mathematics

MDPI

Article

Prediction of Discretization of GMsFEM Using Deep Learning

Min Wang [1], Siu Wun Cheung [1], Eric T. Chung [2], Yalchin Efendiev [1,3,*], Wing Tat Leung [4] and Yating Wang [1,5]

1 Department of Mathematics, Texas A&M University, College Station, TX 77843, USA; wangmin@math.tamu.edu (M.W.); tonycsw2905@math.tamu.edu (S.W.C.); wytgloria@math.tamu.edu (Y.W.)
2 Department of Mathematics, The Chinese University of Hong Kong, Hong Kong, China; tschung@math.cuhk.edu.hk
3 Multiscale Model Reduction Laboratory, North-Eastern Federal University, 677980 Yakutsk, Russia
4 Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, USA; sidnet123@gmail.com
5 Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA
* Correspondence: efendiev@math.tamu.edu

Received: 25 March 2019; Accepted: 30 April 2019; Published: 8 May 2019

Abstract: In this paper, we propose a deep-learning-based approach to a class of multiscale problems. The generalized multiscale finite element method (GMsFEM) has been proven successful as a model reduction technique of flow problems in heterogeneous and high-contrast porous media. The key ingredients of GMsFEM include mutlsicale basis functions and coarse-scale parameters, which are obtained from solving local problems in each coarse neighborhood. Given a fixed medium, these quantities are precomputed by solving local problems in an offline stage, and result in a reduced-order model. However, these quantities have to be re-computed in case of varying media (various permeability fields). The objective of our work is to use deep learning techniques to mimic the nonlinear relation between the permeability field and the GMsFEM discretizations, and use neural networks to perform fast computation of GMsFEM ingredients repeatedly for a class of media. We provide numerical experiments to investigate the predictive power of neural networks and the usefulness of the resultant multiscale model in solving channelized porous media flow problems.

Keywords: generalized multiscale finite element method; multiscale model reduction; deep learning

1. Introduction

Multiscale features widely exist in many engineering problems. For instance, in porous media flow, the media properties typically vary over many scales and contain high contrast. Multiscale finite element methods (MsFEM) [1–3] and generalized multiscale finite element methods (GMsFEM) [4,5] are designed for solving multiscale problems using local model reduction techniques. In these methods, the computational domain is partitioned into a coarse grid $\mathcal{T}^H$, which does not necessarily resolve all multiscale features. We further perform a refinement of $\mathcal{T}^H$ to obtain a fine grid $\mathcal{T}^h$, which essentially resolves all multiscale features. The idea of local model reduction in these methods is based on idenfications of local multiscale basis functions supported in coarse regions on the fine grid, and replacement of the macroscopic equations by a coarse-scale system using a limited number of local multiscale basis functions. As in many model reduction techniques, the computations of multiscale basis functions, which constitute a small dimensional subspace, can be performed in an offline stage. For a fixed medium, these multiscale basis functions are reusable for any force terms and boundary conditions. Therefore, these methods provide substantial computational savings in the online stage, in which a coarse-scale system is constructed and solved on the reduced-order space.

However, difficulties arise in situations with uncertainties in the media properties in some local regions, which are common for oil reservoirs or aquifers. One straightforward approach for quantifying the uncertainties is to sample realizations of media properties. In such cases, it is challenging to find an offline principal component subspace which is able to universally solve the multiscale problems with different media properties. The computation of multiscale basis functions has to be performed in an online procedure for each medium. Even though the multiscale basis functions are reusable for different force terms and boundary conditions, the computational effort can grow very huge for a large number of realizations of media properties. To this end, building a functional relationship between the media properties and the multiscale model in an offline stage can avoid repeating expensive computations and thus vastly reduce the computational complexity. Due to the diversity of complexity of the media properties, the functional relationship is highly nonlinear. Modelling such a nonlinear functional relationship typically involves high-order approximations. Therefore, it is natural to use machine learning techniques to devise such complex models. In [6,7], the authors make use of a Bayesian approach for learning multiscale models and incorporating essential observation data in the presence of uncertainties.

Deep neural networks is one class of machine learning algorithm that is based on an artificial neural network, which is composed of a relatively large number of layers of nonlinear processing units, called neurons, for feature extraction. The neurons are connected to other neurons in the successive layers. The information propagates from the input, through the intermediate hidden layers, and to the output layer. In the propagation process, the output in each layer is used as input in the consecutive layer. Each layer transforms its input data into a little more abstract feature representation. In between layers, a nonlinear activation function is used as the nonlinear transformation on the input, which increases the expressive power of neural networks. Recently, deep neural network (DNN) has been successfully used to interpret complicated data sets and applied to tasks with pattern recognition, such as image recognition, speech recognition and natural language processing [8–10]. Extensive researches have also been conducted on investigating the expression power of deep neural networks [11–15].

Results show that neural networks can represent and approximate a large class of functions. Recently, deep learning has been applied to model reductions and partial differential equations. In [16], the authors studied deep convolution networks for surrogate model construction. on dynamic flow problems in heterogeneous media. In [17], the authors studied the relationship between residual networks (ResNet) and characteristic equations of linear transport, and proposed an interpretation of deep neural networks by continuous flow models. In [18], the authors combined the idea of the Ritz method and deep learning techniques to solve elliptic problems and eigenvalue problems. In [19], a neural network has been designed to learn the physical quantities of interest as a function of random input coefficients. The concept of using deep learning to generate a reduced-order model for a dynamic flow has been applied to proper orthogonal decomposition (POD) global model reduction [20] and nonlocal multi-continuum upscaling (NLMC) [21].

In this work, we propose a deep-learning-based method for fast computation of the GMsFEM discretization. Our approach makes use of deep neural networks as a fast proxy to compute GMsFEM discretizations for flow problems in channelized porous media with uncertainties. More specifically, neural networks are used to express the functional relationship between the media properties and the multiscale model. Such networks are built up in an offline stage. Sufficient sample pairs are required to ensure the expressive power of the networks. With different realizations of media properties, one can use the built network and avoid computations of local problems and spectral problems.

The paper is organized as follows. We start with the underlying partial differential equation that describes the flow within a heterogeneous media and the main ingredients of GMsFEM in Section 2. Next, in Section 3, we present the idea of using deep learning as a proxy for prediction of GMsFEM discretizations. The networks will be precisely defined and the sampling will be explained in detail.

In Section 4, we present numerical experiments to show the effectiveness of our presented networks on several examples with different configurations. Finally, a conclusive discussion is provided in Section 5.

2. Preliminaries

In this paper, we are considering the flow problem in highly heterogeneous media

$$\begin{aligned} -\mathrm{div}(\kappa \nabla u) &= f \quad \text{in} \quad \Omega, \\ u = 0 \quad \text{or} \quad \frac{\partial u}{\partial n} &= 0 \quad \text{on} \quad \partial\Omega, \end{aligned} \tag{1}$$

where Ω is the computational domain, κ is the permeability coefficient in $L^\infty(\Omega)$, and f is a source function in $L^2(\Omega)$. We assume the coefficient κ is highly heterogeneous with high contrast. The classical finite element method for solving (1) numerically is given by: find $u_h \in V_h$ such that

$$a(u_h, v) = \int_\Omega \kappa \nabla u_h \cdot \nabla v \, dx = \int_\Omega f v \, dx = (f, v) \quad \text{for all } v \in V_h, \tag{2}$$

where V_h is a standard conforming finite element space over a partition $\mathcal{T}_h$ of Ω with mesh size h.

