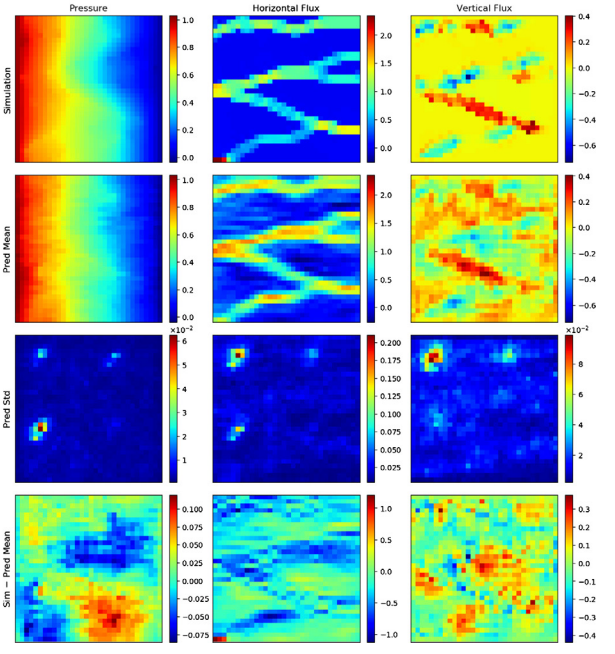
(a) GRF KLE256.



(b) GRF KLE512.



(c) Warped GRF.

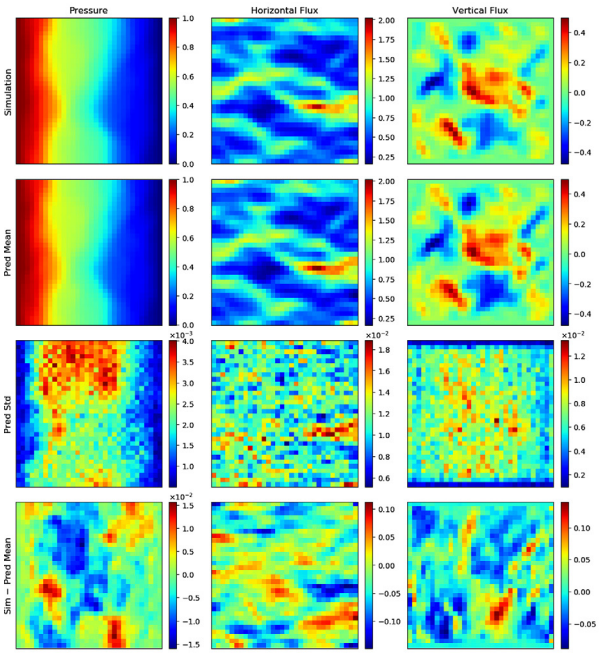


(d) Channelized field.

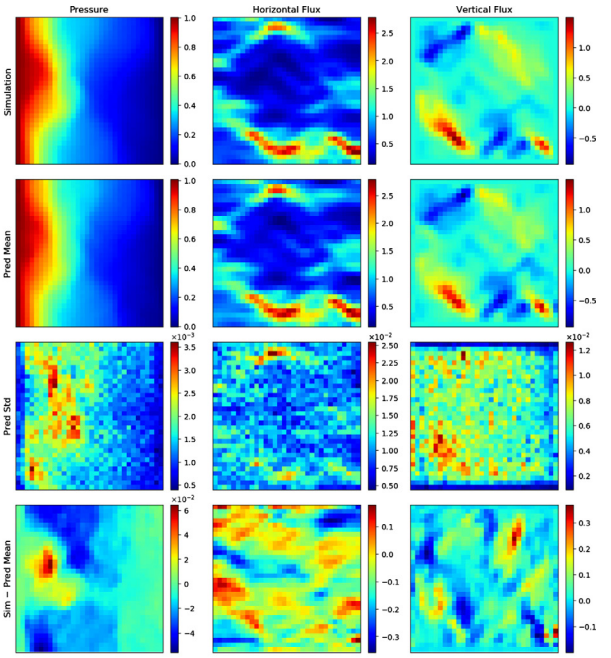
Fig. 17. Generalization of conditional Glow to out-of-distribution input. The model is trained on GRF KLE100, and tested on (a) GRF KLE256, (b) GRF KLE512, (c) Warped GRF, (d) channelized field. Note that the results are cherry-picked.

dictive mean and variance of the model at this test point. We next measure what fraction of the test output/observations falls within the p% prediction interval, shown as `frequency` in the vertical axis of the plot. We compute the frequency for $p = 10, 20, \cdots, 90$. A well-calibrated model should have a diagram that is close to the diagonal.

