mathematics

Article

Learning Algorithms for Coarsening Uncertainty Space and Applications to Multiscale Simulations

Zecheng Zhang [1], Eric T. Chung [2], Yalchin Efendiev [1,3,*] and Wing Tat Leung [4]

[1] Department of Mathematics, Texas A&M University, College Station, TX 77843, USA; tom_z_z@tamu.edu
[2] Department of Mathematics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, China; tschung@math.cuhk.edu.hk
[3] Multiscale Model Reduction Laboratory, North-Eastern Federal University, Yakutsk 677980, Russia
[4] Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, USA; sidnet123@gmail.com
* Correspondence: efendiev@math.tamu.edu

Received: 21 March 2020; Accepted: 24 April 2020; Published: 4 May 2020

Abstract: In this paper, we investigate and design multiscale simulations for stochastic multiscale PDEs. As for the space, we consider a coarse grid and a known multiscale method, the generalized multiscale finite element method (GMsFEM). In order to obtain a small dimensional representation of the solution in each coarse block, the uncertainty space needs to be partitioned (coarsened). This coarsenining collects realizations that provide similar multiscale features as outlined in GMsFEM (or other method of choice). This step is known to be computationally demanding as it requires many local solves and clustering based on them. In this work, we take a different approach and learn coarsening the uncertainty space. Our methods use deep learning techniques in identifying clusters (coarsening) in the uncertainty space. We use convolutional neural networks combined with some techniques in adversary neural networks. We define appropriate loss functions in the proposed neural networks, where the loss function is composed of several parts that includes terms related to clusters and reconstruction of basis functions. We present numerical results for channelized permeability fields in the examples of flows in porous media.

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

1. Introduction

Many problems are multiscale with uncertainties. Examples include problems in porous media, material sciences, biological sciences, and so on. For example, in porous media applications, engineers can obtain fine-scale data about pore geometries or subsurface properties at very fine resolutions. These data are obtained in some spatial locations and then generalized to the entire reservoir domain. As a result, one uses geostatistical or other statistical tools to populate the media properties in space. The resulting porous media properties are stochastic and one needs to deal with many porous media realizations, where each realization is multiscale and varies at very fine scales. There are other realistic problems which have multiscale properties with uncertainties such as the multiscale public safety systems, [1], multiscale social networks [2]; these problems usually have more data.

Simulating each realization can be computationally expensive because of the media's multiscale nature. Our objective is to simulate many of these realizations. To address the issues associated with spatial and temporal scales, many multiscale methods have been developed [3–12]. These methods perform simulations on the coarse grid by developing reduced-order models. However, developing reduced-order models requires local computations, which can be expensive when one deals with many realizations. For this reason, some type of coarsening of the uncertainty space is needed [13]. In this paper, we consider some novel approaches for developing coarsening of uncertainty space as discussed below.

To coarsen the uncertainty space, clustering algorithms are often used; but a proper distance function should be designed in order to make the clusters have physical sense and achieve a reduction in the uncertainty space. The paper [13] proposed a method that uses the distance between local solutions. The motivation is that the local problems with random boundary conditions can represent the main models with all boundary conditions. Due to a high dimension of the uncertainty space, the authors in [13] proposed to compute the local solutions of only several realizations and then use the Karhunen–Loeve expansion [14] to approximate the solutions of all the other realizations. The distance function is then defined to be the distance between solutions and the standard K-means [15] algorithm is used to cluster the uncertainty space.

The issue with this method is computing the local solutions in the local neighborhoods. It is computationally expensive to compute the local solutions; although the KL expansion can save time to approximate the solutions of other realizations, one still needs to decide how many selected realizations we need to represent all the other solutions. In this paper, we propose the use of deep learning methodology and avoid explicit clustering as in earlier works. We remark that the development of deep learning techniques for multiscale simulations are recently reported in [16–20].

In this work, to coarsen the uncertainty space, we propose a deep learning algorithm which will learn the clusters for each local neighborhood. Due the nature of the permeability fields, we can use the transfer learning which uses the parameters of one local neighborhood to initialize the learning of all the other neighborhoods. This saves significantly computational time.