However, with the highly heterogeneous property of coefficient κ, the mesh size h has to be taken extremely small to capture the underlying fine-scale features of κ. This ends up with a large computational cost. GMsFEM [4,5] serves as a model reduction technique to reduce the number of degrees of freedom and attain both efficiency and accuracy to a considerable extent. GMsFEM has been successfully extended to other formulations and applied to other problems. Here we provide a brief introduction of the main ingredients of GMsFEM. For a more detailed discussion of GMsFEM and related concepts, the reader is referred to [22–26].

In GMsFEM, we define a coarse mesh $\mathcal{T}^H$ over the domain Ω and refine to obtain a fine mesh $\mathcal{T}^h$ with mesh size $h \ll H$, which is fine enough to restore the multiscale properties of the problem. Multiscale basis functions are defined on coarse grid blocks using linear combinations of finite element basis functions on $\mathcal{T}^h$, and designed to resolve the local multiscale behaviors of the exact solution. The multiscale finite element space V_{ms}, which is a principal component subspace of the conforming finite space V_h with $\dim(V_{\mathrm{ms}}) \ll \dim(V_h)$, is constructed by the linear span of multiscale basis functions. The multiscale solution $u_{\mathrm{ms}} \in V_{\mathrm{ms}}$ is then defined by

$$a(u_{\mathrm{ms}}, v) = (f, v) \quad \text{for all } v \in V_{\mathrm{ms}}. \tag{3}$$

We consider the identification of dominant modes for solving (1) by multiscale basis functions, including spectral basis functions and simplified basis functions, in GMsFEM. Here, we present the details of the construction of multiscale basis functions in GMsFEM. Let $N_x = \{x_i \,|1 \leq i \leq N_v\}$ be the set of nodes of the coarse mesh $\mathcal{T}^H$. For each coarse grid node $x_i \in N_x$, the coarse neighborhood ω_i is defined by

$$\omega_i = \bigcup \{K_j \in \mathcal{T}^H; \quad x_i \in \overline{K}_j\}, \tag{4}$$

that is, the union of the coarse elements $K_j \in \mathcal{T}^H$ containing the coarse grid node x_i. An example of the coarse and fine mesh, coarse blocks and a coarse neighborhood is shown in Figure 1. For each coarse neighbourhood ω_i, we construct multiscale basis functions $\{\phi_j^{\omega_i}\}_{j=1}^{L_i}$ supported on ω_i.

For the construction of spectral basis functions, we first construct a snapshot space $V_{\mathrm{snap}}^{(i)}$ spanned by local snapshot basis functions $\psi_{\mathrm{snap}}^{i,k}$ for each local coarse neighborhood ω_i. The snapshot basis function $\psi_{\mathrm{snap}}^{i,k}$ is the solution of a local problem

$$\begin{aligned} -\mathrm{div}(\kappa \nabla \psi_{\mathrm{snap}}^{i,k}) = 0, \quad \text{in} \quad \omega_i \\ \psi_{\mathrm{snap}}^{i,k} = \delta_{i,k}, \quad \text{on} \quad \partial\omega_i. \end{aligned} \tag{5}$$

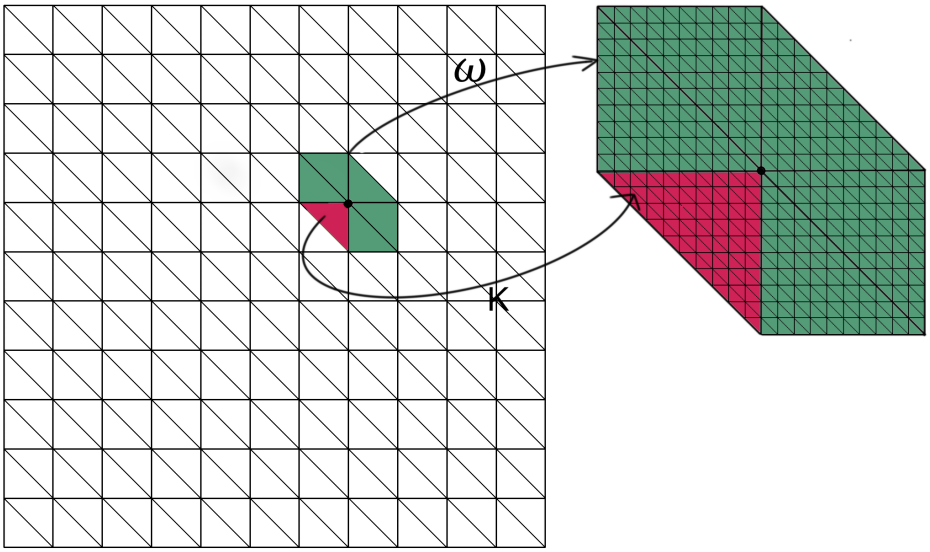

Figure 1. An illustration of coarse mesh (**left**), a coarse neighborhood and coarse blocks (**right**).

The fine grid function $\delta_{i,k}$ is a function defined for all $x_s \in \partial\omega_i$, where $\{x_s\}$ denote the fine degrees of freedom on the boundary of local coarse region ω_i. Specifically,

$$\delta_{i,k}(x_s) = \begin{cases} 1 & \text{if } s = k \\ 0 & \text{if } s \neq k. \end{cases} \tag{6}$$

The linear span of these harmonic extensions forms the local snapshot space $V_{\text{snap}}^{(i)} = \text{span}_k\{\psi_{\text{snap}}^{i,k}\}$. One can also use randomized boundary conditions to reduce the computational cost associated with snapshot calculations [25]. Next, a spectral problem is designed based on our analysis and used to reduce the dimension of local multiscale space. More precisely, we seek for eigenvalues λ_m^i and corresponding eigenfunctions $\phi_m^{\omega_i} \in V_{\text{snap}}^{(i)}$ satisfying

$$a_i(\phi_m^{\omega_i}, v) = \lambda_m^i s_i(\phi_m^{\omega_i}, v), \quad \forall v \in V_{\text{snap}}^{(i)}, \tag{7}$$

where the bilinear forms in the spectral problem are defined as

$$\begin{aligned} a_i(u, v) &= \int_{\omega_i} \kappa \nabla u \cdot \nabla v, \\ s_i(u, v) &= \int_{\omega_i} \tilde{\kappa} u v, \end{aligned} \tag{8}$$

where $\tilde{\kappa} = \sum_j \kappa |\nabla \chi_j|^2$, and χ_j denotes the multiscale partition of the unity function. We arrange the eigenvalues λ_m^i of the spectral problem (7) in ascending order, and select the first l_i eigenfunctions $\{\phi_m^{\omega_i}\}_{m=1}^{l_i}$ corresponding to the small eigenvalues as the multiscale basis functions.