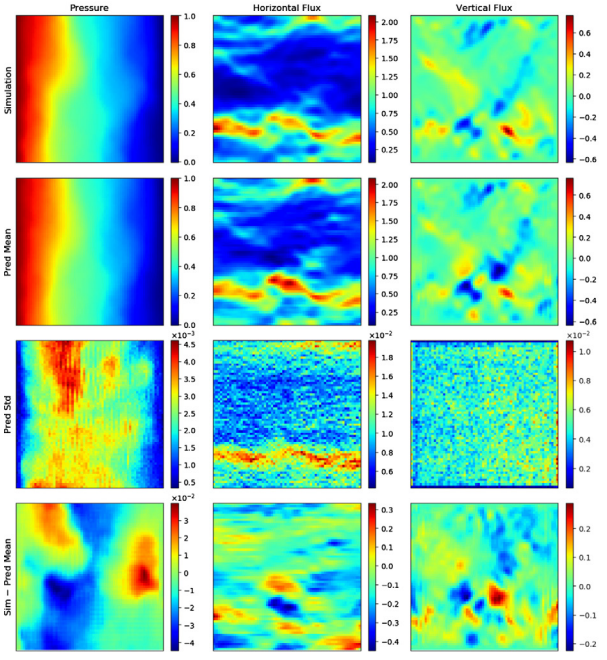
Larger β puts more penalty of the PDE loss term $L(\mathbf{y}, \mathbf{x})$ and less on the negative conditional entropy, thus the predictions become more accurate but less diverse, and to some extent, the probabilistic surrogate becomes over confident as shown in Fig. 16b when $\beta = 500$. On the other hand, when β is too small, the probabilistic surrogate is prudent (large uncertainty



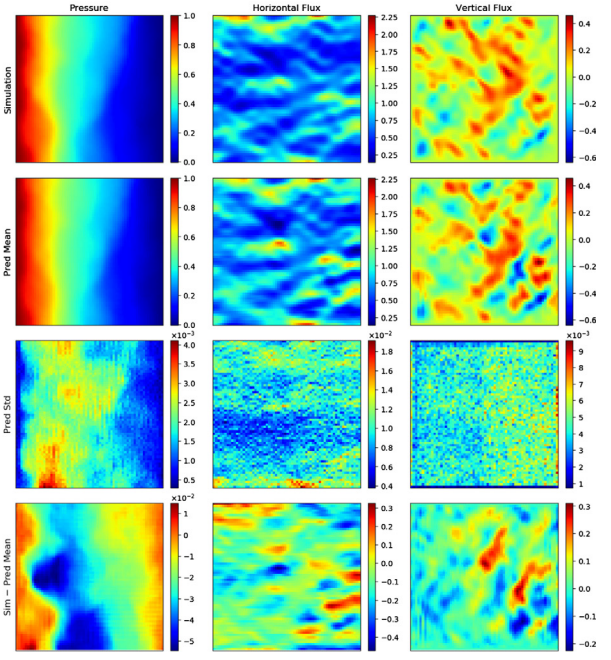
(a) GRF KLE256 over 32×32, test realization 1.



(b) GRF KLE256 over 32×32, test realization 2.



(c) GRF KLE512 over 64×64, test realization 1.



(d) GRF KLE512 over 64×64, test realization 2.

Fig. 18. Predictions of the multiscale conditional Glow model with higher input dimension. (a) and (b) are predictions for two test inputs sampled from GRF KLE256 over 32×32 grid with $\beta = 200$, (c) and (d) are predictions for two test inputs sampled from GRF KLE512 over 64×64 grid with $\beta = 150$. The predictive mean (2nd row) and one standard deviation (3rd row) are obtained with 20 output samples. The first row shows three simulated output fields, and the fourth row shows the error between the simulation and predictive mean. The relative L_2 error for the predicted pressure field is 0.019875, evaluated on 512 test samples from 32×32 GRF KLE256, and 0.0346, evaluated on 512 test samples from 64×64 GRF KLE512.

estimate) and less accurate about the solution, e.g. the case of $\beta = 50$. From the figure, the model trained under $\beta = 150$ is well-calibrated (its reliability diagram is close to the diagonal dashed line) and achieves high accuracy at the same time.

Generalization. We test the generalization of conditional Glow on input distributions different from the training input (GRF KLE100), including GRF KLE256, GRF KLE512, warped GRF, and channelized fields, as in Fig. 17. However, we could not observe larger uncertainty when the test input is far away from the training input. The error between the predictive mean and simulation is in general one magnitude larger than the uncertainty. Thus the current surrogate cannot express what it does not know which in practice is a highly desirable outcome.

Higher dimension. One well-known limitation of the Glow model is its scalability to larger spatial dimension (as well as intrinsic dimension) because of the restriction of its model structure. For completeness, we trained two conditional Glow models for input permeability with higher dimension, i.e. GRF KLE256 over a 32×32 grid and GRF KLE512 over a 64×64 grid. The prediction results for test inputs are shown in Fig. 18.