The auto encoder structure [21] has been widely used in improving the K-mean clustering algorithm [22–24]. The idea is to use the encoder to extract features and reduce the dimension; the encoding process can also be taken as a kernel method [25] which maps the data to a space which is easier to be separated. The decoder is used to upsample the latent space (reduced dimension feature space) back to the input space. The clustering algorithm is then used to cluster the latent space, which will save time due to the low dimension of the latent space and also preserve the accuracy due to the features extracted by the encoder.

Traditionally, the learning process is only involved in reconstructing the input space. Such kind of methods ignore the features extracted by latent space; so, it is not clear if the latent space is good enough to represent the input space and is easily clustered by the K-means method. In [24], the authors proposed a new loss which includes the reconstruction loss meanwhile the loss results from the clustering. The authors claimed that the new loss improves the clustering results.

We will apply the auto encoder structure and the multiple loss function; however, we will design the auto encoder as a generative network, i.e., the input and output space are different. More precisely, the input is the uncertain space (permeability fields) and the output will be the multiscale functions co-responding to the uncertain space. Intuitively, we want to use the multiscale basis to supervise the learning of the clusters so that the clusters will inherit the property of the solution. The motivation is the multiscale basis can somehow represent the real solutions and permeability fields; hence, the latent space is no longer good for clustering the input space but will be suitable for representing the multiscale basis function space.

To define the reconstructing loss, the common idea is the mean square error (MSE); but many works [26–29] have shown that the MSE tends to produce the average effect. In fact, in the area of image super-resolution [26–36] and other low level computer vision tasks, the generated images are usually over-smooth if trained using MSE. The theory is the MSE will capture the low frequency features like the background which is relatively steady; but for images with high contrast, the MSE will usually try to blur the images and the resulting images will lose the colorfulness and become less vivid [26]. Our problem has multiscale nature and we want to capture the dominant modes and multiscale features, hence a single MSE is clearly not enough.

Following the idea from [27,29], we consider adding an adversary net [37]. The motivation is the fact that different layers of fully convolutional network extract different features [29,38,39]. Deep fully convolutional neural networks (FCN) [40–45] have demonstrated its power in almost all computer vision tasks. Convolution operation is a local operation and the network with full convolutions are independent with the input size. People now are clear about the functioning of the different layers of the FCN. In computer vision task, the lower layers (layers near input) tend to generate sharing features of all objects like edges and curves while the higher layers (near output) are more object oriented. If we train the network using the loss from the lower layers, the texture and details are persevered, while the higher layers will keep the general spatial structure.

This motivates us using the losses from different layers of the fully convolutional layers. Multiple layers will give us a multilevel capture of the basis features and hence measure the basis in a more complete way. To implement the idea, we will pretrain an adversary net; and input the multiscale basis of the generative net and the real basis. The losses then come from some selected layers of the adversary net. Although it is still not clear the speciality of each layer, if we consider the multiscale physical problem, the experiments show that the accuracy is improved and, amazingly, the training becomes easier when compared to the MSE of the basis directly.

The uncertain space coarsening (cluster) is performed using the deep learning idea described above. Due to the space dimension, we will perform the clustering algorithm locally in space; that is, we first need a spatial coarsening. Due to the multiscale natural of the problem, this motivates us using the generalized multiscale finite element methods (GMsFEM) which derive the multiscale basis of a coarse neighborhood by solving the local problem. GMeFEM was first proposed in [46] and further studied in [3–10]. This method is a generalization of the multiscale finite element method [47,48]. The work starts from constructing the snapshot space for each local neighborhood. The snapshot space is constructed by solving local problems and several methods including harmonic extension, random boundary condition [49] have been proposed. Once we have the snapshot space, the offline space which will be used as computing the solution are constructed by using spectral decomposition.

Our method is designed for solving PDEs with heterogeneous properties and uncertainty. The heterogeneity and uncertainty in our models come from the permeability $\kappa(x,s)$. To verify our method, we numerically simulate around 240,000 local spatial fields which contain complex information such as the moving channels. Our model is then trained and tested based on the generated spatial fields. It should be noted that our method could be applied to some other realistic problems which contain large-scale data such as detecting extreme values with order statistics in samples from continuous distributions [50], as well as to some other subjects, e.g., multiscale social networks [2] and the multiscale public safety systems [1]. These topic will be studied in the future.