An alternative way to construct the multiscale basis function is by using the idea of simplified basis functions. This approach assumes the number of channels and position of the channelized permeability field are known. Therefore we can obtain multiscale basis functions $\{\phi_m^{\omega_i}\}_{m=1}^{l_i}$ using these information and without solving the spectral problem [27].

Once the multiscale basis functions are constructed, the span of the multiscale basis functions will form the offline space

$$\begin{aligned} V_{ms}^{(i)} &= \text{span}\{\phi_m^{\omega_i}\}_{m=1}^{l_i}, \\ V_{ms} &= \oplus_i V_{ms}^{(i)}. \end{aligned} \tag{9}$$

We will then seek a multlscale solution $u_{ms} \in V_{ms}$ satisfying

$$a(u_{ms}, v) = (f, v) \quad \text{for all } v \in V_{ms}, \tag{10}$$

which is a Galerkin projection of the (1) onto V_{ms}, and can be written as a system of linear equations

$$A_c u_c = b_c, \tag{11}$$

where A_c and b_c are the coarse-scale stiffness matrix and load vector. If we collect all the multiscale basis functions and arrange the fine-scale coordinate representation in columns, we obtain the downscaling operator R. Then the fine-scale representation of the multiscale solution is given by

$$u_{ms} = R u_c. \tag{12}$$

3. Deep Learning for GMsFEM

In applications, there are uncertainties within some local regions of the permeability field κ in the flow problem. Thousands of forward simulations are needed to quantify the uncertainties of the flow solution. GMsFEM provides us with a fast solver to compute the solutions accurately and efficiently. Considering that there is a large amount of simulation data, we are interested in developing a method utilizing the existing offline data and reducing direct computational effort later. In this work, we aim at using DNN to model the relationship between heterogeneous permeability coefficient κ and the key ingredients of GMsFEM solver, i.e., coarse scale stiffness matrices and multiscale basis functions. When the relation is built up, we can feed the network any realization of the permeability field and obtain the corresponding GMsFEM ingredients, and further restore fine-grid GMsFEM solution of (1). The general idea of utilizing deep learning in the GMsFEM framework is illustrated in Figure 2.

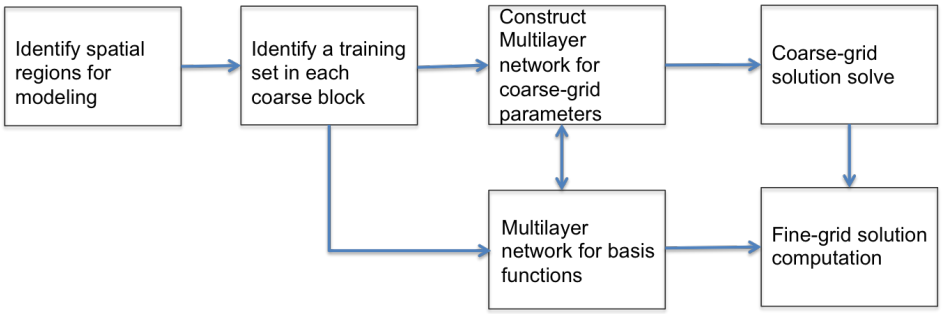

Figure 2. A flow chart in illustrating the idea of using deep learning in the generalized multiscale finite element method (GMsFEM) framework.

Suppose that there are uncertainties for the heterogeneous coefficient in a local coarse block K_0, which we call the target block, and the permeability outside the target block remains the same. For example, for a channelized permeability field, the position, location and the permeability values of the channels in the target block can vary. The target block K_0 is inside three coarse neighborhoods, denoted by $\omega_1, \omega_2, \omega_3$. The union of the 3 neighborhoods, i.e.,

$$\omega^+(K_0) = \omega_1 \cup \omega_2 \cup \omega_3, \tag{13}$$

are constituted of by the target block K_0 and 12 other coarse blocks, denoted by $\{K_l\}_{l=1}^{12}$ A target block and its surrounding neighborhoods are depicted in Figure 3.

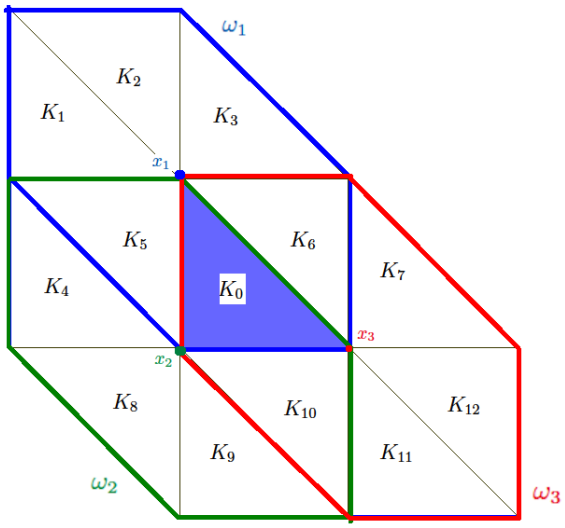

Figure 3. An illustration of a target coarse block K_0 and related neighborhoods.

For a fixed permeability field κ, one can compute the multiscale basis functions $\phi_m^{\omega_i}(\kappa)$ defined by (7), for $i = 1, 2, 3$, and the local coarse-scale stiffness matrices $A_c^{K_l}(\kappa)$, defined by

$$[A_c^{K_l}(\kappa)]_{m,n}^{i,j} = \int_{K_l} \kappa \nabla \phi_m^{\omega_i}(\kappa) \cdot \nabla \phi_n^{\omega_j}(\kappa), \tag{14}$$

for $l = 0, 1, \ldots, 12$. We are interested in constructing the maps $g_B^{m,i}$ and g_M^l, where

- $g_B^{m,i}$ maps the permeability coefficient κ to a local multiscale basis function $\phi_m^{\omega_i}$, where i denotes the index of the coarse block, and m denotes the index of the basis in coarse block ω_i

$$g_B^{m,i} : \kappa \mapsto \phi_m^{\omega_i}(\kappa), \tag{15}$$

- g_M^l maps the permeability coefficient κ to the coarse grid parameters $A_c^{K_l}$ $(l = 0, \cdots, 12)$

$$g_M^l : \kappa \mapsto A_c^{K_l}(\kappa). \tag{16}$$

In this work, our goal is to make use of deep learning to build fast approximations of these quantities associated with the uncertainties in the permeability field κ, which can provide fast and accurate solutions to the heterogeneous flow problem (1).