5. Conclusions

This paper has offered a foray in physics-aware machine learning for surrogate modeling and uncertainty quantification, with emphasis on the solution of PDEs. The most significant contribution of the proposed framework and simultaneously the biggest difference with other efforts along these lines, is that *no labeled data* are needed i.e. one does not need to solve governing PDEs for the training inputs. This is accomplished by incorporating appropriately the governing equations into the loss/likelihood functions. We have demonstrated that convolutional encoder-decoder network-based surrogate models can achieve high predictive accuracy for high-dimensional stochastic input fields. Furthermore, the generalization performance of the physics-constrained surrogates proposed is consistently better than data-driven surrogates for out-of-distribution test inputs. The probabilistic surrogate built on the flow-based conditional generative model and trained by employing the reverse KL-divergence loss, is able to capture predictive uncertainty as demonstrated in several uncertainty propagation and calibration tasks.

Many important unresolved tasks have been identified that will be addressed in forthcoming works. They include (a) Extension of this work to surrogate modeling for dynamical systems, (b) Improving generalization on out-of-distribution input, e.g. fine-tuning the trained surrogate on test input [83,84], learned gradient update [85,86], meta-learning on a distribution of regression tasks [87], etc., (c) Combining physics-aware and data-driven approaches when only limited simulation data and partially known physics are available [88], (d) Scale the flow-based conditional generative models to higher dimensions [89], (e) More reliable probabilistic models, e.g. being able to express what the model does not know [90,91] by showing larger predictive uncertainty when tested on out-of-distribution input, (f) Exploring ways to increase the expressiveness of FC-NNs to better capture the multiscale features of PDE solutions, e.g. by evolving network architectures [92] and (g) Exploring the solution landscape with the conditional generative surrogates [50].

Acknowledgements

The authors acknowledge support from the Defense Advanced Research Projects Agency (DARPA) under the Physics of Artificial Intelligence (PAI) program (contract HR00111890034). Additional computing resources were provided by the University of Notre Dame's Center for Research Computing (CRC) and by the AFOSR Office of Scientific Research through the DURIP program. The authors would also like to acknowledge many constructive comments from the readers of the first version of the manuscript posted on arXiv that have allowed us to significantly improve the presentation of the methodologies and results.

Appendix A. Sobel filter to estimate spatial gradients

Sobel filter is used to estimate horizontal and vertical spatial gradients by applying one convolution with the following 3×3 kernels, respectively:

$$\mathcal{H} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad \mathcal{V} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

Intuitively it is a smoothed finite difference method. The convolution operation goes together naturally with CNN representation of solution fields and is highly efficient. Sobel filter is more efficient than using automatic differentiation to obtain spatial gradients in the FC-NN parameterization, with the compromise of reduced accuracy, especially for locations close to the boundaries.

To improve the accuracy of gradient estimate on the boundary, we use the following correction. For 2D image matrix $\mathbf{I}$ of size $H \times W$, Sobel kernel $\mathcal{H}$, and correction matrix $M_{\mathcal{H}}$ of size $W \times W$,

$$M_{\mathcal{H}} = \begin{bmatrix} 4 & 0 & 0 & & & 0 \\ -1 & 1 & 0 & & & \\ 0 & 0 & 1 & & & \\ & & & \cdots & 0 & 0 \\ & & & & 1 & -1 \\ 0 & & & 0 & 0 & 4 \end{bmatrix},$$

the horizontal gradient is estimated as $(\mathbf{I} \star \mathcal{H}) M_{\mathcal{H}}$, where $\star$ is convolution with replicate padding on the boundary. This is effectively using forward finite differences on the left boundary and backward finite differences on the right boundary. The vertical gradient estimate is corrected similarly. We found that this correction reduces the error of the learned solution by several times. However, there are still errors in four corners, which can be further improved by more refined correction.

Appendix B. Solving PDEs with FC-NNs and CNNs

B.1. Loss functions

The loss functions for optimizing DNNs to solve PDEs are illustrated here for the model problem in Section 4. For FC-NNs, the solution is paramterized as $[\hat{\tau}, \hat{u}] = \hat{y}_{\boldsymbol{\phi}}(s)$, where $\hat{\tau} = [\hat{\tau}_1, \hat{\tau}_2]$. The mixed residual loss is taken as the following

$$\begin{aligned} L(\boldsymbol{\phi}; \mathbf{x}) \approx & \frac{1}{S} \sum_{i=1}^{S} \Big[(\hat{\tau}(s_i) + K(s_i) \nabla \hat{u}(s_i))^2 + (\nabla \hat{\tau}(s_i) - f(s_i))^2 \Big] \\ & + \lambda \Big[\frac{1}{S_D} \sum_{j=1}^{S_D} (\hat{u}(s_j) - u_D(s_j))^2 + \frac{1}{S_N} \sum_{k=1}^{S_N} (\hat{\tau}(s_k) + g(s_k))^2 \Big], \end{aligned}$$

where $\{s_i\}_{s=1}^{S}, \{s_j\}_{s=1}^{S_D}, \{s_k\}_{k=1}^{S_N}$ are the collocation points for the PDE constraints in the domain $\mathcal{S}$, for the Dirichlet boundary condition on Γ_D, and for the Neumann boundary condition on Γ_N. Here, the gradients $\nabla \hat{u}(s_i), \nabla \hat{\tau}(s_i)$ are computed with automatic differentiation.