The rest of the work is organized as follow: in Section 2, we consider the problem setup and introduce both uncertain space and spatial coarsening. In Section 3, we introduce the structure of the network and the training algorithm. In Section 4, we will present the numerical results. The paper ends with conclusions.

2. Problem Settings

In this section, we will present some basic ideas involving the use of the generalized multiscale finite element method (GMsFEM) for parameter-dependent problems. Let D be a bounded domain in $\mathbb{R}^2$ and Ω be the parameter space in $\mathbb{R}^N$. We consider the following parameter-dependent elliptic problem:

$$-\nabla \cdot (\kappa(x,s)\nabla u(x,s)) = f(x,s), (x,s) \in D \times \Omega, \quad (1)$$

$$u(x,s) = 0, (x,s) \in \partial D \times \Omega, \quad (2)$$

where $\kappa(x,s)$ is a heterogeneous coefficient depending on both the spatial variable x and the parameter s, and $f \in L^2(D)$ is a given source. We remark that the differential operators in Equation (1) are defined with respect to the spatial variable x. This is the case for the rest of the paper.

2.1. The Coarsening of the Parameter Space. The Main Idea

The parameter space Ω is assumed to be of very high dimension (i.e., large N) and consists of very large number of realizations. For a given realization, the idea is to find its representation in the coarse space and use the coarse space to perform the computation. We will use the deep cluster learning algorithm to perform the coarsening. Due to the heterogeneous properties of the proposed problem, fine mesh is used; this will bring difficulties in coarsening the parameter space and in computation of the solution. We hence perform the parameter coarsening locally in the space D, i.e., we also coarsen the spatial domain. To coarsen the spatial domain, we use coarse grids and consider the GMsFEM.

In Figure 1, we present an illustration of the proposed coarsening technique. On the left figure, the coarse grid blocks in the space are shown. Each coarse grid has a different cluster in the uncertainty space Ω, which corresponds to the coarsening of the uncertainty space. The main objective in multiscale methods is efficiently finding the clustering of the uncertainty space, which is our main goal.

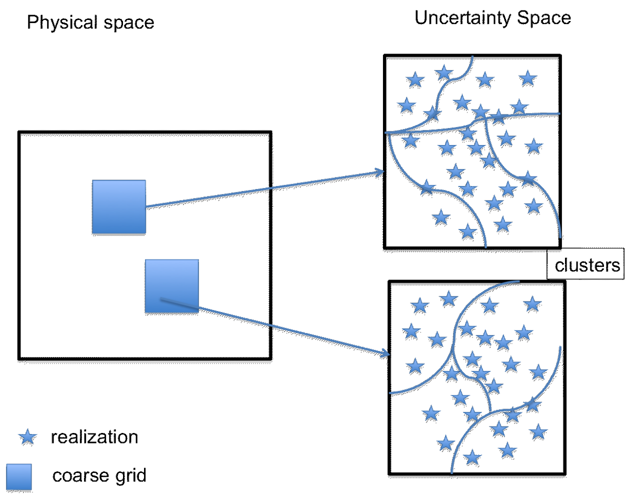

Figure 1. Illustration of coarsening of space and uncertainties. Different clusters for different coarse blocks. On the left plot, two coarse blocks are shown. On the right plot, clusters are illustrated.

2.2. Space Coarsening—Generalized Multiscale Finite Element Method

It is computationally expensive to capture heterogeneous properties using very fine mesh. For this reason, we use GMsFEM to coarsen the spatial representation of the solution. The coarsening of the parameter space will be performed in each local spatial neighborhood. We will achieve this goal by the GMsFEM, which will briefly be discussed. Consider the second order elliptic equation $Lu = f$ in D with proper boundary conditions; denote the the elliptic operator as:

$$L(u) = -\frac{\partial}{\partial x_i}(k_{ij}(x)\frac{\partial}{\partial x_j}u). \tag{3}$$

Let the spatial domain D be partitioned by a coarse grid $\mathcal{T}^H$; this does not resolve the multiscale features. Let us denote K as one cell in $\mathcal{T}^H$ and refine K to obtain the fine grid partition $\mathcal{T}^h$ (blue box in Figure 2). We assume the fine grid is a conforming refinement of the coarse grid. See Figure 2 for details.