For each realization κ, one can compute the images of κ under the local multiscale basis maps $g_B^{m,i}$ and the local coarse-scale matrix maps g_M^l. These forward calculations serve as training samples for building a deep neural network for approximation of the corresponding maps, i.e.,

$$\begin{aligned} \mathcal{N}_B^{m,i}(\kappa) &\approx g_B^{m,i}(\kappa), \\ \mathcal{N}_M^l(\kappa) &\approx g_M^l(\kappa). \end{aligned} \tag{17}$$

In our networks, the permeability field κ is the input, while the multiscale basis functions $\phi_m^{\omega_i}$ and the coarse-scale matrix $A_c^{K_l}$ are the outputs. Once the neural networks are built, we can use the networks to compute the multiscale basis functions and coarse-scale parameters in the associated region for any permeability field κ. Using these local information from the neural networks together with the global information which can be pre-computed, we can form the downscale operator R with the multiscale basis functions, form and solve the linear system (11), and obtain the multiscale solution by (12).

3.1. Network Architecture

In general, an L-layer neural network $\mathcal{N}$ can be written in the form

$$\mathcal{N}(x;\theta) = \sigma(W_L\sigma(\cdots\sigma(W_2\sigma(W_1x+b_1)+b_2)\cdots)+b_L), \tag{18}$$

where $\theta := (W_1, W_2, \cdots, W_L, b_1, b_2, \cdots, b_L)$, W's are the weight matrices and b's are the bias vectors, σ is the activation function, x is the input. Such a network is used to approximate the output y. Our goal is then to find θ^* by solving an optimization problem

$$\theta^* = \underset{\theta}{\text{argmin}} \ \ \mathcal{L}(\theta), \tag{19}$$

where $\mathcal{L}(\theta)$ is called loss function, which measures the mismatch between the image of the input x under the the neural network $\mathcal{N}(x,y;\theta)$ and the desired output y in a set of training samples (x_j, y_j). In this paper, we use the mean-squared error metric to be our loss function

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{j=1}^{N}\|y_j - \mathcal{N}(x_j;\theta)\|_2^2, \tag{20}$$

where N is the number of the training samples. An illustration of a deep neural network is shown in Figure 4.

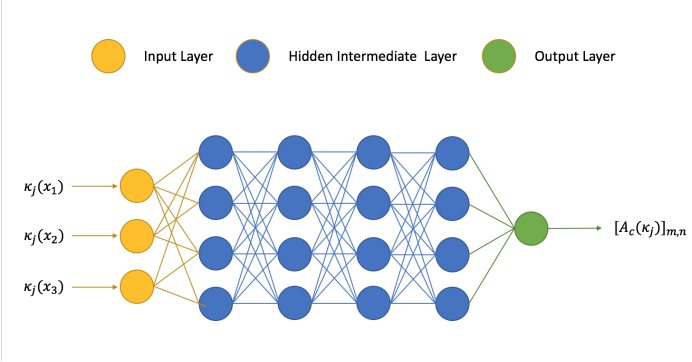

Figure 4. An illustration of a deep neural network.

Suppose we have a set of different realizations of the permeability $\{\kappa_1, \kappa_2, \cdots, \kappa_N\}$ in the target block. In our network, the input $x_j = \kappa_j \in \mathbb{R}^d$ is a vector containing the permeability image pixels in the target block. The output y_j is an entry of the local stiffness matrix $A_c^{K_l}$, or the coordinate representation of a multiscale basis function $\phi_m^{\omega_i}$. We will make use of these sample pairs (x_j, y_j) to train a deep neural network $\mathcal{N}_B^{m,i}(x;\theta_B^*)$ and $\mathcal{N}_M^l(x;\theta_M^*)$ by minimizing the loss function with respect to the network parameter θ, such that the trained neural networks can approximate the functions $g_B^{m,i}$ and g_M^l, respectively. Once the neural is constructed, for some given new permeability coefficient κ_{N+1}, we use our trained networks to compute a fast prediction of the outputs, i.e., local multiscale basis functions $\phi_m^{\omega_i,\text{pred}}$ by

$$\phi_m^{\omega_i,\text{pred}}(\kappa_{N+1}) = \mathcal{N}_B^{m,i}(\kappa_{N+1};\theta_B^*) \approx g_B^{m,i}(\kappa_{N+1}) = \phi_m^{\omega_i}(\kappa_{N+1}), \tag{21}$$

and local coarse-scale stiffness matrix $A_c^{K_l,\text{pred}}$ by

$$A_c^{K_l,\text{pred}}(\kappa_{N+1}) = \mathcal{N}_M^l(\kappa_{N+1};\theta_M^*) \approx g_M^l(\kappa_{N+1}) = A_c^{K_l}(\kappa_{N+1}). \tag{22}$$

3.2. Network-Based Multiscale Solver

Once the neural networks are built, we can assemble the predicted multiscale basis functions to obtain a prediction R^{pred} for the downscaling operator, and assemble the predicted local coarse-scale stiffness matrix $A_c^{K_l,\text{pred}}$ in the global matrix A_c^{pred}. Following (11) and (12), we solve the predicted coarse-scale coefficient vector u_c^{pred} from the following linear system

$$A_c^{\text{pred}} u_c^{\text{pred}} = b_c, \tag{23}$$

and obtain the predicted multiscale solution u_{ms}^{pred} by

$$u_{ms}^{\text{pred}} = R^{\text{pred}} u_c^{\text{pred}}. \tag{24}$$

4. Numerical Results

In this section, we present some numerical results for predicting the GMsFEM ingredients and solutions using our proposed method. We considered permeability fields κ with high-contrast channels inside the domain $\Omega = (0,1)^2$, which consist of uncertainties in a target cell K_0. More precisely, we considered a number of random realizations of permeability fields $\kappa_1, \kappa_2, \kappa_3, \cdots, \kappa_{N+M}$. Each permeability field contained two high-conductivity channels, and the fields differ in the target cell K_0 by:

- in experiment 1, the channel configurations were all distinct, and the permeability coefficients inside the channels were fixed in each sample (see Figure 5 for illustrations), and
- in experiment 2, the channel configurations were randomly chosen among five configurations, and the permeability coefficients inside the channels followed a random distribution (see Figure 6 for illustrations).

In these numerical experiments, we assumed there were uncertainties in only the target block K_0. The permeability field in $\Omega \setminus K_0$ was fixed across all the samples.