For the CNNs, the solution is parameterized as a convolutional decoder $[\hat{\boldsymbol{\tau}}, \hat{\mathbf{y}}] = \hat{\mathbf{y}}_{\boldsymbol{\theta}}(\mathbf{z})$, where $\hat{\boldsymbol{\tau}} = [\hat{\boldsymbol{\tau}}_1, \hat{\boldsymbol{\tau}}_2]$. The mixed residual loss is given as follows:

$$L(\boldsymbol{\theta}; \mathbf{x}) \approx \frac{1}{n_s} \Big(\big\| \hat{\boldsymbol{\tau}} + \mathbf{x} \odot \nabla \hat{\mathbf{u}} \big\|_2^2 + \big\| \nabla \cdot \hat{\boldsymbol{\tau}} - \mathbf{f} \big\|_2^2 \Big) + \lambda \Big(\big\| \hat{\mathbf{u}}[:, 0] - \mathbf{1} \big\|_2^2 + \big\| \hat{\mathbf{u}}[:, -1] \big\|_2^2 + \big\| \hat{\boldsymbol{\tau}}_2[[0, -1], :] \big\|_2^2 \Big),$$

where n_s is the number of uniform grid points, $\nabla \mathbf{u} = [\mathbf{u}_h, \mathbf{u}_v]$, $\mathbf{u}_h, \mathbf{u}_v$ are two gradient images along the horizontal and vertical directions estimated by Sobel filter, similarly for $\nabla \cdot \boldsymbol{\tau} = (\boldsymbol{\tau}_1)_h + (\boldsymbol{\tau}_2)_v$, and $\odot$ denotes the element-wise product. $\mathbf{z}$ is the latent variable that is kept fixed after arbitrary initialization. While the effects of the initial values of $\mathbf{z}$ and initialization of the decoder parameters were not systematically investigated, our experience shows that different initializations lead to similar but not exactly the same results.

B.2. Network architecture

The FC-NN used in the experiments in Section 4.1 has 8 hidden layers and 512 nodes per hidden layer, with the input and output dimensions being 2 and 3, respectively. The nonlinear activation is `Tanh`. The total number of parameters is $1,841,155$. We increased the number of nodes in the hidden layer from 20 to 512 to overfit the solution. We considered the collocation points to be at random locations in the domain, and increased their number. However, none of these modifications lead to improvement of the learned solution.

The convolutional decoder network uses two dense blocks with 8 and 6 dense layers respectively to transform the latent $\mathbf{z}$ of size $1 \times 16 \times 16$ to the output $\mathbf{y}$ of size $3 \times 64 \times 64$. The decoding layers use nearest upsampling followed by one 3×3 convolution. The network has $514,278$ parameters and 20 convolution layers.

We train FC-NNs and CNNs with mixed residual loss using L-BFGS optimizer (with history size 50 and maximum iteration 20), learning rate 0.5. The weight for boundary loss is $\lambda = 10$.

Appendix C. Conditional Glow network structure

In Fig. 1a, the encoder network includes a cascade of L `Dense Blocks` (that maintain the feature map size) and $L-1$ `Trans Down` layers (that typically half the feature map size, e.g. from 32×32 to 16×16). The features extracted after each dense block are treated as input features $\{\boldsymbol{\xi}_l\}_{l=1}^{L}$. For details of `Dense Blocks` and `Trans Down` layers (encoding layer), please refer to Section 2.2 in [9]. The encoder network has only a forward pass, i.e. from the condition $\mathbf{x}$ to its multiscale features $\{\boldsymbol{\xi}_l\}_{l=1}^{L}$. The parameters in the encoder are jointly optimized with those of the Glow model.

Table C.1
Forward (from $\mathbf{y}'$ to $\mathbf{z}'$) and reverse paths of affine coupling layer with condition of input features $\boldsymbol{\xi}_l$ as in Fig. 1c.