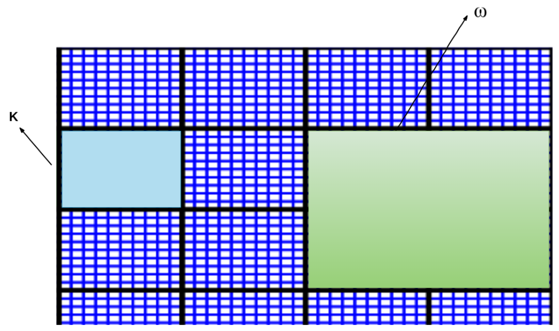

Figure 2. Domain Partition $\mathcal{T}^H$.

For the i-th coarse grid node, let ω_i be the set of all coarse elements having the vertex i (green region in Figure 2). We will solve local problem in each coarse neighborhood to obtain set of multiscale basis functions $\{\phi_i^{\omega_i}\}$ and seek solution in the form:

$$u = \sum_i \sum_j c_{ij} \phi_j^{\omega_i}, \tag{4}$$

where $\phi_j^{\omega_i}$ is the offline basis function in the i-th coarse neighborhood ω_i and j denotes the j-th basis function. Before we construct the offline basis, we first need to derive the snapshot basis.

2.2.1. Snapshot Space

There are several methods to construct the snapshot space; we will use the harmonic extension of the fine grid functions defined on the boundary of ω_i. Let us denote $\delta_l^h(x)$ as fine grid delta function, which is defined as $\delta_l^h(x_k) = \delta_{l,k}$ for $x_k \in J_h(\omega_i)$ where $J_h(\omega_i)$ denotes the boundary nodes of ω_i. The snapshot function $\psi_l^{\omega_i}$ is then calculated by solving local problem in ω_i:

$$L(\psi_l^{\omega_i}) = 0 \tag{5}$$

subject to the boundary condition $\psi_l^{\omega_i} = \delta_l^h(x)$. The snapshot space $V_{snap}^{\omega_i}$ is then constructed as the span of all snapshot functions.

2.2.2. Offline Spaces

The offline space $V_{off}^{\omega_i}$ is derived from the snapshot space and is used for computing the solution of the problem. We need to solve for a spetral problem and this can be summarized as finding λ and $v \in V_{snap}^{\omega_i}$ such that:

$$a_{\omega_i}(v, w) = \lambda s_{\omega_i}(v, w), \forall w \in V_{snap}^{\omega_i}, \tag{6}$$

where a_{ω_i} is symmetric non-negative definite bilinear form and s_{ω_i} is symmetric positive definite bilinear form. By convergence analysis, they are given by

$$a_{\omega_i}(v, w) = \int_{\omega_i} \kappa \nabla v \cdot \nabla w, \tag{7}$$

$$s_{\omega_i}(v, w) = \int_{\omega_i} \tilde{\kappa} v \cdot w. \tag{8}$$

In the above definition of s_{ω_i}, the function $\tilde{\kappa} = \kappa \sum |\nabla \chi_j|^2$ where $\{\chi_j\}$ is a set of partition of unity functions corresponding to the coarse grid partition of the domain D and the summation is taken over all the functions in this set. The offline space is then constructed by choosing the smallest L_i eigenvalues and we can form the space by the linear combination of snapshot basis using corresponding eigenvectors:

$$\phi_k^{\omega_i} = \sum_{j=1}^{L_i} \Psi_{kj}^{\omega_i} \psi_j^{\omega_i}, \tag{9}$$

where $\Psi_{kj}^{\omega_i}$ is the jth element of kth eigenvector and L_i is the number of snapshot basis. V_{off} is then defined as the collection of all local offline basis functions. Finally we are trying to find $u_{off} \in V_{off}$ such that

$$a(u_{off}, v) = \int_D fv, \forall v \in V_{off} \tag{10}$$

where $a(u,v) = \int_D \kappa \nabla u \cdot \nabla v$. For more details, we refer the readers to the references [8–10].