We followed the procedures in Section 3 and generated sample pairs using GMsFEM. Local spectral problems were solved to obtain the multiscale basis functions $\phi_m^{\omega_i}$. In the neural network, the permeability field $x = \kappa$ was considered to be the input, while the local multiscale basis functions $y = \phi_m^{\omega_i}$ and local coarse-scale matrices $y = A_c^{K_l}$ were regarded as the output. These sample pairs were divided into the training set and the learning set in a random manner. A large number N of realizations, namely $\kappa_1, \kappa_2, \ldots, \kappa_N$, were used to generate sample pairs in the training set, while the remaining M realizations, namely, $\kappa_{N+1}, \kappa_{N+2}, \ldots, \kappa_{N+M}$ are used in testing the predictive power of the trained network. We remark that, for each basis function and each local matrix, we solved an optimization problem in minimizing the loss function defined by the sample pairs in the training set, and build a separate deep neural network. We summarize the network architectures for training local coarse scale stiffness matrix and multiscale basis functions as below:

- For the multiscale basis function $\phi_m^{\omega_i}$, we built a network $\mathcal{N}_B^{m,i}$ using
  - Input: vectorized permeability pixels values κ,
  - Output: coefficient vector of multiscale basis $\phi_m^{\omega_i}(\kappa)$ on coarse neighborhood ω_i,
  - Loss function: mean squared error $\frac{1}{N}\sum_{j=1}^{N} ||\phi_m^{\omega_i}(\kappa_j) - \mathcal{N}_B^{m,i}(\kappa_j;\theta_B)||_2^2$,
  - Activation function: leaky ReLu function,
  - DNN structure: 10–20 hidden layers, each layer have 250–350 neurons,
  - Training optimizer: Adamax.
- For the local coarse scale stiffness matrix $A_c^{K_l}$, we build a network $\mathcal{N}_M^l$ using

- Input: vectorized permeability pixels values κ,
- Output: vectorized coarse scale stiffness matrix $A_c^{K_l}(\kappa)$ on the coarse block K_l,
- Loss function: mean squared error $\frac{1}{N}\sum_{j=1}^{N}||A_c^{K_l}(\kappa_j) - \mathcal{N}_M^l(\kappa_j;\theta_M)||_2^2$,
- Activation function: ReLu function (rectifier),
- DNN structure: 10–16 hidden layers, each layer have 100–500 neurons,
- Training optimizer: Proximal Adagrad.

For simplicity, the activation functions ReLU function [28] and Leaky ReLU function were used as they have the simplest derivatives among all nonlinear functions. The ReLU function proved to be useful in training deep neural network architectures. The Leaky ReLU function can resolve the vanishing gradient problem which can accelerate the training in some occasions. The optimizers Adamax and Proximal Adagrad are stochastic gradient descent (SGD)-based methods commonly used in neural network training [29]. In both experiments, we trained our network using Python API Tensorflow and Keras [30].

Once a neural network was built on training, it can be used to predict the output given a new input. The accuracy of the predictions is essential in making the network useful. In our experiments, we used M sample pairs, which were not used in training the network, to examine the predictive power of our network. On these sample pairs, referred to as the testing set, we compared the prediction and the exact output and computed the mismatch in some suitable metric. Here, we summarize the metric used in our numerical experiment. For the multiscale basis functions, we compute the relative error in L^2 and H^1 norm, i.e.,

$$\begin{aligned} e_{L^2}(\kappa_{N+j}) &= \left(\frac{\int_\Omega \left|\phi_m^{\omega_i}(\kappa_{N+j}) - \phi_m^{\omega_i,\text{pred}}(\kappa_{N+j})\right|^2}{\int_\Omega \left|\phi_m^{\omega_i}(\kappa_{N+j})\right|^2}\right)^{\frac{1}{2}}, \\ e_{H^1}(\kappa_{N+j}) &= \left(\frac{\int_\Omega \left|\nabla\phi_m^{\omega_i}(\kappa_{N+j}) - \nabla\phi_m^{\omega_i,\text{pred}}(\kappa_{N+j})\right|^2}{\int_\Omega \left|\nabla\phi_m^{\omega_i}(\kappa_{N+j})\right|^2}\right)^{\frac{1}{2}}. \end{aligned} \tag{25}$$

For the local stiffness matrices, we computed the relative error in entrywise ℓ^2, entrywise ℓ^∞ and Frobenius norm, i.e.,

$$\begin{aligned} e_{\ell^2}(\kappa_{N+j}) &= \frac{\|A_c^{K_l}(\kappa_{N+j}) - A_c^{K_l,\text{pred}}(\kappa_{N+j})\|_2}{\|A_c^{K_l}(\kappa_{N+j})\|_2}, \\ e_{\ell^\infty}(\kappa_{N+j}) &= \frac{\|A_c^{K_l}(\kappa_{N+j}) - A_c^{K_l,\text{pred}}(\kappa_{N+j})\|_\infty}{\|A_c^{K_l}(\kappa_{N+j})\|_\infty}, \\ e_F(\kappa_{N+j}) &= \frac{\|A_c^{K_l}(\kappa_{N+j}) - A_c^{K_l,\text{pred}}(\kappa_{N+j})\|_F}{\|A_c^{K_l}(\kappa_{N+j})\|_F}. \end{aligned} \tag{26}$$

A more important measure of the usefulness of the trained neural network is the predicted multiscale solution $u_{ms}^{\text{pred}}(\kappa)$ given by (23) and (24). We compared the predicted solution to u_{ms} defined by (11) and (12), and computed the relative error in L^2 and energy norm, i.e.,

$$e_{L^2}(\kappa_{N+j}) = \left(\frac{\int_\Omega \left| u_{ms}(\kappa_{N+j}) - u_{ms}^{\text{pred}}(\kappa_{N+j}) \right|^2}{\int_\Omega \left| u_{ms}(\kappa_{N+j}) \right|^2} \right)^{\frac{1}{2}},$$
$$e_a(\kappa_{N+j}) = \left(\frac{\int_\Omega \kappa_j \left| \nabla u_{ms}(\kappa_{N+j}) - \nabla u_{ms}^{\text{pred}}(\kappa_{N+j}) \right|^2}{\int_\Omega \kappa_j \left| \nabla u_{ms}(\kappa_{N+j}) \right|^2} \right)^{\frac{1}{2}}. \quad (27)$$

4.1. Experiment 1

In this experiment, we considered curved channelized permeability fields. Each permeability field contained a straight channel and a curved channel. The straight channel was fixed and the curved channel struck the boundary of the target cell K_0 at the same points. The curvature of the sine-shaped channel inside K_0 varied among all realizations. We generated 2000 realizations of permeability fields, where the permeability coefficients were fixed. Samples of permeability fields are depicted in Figure 5. Among the 2000 realizations, 1980 sample pairs were randomly chosen and used as training samples, and the remaining 20 sample pairs were used as testing samples.

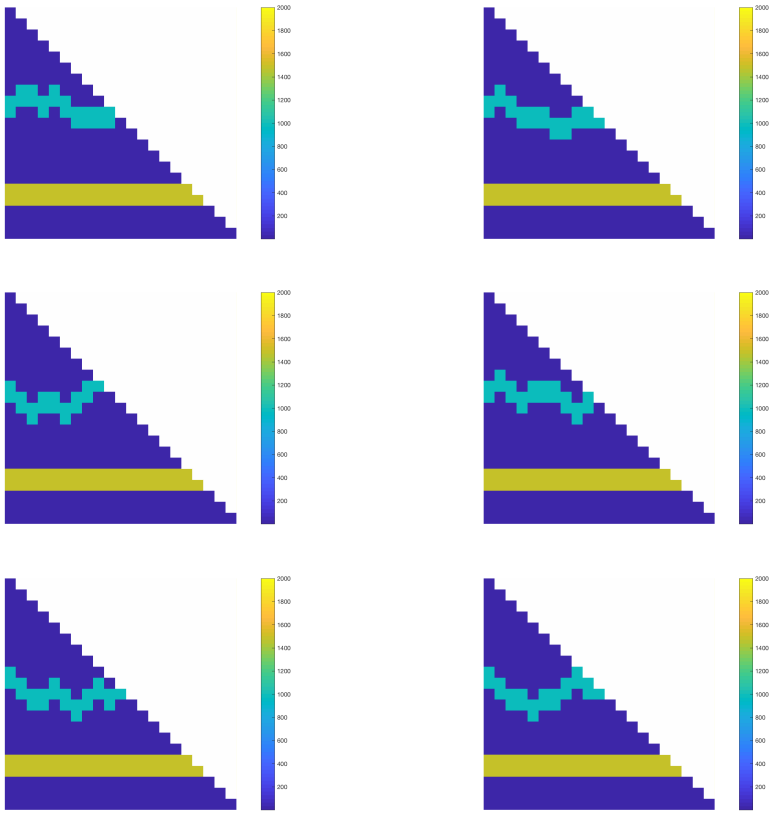

Figure 5. Samples of permeability fields in the target block K_0 in experiment 1.