Forward	Reverse
$\mathbf{y}'_1, \mathbf{y}'_2 = \texttt{split}(\mathbf{y}')$	$\mathbf{z}'_1, \mathbf{z}'_2 = \texttt{split}(\mathbf{z}')$
$\hat{\mathbf{y}}_1 = \texttt{concat}(\mathbf{y}'_1, \boldsymbol{\xi}_l)$	$\hat{\mathbf{z}}_1 = \texttt{concat}(\mathbf{z}'_1, \boldsymbol{\xi}_l)$
$(\hat{\mathbf{s}}, \mathbf{t}) = \texttt{CouplingNN}(\hat{\mathbf{y}}_1)$	$(\hat{\mathbf{s}}, \mathbf{t}) = \texttt{CouplingNN}(\hat{\mathbf{z}}_1)$
$\mathbf{s} = \texttt{sigmoid}(\hat{\mathbf{s}} + 2)$	$\mathbf{s} = \texttt{sigmoid}(\hat{\mathbf{s}} + 2)$
$\mathbf{z}'_2 = \mathbf{s} \odot \mathbf{y}'_2 + \mathbf{t}$	$\mathbf{y}'_2 = (\mathbf{z}'_2 - \mathbf{t})/\mathbf{s}$
$\mathbf{z}'_1 = \mathbf{y}'_1$	$\mathbf{y}'_1 = \mathbf{z}'_1$
$\mathbf{z}' = \texttt{concat}(\mathbf{z}'_1, \mathbf{z}'_2)$	$\mathbf{y}' = \texttt{concat}(\mathbf{y}'_1, \mathbf{y}'_2)$

In Fig. 1a, the `Squeeze` operator rearranges the features of size $C \times H \times W$ into $4C \times \frac{1}{2}H \times \frac{1}{2}W$ if the squeeze factor is 2, where C, H, W denote the number of channels, height, and width of the feature maps.

The `Split` operator splits the feature maps into two parts $[\mathbf{z}_l, \mathbf{h}_l]$. Here, we model half of the features/channels $\mathbf{z}_l$ as latent variable, which is diagonal Gaussian parameterized by the other half features $\mathbf{h}_l$, i.e. $\mathbf{z}_l \sim \mathcal{N}(\mathbf{z}_l | \boldsymbol{\mu}^l_{\boldsymbol{\theta}}(\mathbf{h}_l), (\boldsymbol{\sigma}^l_{\boldsymbol{\theta}}(\mathbf{h}_l))^2)$, where $\boldsymbol{\mu}^l_{\boldsymbol{\theta}}, \log \boldsymbol{\sigma}^l_{\boldsymbol{\theta}}$ are parameterized with a 3×3 convolution, with stride 1, padding 0, and zero initialization. For the reverse path, given $\mathbf{h}_l$, we first sample the latent variables $\mathbf{z}_l = \boldsymbol{\mu}^l_{\boldsymbol{\theta}} + \boldsymbol{\sigma}^l_{\boldsymbol{\theta}} \odot \boldsymbol{\epsilon}_l, \boldsymbol{\epsilon}_l \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$), then concatenate the two parts $[\mathbf{z}_l, \mathbf{h}_l]$ (i.e. the reverse of `Split`) as the output.

In Fig. 1b, one step of flow contains an activation normalization layer (`ActNorm`), an invertable 1×1 convolution layer and an affine coupling layer. Assume the input $\mathbf{x}$ and the output $\mathbf{y}$ have C channels, with each of the channels being of size $H \times W$, where the spatial location is indexed by i, j. The `ActNorm` performs affine transformation of the activation with a scale and bias parameter per-channel. The `ActNorm` layer computes $\mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}, \forall i, j$, where $\mathbf{s}, \mathbf{b}$ are the learnable scale and bias parameters. `ActNorm` is reversible, $\mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$. The log-determinant of its Jacobian is $H \cdot W \cdot \texttt{sum}(\log|\mathbf{s}|)$.

The invertible 1×1 convolution layer is a convolution with kernel size 1, stride 1, padding 0, with the same output channels as the input. The kernel matrix $\mathbf{W}$ is of size (C, C), and the forward path is computed as $\mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}, \forall i, j$. Thus it is essentially a learnable permutation operation to mix the two parts of the flow features, before passing them to the affine coupling layer. It is also invertible, i.e. $\mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}, \forall i, j$. The log-determinant of its Jacobian is $H \cdot W \cdot \log|\det(\mathbf{W})|$.

We show the detailed computation of the forward and reverse paths of the affine coupling layer (Fig. 1c) in Table C.1. The nonlinear transform `CouplingNN` includes 3 dense layers, followed by a 3×3 convolution layer with zero initialization, whose output channels split into two parts, i.e. $(\hat{\mathbf{s}}, \mathbf{t})$. The log-determinant of its Jacobian is $\texttt{sum}(\log(|\mathbf{s}|))$.

References

[1] I. Bilionis, N. Zabaras, B.A. Konomi, G. Lin, Multi-output separable Gaussian process: towards an efficient, fully Bayesian paradigm for uncertainty quantification, J. Comput. Phys. 241 (2013) 212–239, https://doi.org/10.1016/j.jcp.2013.01.011, http://www.sciencedirect.com/science/article/pii/S0021999113000417.

[2] E. Charalampidis, P. Kevrekidis, P. Farrell, Computing stationary solutions of the two-dimensional Gross–Pitaevskii equation with deflated continuation, Commun. Nonlinear Sci. Numer. Simul. 54 (2018) 482–499.