2.3. The Idea of the Proposed Method

We present the general methodology in this section. The target is to save the time in computing the GMsFEM basis $\phi_k^{\omega_i}$ for all ω_i and for all uncertain space parameters. We propose the clustering algorithm to coarsen the uncertain space in each local neighborhood. The key to the success of the clustering is that: the cluster should inherit the property of the solution, that is, the local heterogeneous fields $\kappa(x,s)$ clustered into the same group should have similar solution properties. When the cluster is learned by the some learning algorithm, the only computation involved is to fit the local neighborhood of the given testing heterogeneous field into some cluster. This is a feed forward process including several convolution operations and matrix multiplications and compared to the direct computing, we save a lot of time in computing the spectral problem in Equation (6) and the inverse of a matrix Equation (10). The detailed process is illustrated in the following chart (Figure 3):

1. (Training) For a given input local neighborhood ω_j, we train the cluster (which will be detailed in next section) of the parameter space Ω and get the clusters $S_1^j, ..., S_n^j$, where n is the number of clusters and is uniform for all j. Please note that we may have different cluster assignments in different local neighborhoods.
2. (Training) For each local neighborhood ω_j and cluster S_i^j, define the average $\bar{\kappa}_{ij}$ and compute generalized multiscale basis for $\bar{\kappa}_{ij}$.
3. (Testing) Given a new $\kappa(x,s)$ and for each local neighborhood ω_j, fit $\kappa(x,s)$ into a S_i^j by the trained network (step 1) and use the pre-computed GMsFEM basis (step 2) to find the solution.

It should be noted that we perform clustering using the heterogeneous fields; however, the cluster should inherit the property of the solution corresponding to the heterogeneous fields. This makes the clustering challenging. The performance of the standard K-means algorithm relies on the initialization and the distance metric. We may initialize the algorithm based on the clustering of the heterogeneous fields but we need to design a good metric. In the next section, we are going to introduce a learning algorithm which uses an auto-encoder structure and multiple losses to achieve the required clustering task.

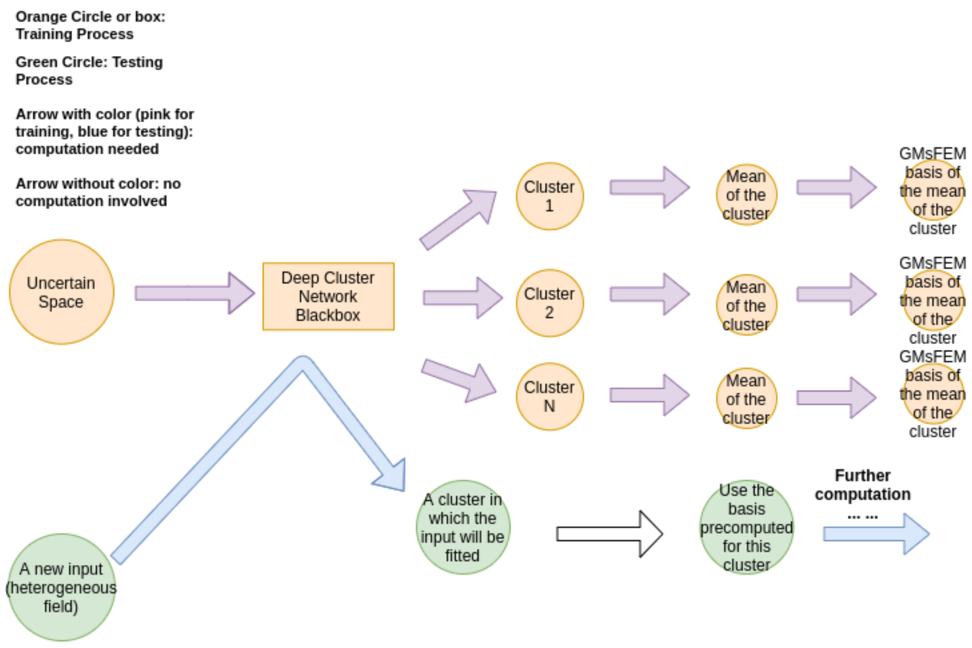
Orange Circle or box: Training Process
Green Circle: Testing Process
Arrow with color (pink for training, blue for testing): computation needed
Arrow without color: no computation involved
Uncertain Space
Deep Cluster Network Blackbox
Cluster 1
Cluster 2
Cluster N
Mean of the cluster
GMsFEM basis of the mean of the cluster
A new input (heterogeneous field)
A cluster in which the input will be fitted
Use the basis precomputed for this cluster
Further computation
... ...