For each realization, we computed the local multiscale basis functions and local coarse-scale stiffness matrix. In building the local snapshot space, we solved for harmonic extension of all the fine-grid boundary conditions. Local multiscale basis functions were then constructed by solving the spectral problem and multiplied the spectral basis functions with the multiscale partition of unity functions. With the offline space constructed, we computed the coarse-scale stiffness matrix. We used the training samples to build deep neural networks for approximating these GMsFEM quantities, and examined the performance of the approximations on the testing set.

Tables 1–3 record the error of the prediction by the neural networks in each testing sample and the mean error measured in the defined metric. It can be seen that the predictions were of high accuracy. This is vital in ensuring the predicted GMsFEM solver is useful. Table 4 records the error of the multiscale solution in each testing sample and the mean error using our proposed method. It can be observed that using the predicted GMsFEM solver, we obtained a good approximation of the multiscale solution compared with the exact GMsFEM solver.

Table 1. Percentage error of multiscale basis functions $\phi_1^{\omega_i}$ in experiment 1.

Sample	ω_1		ω_2		ω_3	
j	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}
1	0.47%	3.2%	0.40%	3.6%	0.84%	5.1%
2	0.45%	4.4%	0.39%	3.3%	1.00%	6.3%
3	0.34%	2.3%	0.40%	3.1%	0.88%	4.3%
4	0.35%	4.2%	0.43%	5.4%	0.94%	6.6%
5	0.35%	3.3%	0.37%	3.9%	0.90%	6.1%
6	0.51%	4.7%	0.92%	12.0%	2.60%	19.0%
7	0.45%	4.1%	0.38%	3.2%	1.00%	6.4%
8	0.31%	3.4%	0.43%	5.5%	1.10%	7.7%
9	0.25%	2.2%	0.46%	5.6%	1.10%	6.2%
10	0.31%	3.5%	0.42%	4.5%	1.30%	7.6%
Mean	0.38%	3.5%	0.46%	5.0%	1.17%	7.5%

Table 2. Percentage error of multiscale basis functions $\phi_2^{\omega_i}$ in experiment 1.

Sample	ω_1		ω_2		ω_3	
j	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}
1	0.47%	4.2%	0.40%	1.4%	0.32%	1.1%
2	0.57%	3.2%	0.31%	1.4%	0.30%	1.1%
3	0.58%	2.7%	0.31%	1.4%	0.33%	1.1%
4	0.59%	3.6%	0.13%	1.3%	0.32%	1.1%
5	0.53%	4.0%	0.51%	1.6%	0.27%	1.0%
6	0.85%	4.3%	0.51%	2.1%	0.29%	1.3%
7	0.50%	2.7%	0.22%	1.5%	0.29%	1.0%
8	0.43%	4.5%	0.61%	1.9%	0.35%	1.1%
9	0.71%	2.9%	0.14%	1.4%	0.27%	1.1%
10	0.66%	4.4%	0.53%	1.8%	0.26%	1.1%
Mean	0.59%	3.6%	0.37%	1.6%	0.30%	1.1%

Table 3. Percentage error of the local stiffness matrix $A_c^{K_0}$ in experiment 1.

Sample j	e_{ℓ^2}	e_F
1	0.67%	0.84%
2	0.37%	0.37%
3	0.32%	0.38%
4	1.32%	1.29%
5	0.51%	0.59%
6	4.43%	4.28%
7	0.34%	0.38%
8	0.86%	1.04%
9	1.00%	0.97%
10	0.90%	1.08%
Mean	0.76%	0.81%

Table 4. Percentage error of multiscale solution u_{ms} in experiment 1.

Sample j	e_{L^2}	e_a
1	0.31%	4.58%
2	0.30%	4.60%
3	0.30%	4.51%
4	0.27%	4.60%
5	0.29%	4.56%
6	0.47%	4.67%
7	0.39%	4.70%
8	0.30%	4.63%
9	0.35%	4.65%
10	0.31%	4.65%
Mean	0.33%	4.62%

4.2. Experiment 2

In this experiment, we considered sine-shaped channelized permeability fields. Each permeability field contained a straight channel and a sine-shaped channel. There were altogether five channel configurations, where the straight channel was fixed and the sine-shaped channel struck the boundary of the target cell K_0 at the same points. The curvature of the sine-shaped channel inside K_0 varied among these configurations. For each channel configuration, we generated 500 realizations of permeability fields, where the permeability coefficients followed random distributions. Samples of permeability fields are depicted in Figure 6. Among the 2500 realizations, 2475 sample pairs were randomly chosen and used as training samples, and the remaining 25 sample pairs were used as testing samples.

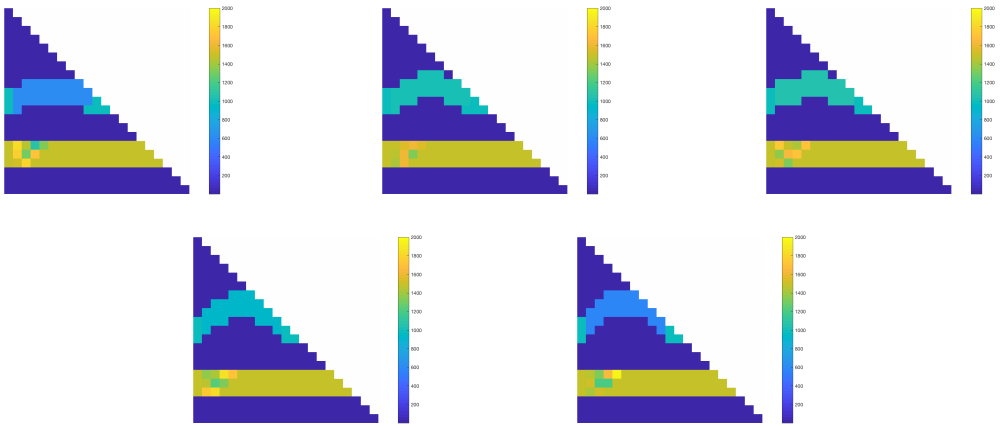

Figure 6. Samples of permeability fields in the target block K_0 in experiment 2.