[3] M.C. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (1) (2000) 1–13, http://www.jstor.org/stable/2673557.

[4] A. Wilson, H. Nickisch, Kernel interpolation for scalable structured gaussian processes (kiss-gp), in: International Conference on Machine Learning, 2015, pp. 1775–1784, http://proceedings.mlr.press/v37/wilson15.pdf.

[5] S. Atkinson, N. Zabaras, Structured bayesian gaussian process latent variable model: applications to data-driven dimensionality reduction and high-dimensional inversion, J. Comput. Phys. 383 (2019) 166–195, https://doi.org/10.1016/j.jcp.2018.12.037, http://www.sciencedirect.com/science/article/pii/S0021999119300397.

[6] M. van der Wilk, C.E. Rasmussen, J. Hensman, Convolutional gaussian processes, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 2849–2858, http://papers.nips.cc/paper/6877-convolutional-gaussian-processes.pdf, 2017.

[7] J.R. Gardner, G. Pleiss, D. Bindel, K.Q. Weinberger, A.G. Wilson, Gpytorch: blackbox matrix-matrix gaussian process inference with gpu acceleration, in: Advances in Neural Information Processing Systems, 2018.

[8] C. Yang, X. Yang, X. Xiao, Data-driven projection method in fluid simulation, Comput. Animat. Virtual Worlds 27 (3–4) (2016) 415–424, https://doi.org/10.1002/cav.1695, https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1695.

[9] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447, https://doi.org/10.1016/j.jcp.2018.04.018, http://www.sciencedirect.com/science/article/pii/S0021999118302341.

[10] R.K. Tripathy, I. Bilionis, Deep UQ: learning deep neural network surrogate models for high dimensional uncertainty quantification, J. Comput. Phys. 375 (2018) 565–588, https://doi.org/10.1016/j.jcp.2018.08.036, http://www.sciencedirect.com/science/article/pii/S0021999118305655.

[11] S. Mo, Y. Zhu, N. Zabaras, X. Shi, J. Wu, Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media, Water Resour. Res. 55 (1) (2019) 703–728, https://doi.org/10.1029/2018WR023528.

[12] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166, https://doi.org/10.1017/jfm.2016.615.

[13] N. Thuerey, K. Weissenow, H. Mehrotra, N. Mainali, L. Prantl, X. Hu, Well, how accurate is it? A study of deep learning methods for Reynolds-averaged Navier-Stokes simulations, arXiv preprint, arXiv:1810.08217.

[14] N. Geneva, N. Zabaras, Quantifying model form uncertainty in Reynolds-averaged turbulence models with bayesian deep neural networks, J. Comput. Phys. 383 (2019) 125–147, https://doi.org/10.1016/j.jcp.2019.01.021, http://www.sciencedirect.com/science/article/pii/S0021999119300464.

[15] D.J.C. MacKay, A practical bayesian framework for backpropagation networks, Neural Comput. 4 (3) (1992) 448–472, https://doi.org/10.1162/neco.1992.4.3.448.
[16] D.P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, in: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 2575–2583, http://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick.pdf.
[17] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: representing model uncertainty in deep learning, in: International Conference on Machine Learning, 2016, pp. 1050–1059.
[18] Q. Liu, D. Wang, Stein variational gradient descent: a general purpose bayesian inference algorithm, in: D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems 29, Curran Associates, Inc., 2016, pp. 2378–2386, http://papers.nips.cc/paper/6338-stein-variational-gradient-descent-a-general-purpose-bayesian-inference-algorithm.pdf.
[19] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: Advances in Neural Information Processing Systems, 2017, pp. 6402–6413.
[20] C. Grigo, P.-S. Koutsourelakis, Bayesian model and dimension reduction for uncertainty propagation: applications in random media, arXiv:1711.02475.
[21] C.M. Jiang, J. Huang, K. Kashinath Prabhat, P. Marcus, M. Niessner, Spherical CNNs on unstructured grids, in: International Conference on Learning Representations, 2019, https://openreview.net/forum?id=Bkl-43C9FQ.
[22] A. Sanchez-Gonzalez, N. Heess, J.T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, P. Battaglia, Graph networks as learnable physics engines for inference and control, in: Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 4470–4479.
[23] B. Lusch, J.N. Kutz, S.L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nat. Commun. 9 (1) (2018) 4950.
[24] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: learning PDEs from data, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm Sweden, in: Proceedings of Machine Learning Research, vol. 80, 2018, pp. 3208–3216, http://proceedings.mlr.press/v80/long18a.html.
[25] B. Kim, V.C. Azevedo, N. Thuerey, T. Kim, M. Gross, B. Solenthaler, Deep fluids: a generative network for parameterized fluid simulations, arXiv:1806.02071.
[26] R. Stewart, S. Ermon, Label-free supervision of neural networks with physics and domain knowledge, arXiv:1609.05566.
[27] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: IEEE International Conference on Computer Vision (ICCV), 2017, 2017.
[28] Y. Xie, E. Franz, M. Chu, N. Thuerey, Tempogan: a temporally coherent, volumetric GAN for super-resolution fluid flow, arXiv preprint, arXiv:1801.09710.
[29] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, B. Catanzaro, Video-to-video synthesis, in: Advances in Neural Information Processing Systems (NeurIPS), 2018.
[30] D.C. Psichogios, L.H. Ungar, A hybrid neural network-first principles approach to process modeling, AIChE J. 38 (10) (1992) 1499–1511, https://doi.org/10.1002/aic.690381003, https://onlinelibrary.wiley.com/doi/abs/10.1002/aic.690381003.
[31] A. Meade, A. Fernandez, The numerical solution of linear ordinary differential equations by feedforward neural networks, Math. Comput. Model. 19 (12) (1994) 1–25, https://doi.org/10.1016/0895-7177(94)90095-7, http://www.sciencedirect.com/science/article/pii/0895717794900957.
[32] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000, https://doi.org/10.1109/72.712178.
[33] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, arXiv:1711.10561.
[34] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41, https://doi.org/10.1016/j.neucom.2018.06.056.
[35] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018), https://doi.org/10.1007/s40304-018-0127-z.
[36] M.A. Nabian, H. Meidani, A deep neural network surrogate for high-dimensional random partial differential equations, arXiv:1806.02957.
[37] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364, https://doi.org/10.1016/j.jcp.2018.08.029, http://www.sciencedirect.com/science/article/pii/S0021999118305527.
[38] P. Grohs, F. Hornung, A. Jentzen, P. Von Wurstemberger, A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations, arXiv:1809.02362.
[39] J. Han, A. Jentzen, W.E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510, https://doi.org/10.1073/pnas.1718942115, https://www.pnas.org/content/115/34/8505.
[40] C. Beck, W. E, A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, arXiv:1709.05963, 2017.
[41] M. Raissi, Forward-backward stochastic neural networks: deep learning of high-dimensional partial differential equations, arXiv:1804.07010.
[42] Y. Wang, S.W. Cheung, E.T. Chung, Y. Efendiev, M. Wang, Deep multiscale model learning, arXiv:1806.04830.
[43] Y. Fan, L. Lin, L. Ying, L. Zepeda-Núnez, A multiscale neural network based on hierarchical matrices, arXiv:1807.01883.
[44] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, Accelerating Eulerian fluid simulation with convolutional networks, arXiv:1607.03597.
[45] Y. Khoo, J. Lu, L. Ying, Solving PDE problems with uncertainty using neural-networks, arXiv:1707.03351.
[46] V.M. Filippov, V.M. Savchin, S.G. Shorokhov, Variational principles for nonpotential operators, J. Math. Sci. 68 (3) (1994) 275–398, https://doi.org/10.1007/BF01252319.
[47] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, https://doi.org/10.1016/j.jcp.2018.10.045, http://www.sciencedirect.com/science/article/pii/S0021999118307125.
[48] D. Ulyanov, A. Vedaldi, V. Lempitsky, Deep image prior, arXiv:1711.10925.
[49] D.N. Arnold, Mixed finite element methods for elliptic problems, Comput. Methods Appl. Mech. Eng. 82 (1) (1990) 281–300, https://doi.org/10.1016/0045-7825(90)90168-L, proceedings of the Workshop on Reliability in Computational Mechanics http://www.sciencedirect.com/science/article/pii/004578259090168L.
[50] P.E. Farrell, A. Birkisson, S.W. Funke, Deflation techniques for finding distinct solutions of nonlinear partial differential equations, SIAM J. Sci. Comput. 37 (4) (2015) A2026–A2045.
[51] K. Sohn, H. Lee, X. Yan, Learning structured output representation using deep conditional generative models, in: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 3483–3491, http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditional-generative-models.pdf.
[52] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv:1411.1784.
[53] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., Conditional image generation with pixelcnn decoders, in: Advances in Neural Information Processing Systems, 2016, pp. 4790–4798.
[54] Y. LeCun, S. Chopra, R. Hadsell, F.J. Huang, et al., A tutorial on energy-based learning, in: Predicting Structured Data, 2006.
[55] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, arXiv:1811.04026.