Next, for each realization, we computed the local multiscale basis functions and local coarse-scale stiffness matrix. In building the local snapshot space, we solved for harmonic extension of randomized fine-grid boundary conditions, so as to reduce the number of local problems to be solved. Local multiscale basis functions were then constructed by solving the spectral problem and multiplied the spectral basis functions with the multiscale partition of unity functions. With the offline space constructed, we computed the coarse-scale stiffness matrix. We used the training samples to build deep neural networks for approximating these GMsFEM quantities, and examined the performance of the approximations on the testing set.

Figures 7–9 show the comparison of the multiscale basis functions in two respective coarse neighborhoods. It can be observed that the predicted multiscale basis functions were in good agreement with the exact ones. In particular, the neural network successfully interpreted the high conductivity regions as the support localization feature of the multiscale basis functions. Tables 5 and 6 record the mean error of the prediction by the neural networks, measured in the defined metric. Again, it can be seen that the prediction are of high accuracy. Table 7 records the mean error between the multiscale solution using the neural-network-based multiscale solver and using exact GMsFEM. we obtain a good approximation of the multiscale solution compared with the exact GMsFEM solver.

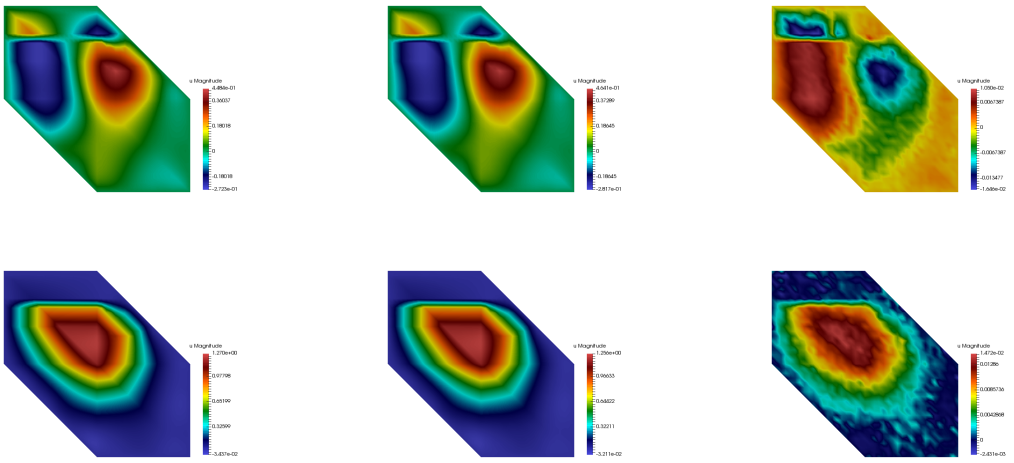

Figure 7. Exact multiscale basis functions $\phi_m^{\omega_1}$ (**left**), predicted multiscale basis functions $\phi_m^{\omega_1,\text{pred}}$ (**middle**) and their differences (**right**) in the coarse neighborhood ω_1 in experiment 2. The first row and the second row illustrate the first basis function $\phi_1^{\omega_1}$ and the second basis function $\phi_2^{\omega_1}$, respecitvely.

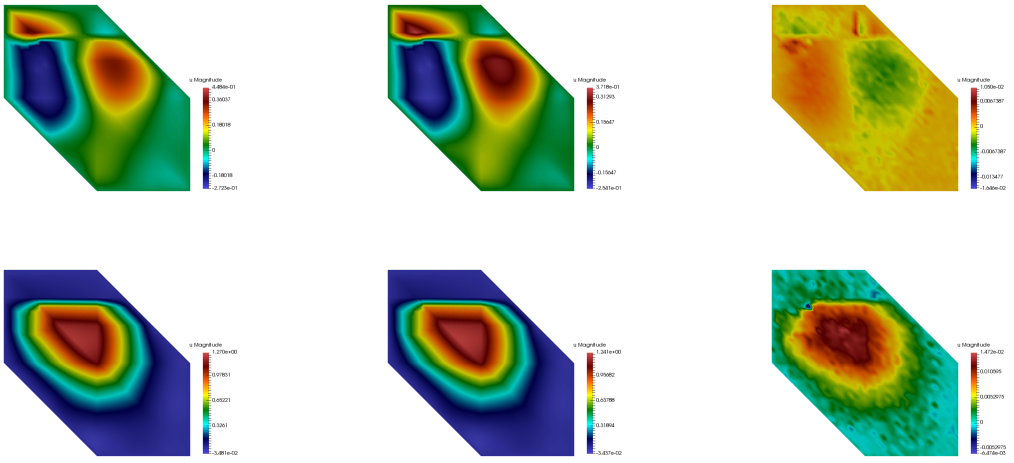

Figure 8. Exact multiscale basis functions $\phi_m^{\omega_2}$ (**left**), predicted multiscale basis functions $\phi_m^{\omega_2,\text{pred}}$ (**middle**) and their differences (**right**) in the coarse neighborhood ω_2 in experiment 2. The first row and the second row illustrate the first basis function $\phi_1^{\omega_2}$ and the second basis function $\phi_2^{\omega_2}$, respecitvely.

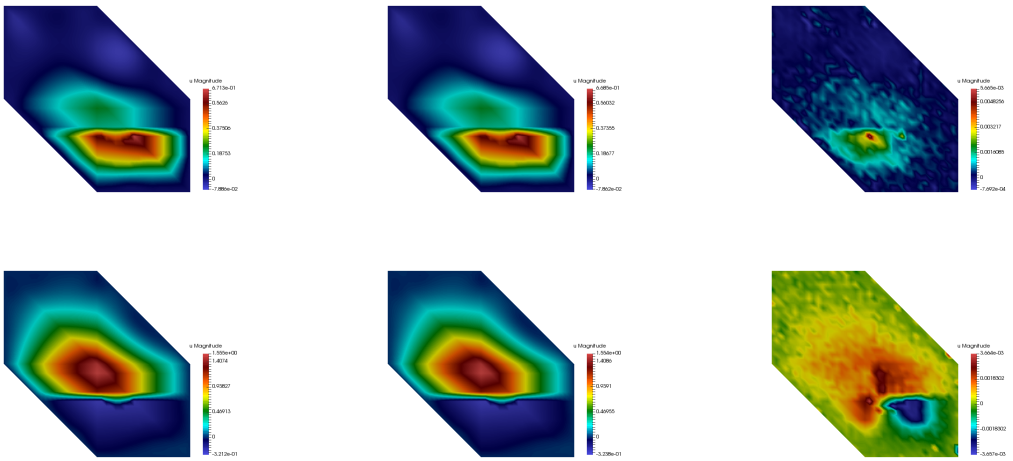

Figure 9. Exact multiscale basis functions $\phi_m^{\omega_3}$ (**left**), predicted multiscale basis functions $\phi_m^{\omega_3,\text{pred}}$ (**middle**) and their differences (**right**) in the coarse neighborhood ω_3 in experiment 2. The first row and the second row illustrate the first basis function $\phi_1^{\omega_3}$ and the second basis function $\phi_2^{\omega_3}$, respecitvely.