[56] A.v.d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G.v.d. Driessche, E. Lockhart, L.C. Cobo, F. Stimberg, et al., Parallel WaveNet: fast high-fidelity speech synthesis, arXiv:1711.10433.
[57] S.-H. Li, L. Wang, Neural network renormalization group, arXiv:1802.02840.
[58] F. Noé, H. Wu, Boltzmann generators - sampling equilibrium states of many-body systems with deep learning, arXiv:1812.01729.
[59] M.H. DeGroot, S.E. Fienberg, The comparison and evaluation of forecasters, J. R. Stat. Soc., Ser. D, Stat. (1983) 12–22, https://doi.org/10.2307/2987588.
[60] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, arXiv:1706.04599.
[61] V. Kuleshov, N. Fenner, S. Ermon, Accurate uncertainties for deep learning using calibrated regression, arXiv:1807.00263.
[62] D.P. Kingma, P. Dhariwal, Glow: generative flow with invertible 1×1 convolutions, arXiv:1807.03039.
[63] P. Hennig, M.A. Osborne, M. Girolami, Probabilistic numerics and uncertainty in computations, Proc. R. Soc. A 471 (2179) (2015) 20150142.
[64] J. Cockayne, C. Oates, T. Sullivan, M. Girolami, Probabilistic numerical methods for partial differential equations and Bayesian inverse problems, arXiv:1605.07811.
[65] J. Cockayne, C. Oates, T. Sullivan, M. Girolami, Bayesian probabilistic numerical methods, arXiv:1702.03673.
[66] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using Real NVP, arXiv:1605.08803.
[67] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv:1312.6114.
[68] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Proc. Advances in Neural Information Processing Systems 27, Curran Associates, Inc., 2014, pp. 2672–2680, http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.
[69] D.J. Rezende, S. Mohamed, Variational inference with normalizing flows, arXiv:1505.05770.
[70] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.
[71] M.S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M.E. Rognes, G.N. Wells, The FEniCS project version 1.5, Arch. Numer. Softw. 3 (100) (2015), https://doi.org/10.11588/ans.2015.100.20553.
[72] E. Laloy, R. Hérault, D. Jacques, N. Linde, Training-image based geostatistical inversion using a spatial generative adversarial neural network, Water Resour. Res. 54 (1) (2018) 381–406, https://doi.org/10.1002/2017WR022148, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017WR022148.
[73] K.O. Stanley, Compositional pattern producing networks: a novel abstraction of development, Genet. Program. Evol. Mach. 8 (2) (2007) 131–162.
[74] C. Mei, J.-L. Auriault, The effect of weak inertia on flow through a porous medium, J. Fluid Mech. 222 (1991) 647–663, https://doi.org/10.1017/S0022112091001258.
[75] M. Firdaouss, J.-L. Guermond, P. Le Quéré, Nonlinear corrections to Darcy's law at low Reynolds numbers, J. Fluid Mech. 343 (1997) 331–350, https://doi.org/10.1017/S0022112097005843.
[76] S. Rojas, J. Koplik, Nonlinear flow in porous media, Phys. Rev. E 58 (1998) 4776–4782, https://doi.org/10.1103/PhysRevE.58.4776, https://link.aps.org/doi/10.1103/PhysRevE.58.4776.
[77] P. Forchheimer, Wasserbewegung durch boden, Z. Ver. Dtsch. Ing. 45 (1901) 1782–1788.
[78] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv:1412.6980.
[79] G. Huang, Z. Liu, L.v.d. Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269.
[80] Anonymous, Diversity-sensitive conditional generative adversarial networks, in: Submitted to International Conference on Learning Representations, 2019, under review, https://openreview.net/forum?id=rJliMh09F7.
[81] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A.A. Efros, O. Wang, E. Shechtman, Toward multimodal image-to-image translation, in: Advances in Neural Information Processing Systems, 2017, pp. 465–476.
[82] Anonymous, Lagging inference networks and posterior collapse in variational autoencoders, in: Submitted to International Conference on Learning Representations, 2019, under review, https://openreview.net/forum?id=rylDfnCqF7.
[83] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: pre-training of deep bidirectional transformers for language understanding, arXiv:1810.04805.
[84] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.
[85] J. Adler, O. Öktem, Solving ill-posed inverse problems using iterative deep neural networks, Inverse Probl. 33 (12) (2017) 124007, http://stacks.iop.org/0266-5611/33/i=12/a=124007.
[86] K. Hammernik, T. Klatzer, E. Kobler, M.P. Recht, D.K. Sodickson, T. Pock, F. Knoll, Learning a variational network for reconstruction of accelerated MRI data, Magn. Reson. Med. 79 (6) (2018) 3055–3071, https://doi.org/10.1002/mrm.26977, https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.26977.
[87] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, arXiv:1703.03400.
[88] L. Yang, D. Zhang, G.E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, arXiv:1811.02033.
[89] W. Grathwohl, R.T.Q. Chen, J. Bettencourt, I. Sutskever, D. Duvenaud, FFJORD: free-form continuous dynamics for scalable reversible generative models, arXiv:1810.01367.
[90] E. Nalisnick, A. Matsukawa, Y.W. Teh, D. Gorur, B. Lakshminarayanan, Do deep generative models know what they don't know?, arXiv:1810.09136.
[91] H. Choi, E. Jang, Generative ensembles for robust anomaly detection, arXiv:1810.01392.
[92] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2) (2002) 99–127.