Table 5. Mean percentage error of multiscale basis functions $\phi_m^{\omega_i}$ in experiment 2.

Basis	ω_1		ω_2		ω_3	
m	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}
1	0.55	0.91	0.37	3.02	0.20	0.63
2	0.80	1.48	2.17	3.55	0.27	1.51

Table 6. Percentage error of the local stiffness matrix $A_c^{K_0}$ in experiment 2.

	e_{ℓ^2}	e_{ℓ^∞}	e_F
Mean	0.75	0.72	0.80

Table 7. Percentage error of multiscale solution u_{ms} in experiment 2.

	e_{L^2}	e_a
Mean	0.03	0.26

5. Conclusions

In this paper, we develop a method using deep learning techniques for fast computation of GMsFEM discretizations. Given a particular permeability field, the main ingredients of GMsFEM, including the multiscale basis functions and coarse-scale matrices, are computed in an offline stage by solving local problems. However, when one is interested in calculating GMsFEM discretizations for multiple choices of permeability fields, repeatedly formulating and solving such local problems could become computationally expensive or even infeasible. Multi-layer networks are used to represent the nonlinear mapping from the fine-scale permeability field coefficients to the multiscale basis functions and the coarse-scale parameters. The networks provide a direct fast approximation of the GMsFEM ingredients in a local neighborhood for any online permeability fields, in contrast to repeatedly formulating and solving local problems. Numerical results are presented to show the performance of our proposed method. We see that, given sufficient samples of GMsFEM discretizations for supervised training, deep neural networks are capable of providing reasonably close approximations of the exact GMsFEM discretization. Moreover, the small consistency error provides good approximations of multiscale solutions.

Author Contributions: The authors have contributed equally to the work.

Funding: The research of Eric Chung is partially supported by the Hong Kong RGC General Research Fund (Project numbers 14304217 and 14302018) and CUHK Faculty of Science Direct Grant 2017-18. YE would like to acknowledge the support of Mega-grant of the Russian Federation Government (N 14.Y26.31.0013) and the partial support from NSF 1620318.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Efendiev, Y.; Hou, T. *Multiscale Finite Element Methods: Theory and Applications*; Surveys and Tutorials in the Applied Mathematical Sciences; Springer: New York, NY, USA, 2009; Volume 4.
2. Hou, T.; Wu, X. A multiscale finite element method for elliptic problems in composite materials and porous media. *J. Comput. Phys.* **1997**, *134*, 169–189. [CrossRef]
3. Jenny, P.; Lee, S.; Tchelepi, H. Multi-scale finite volume method for elliptic problems in subsurface flow simulation. *J. Comput. Phys.* **2003**, *187*, 47–67. [CrossRef]
4. Efendiev, Y.; Galvis, J.; Hou, T. Generalized multiscale finite element methods (GMsFEM). *J. Comput. Phys.* **2013**, *251*, 116–135. [CrossRef]
5. Chung, E.; Efendiev, Y.; Hou, T.Y. Adaptive multiscale model reduction with Generalized Multiscale Finite Element Methods. *J. Comput. Phys.* **2016**, *320*, 69–95. [CrossRef]
6. Efendiev, Y.; Leung, W.T.; Cheung, S.W.; Guha, N.; Hoang, V.H.; Mallick, B. Bayesian Multiscale Finite Element Methods. Modeling Missing Subgrid Information Probabilistically. *Int. J. Multiscale Comput. Eng.* **2017**. [CrossRef]
7. Cheung, S.W.; Guha, N. Dynamic Data-driven Bayesian GMsFEM. *arXiv* **2018**, arXiv:1806.05832.
8. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, 1097–1105. Available online: http://www.informationweek.com/news/201202317 (accessed on 24 March 2019). [CrossRef]
9. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.R.; Jaitly, N.; Senior, A. Approximation Capabilities of Multilayer Feedforward Networks. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
11. Cybenko, G. Approximations by superpositions of sigmoidal functions. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
12. Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.* **1991**, *4*, 251–257. [CrossRef]
13. Telgrasky, M. Benefits of depth in neural nets. *arXiv* **2016**, arXiv:1602.04485.
14. Liao, H.M.; Poggio, T. Learning functions: When is deep better than shallow. *arXiv* **2016**, arXiv:1603.00988v4.
15. Hanin, B. Universal function approximation by deep neural nets with bounded width and relu activations. *arXiv* **2017**, arXiv:1708.02691.
16. Zhu, Y.; Zabaras, N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *J. Comput. Phys.* **2018**, *366*, 415–447. [CrossRef]
17. Li, Z.; Shi, Z. Deep Residual Learning and PDEs on Manifold. *arXiv* **2017**, arXiv:1708.05115.
18. Weinan, E.; Yu, B. The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *arXiv* **2017**, arXiv:1710.00211.
19. Khoo, Y.; Lu, J.; Ying, L. Solving parametric PDE problems with artificial neural networks. *arXiv* **2017**, arXiv:1707.03351.
20. Cheung, S.W.; Chung, E.T.; Efendiev, Y.; Gildin, E.; Wang, Y. Deep Global Model Reduction Learning. *arXiv* **2018**, arXiv:1807.09335.
21. Wang, Y.; Cheung, S.W.; Chung, E.T.; Efendiev, Y.; Wang, M. Deep Multiscale Model Learning. *arXiv* **2018**, arXiv:1806.04830.
22. Chung, E.; Efendiev, Y.; Leung, W.T. Generalized multiscale finite element method for wave propagation in heterogeneous media. *SIAM Multicale Model. Simul.* **2014**, *12*, 1691–1721. [CrossRef]
23. Chung, E.; Efendiev, Y.; Lee, C. Mixed generalized multiscale finite element methods and applications. *SIAM Multicale Model. Simul.* **2014**, *13*, 338–366. [CrossRef]

24. Efendiev, Y.; Galvis, J.; Li, G.; Presho, M. Generalized Multiscale Finite Element Methods Oversampling strategies. *Int. J. Multiscale Comput. Eng.* **2013**, *12*, 465–484. [CrossRef]
25. Calo, V.; Efendiev, Y.; Galvis, J.; Li, G. Randomized Oversampling for Generalized Multiscale Finite Element Methods. *arXiv* **2014**, arXiv:1409.7114.
26. Chung, E.T.; Efendiev, Y.; Leung, W.T. Residual-driven online Generalized Multiscale Finite Element Methods. *J. Comput. Phys.* **2015**, *302*, 176–190. [CrossRef]
27. Chung, E.T.; Efendiev, Y.; Leung, W.T.; Vasilyeva, M.; Wang, Y. Non-local Multi-continua upscaling for flows in heterogeneous fractured media. *arXiv* **2018**, arXiv:1708.08379.
28. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. PMLR, Ft. Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
29. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
30. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 1 May 2019).