Figure 3. Work flow of the proposed method.

3. Deep Learning

The network is consisted of two sub networks. The first one is targeted to performing the clustering and the second one, which is the adversary net, will serve as the reconstruction of loss function.

3.1. Clustering Net

The cluster net is aimed for clustering the heterogeneous fields $\kappa(x,s)$; but the resulting clusters should inherit the properties of the solution corresponding to the $\kappa(x,s)$, i.e., the heterogeneous fields grouped in the same cluster should have similar corresponding solution properties. This similarity will be measured by the adversary net which will be introduced in Section 3.3. We hence design the network demonstrated in Figure 4.

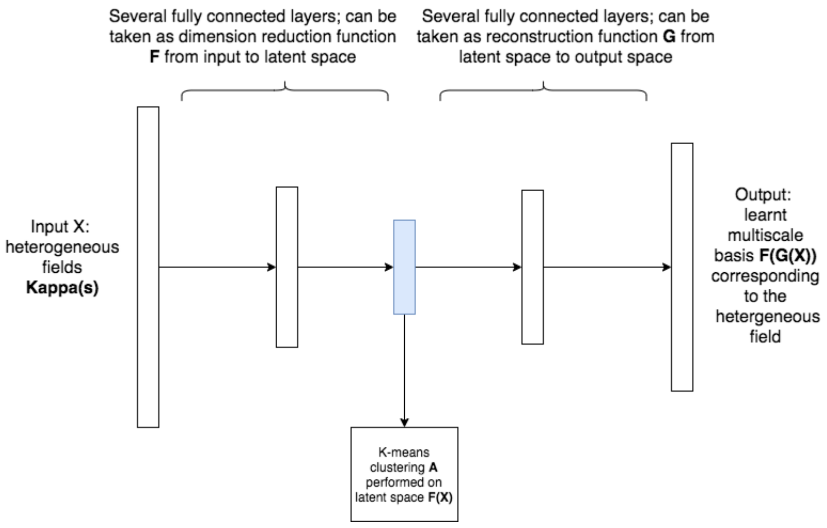
Several fully connected layers; can be taken as dimension reduction function F from input to latent space
Several fully connected layers; can be taken as reconstruction function G from latent space to output space
Input X: heterogeneous fields Kappa(s)
Output: learnt multiscale basis F(G(X)) corresponding to the hetergeneous field
K-means clustering A performed on latent space F(X)

Figure 4. Cluster network.

The input $X \in \mathbb{R}^{m,d}$, where m is the number of samples and d is the dimension of one local heterogeneous field, of the network is the local heterogeneous fields which are parametrized by the random variable $s \in \Omega$. The output of the network is the multiscale basis (first GMsFEM basis) which somehow represents the solution corresponding to the coefficient $\kappa(x,s)$. This is a generative network which has an auto encoder structure. The dimension reduction function $F(X)$ can be interpreted as some kind of kernel method which maps the input data to a new space which is easier to be separated. This can also be interpreted as the learning of a good metric function for the later performed K-mean clustering. We will perform K-means clustering algorithm in latent space $F(X)$. $G(\cdot)$ will then transfer the latent space data to the space of multiscale basis function. This process can be taken as a generative process and we reconstruct the basis from the extracted features. The detailed algorithm is as follow (see Figure 5 for an illustration):

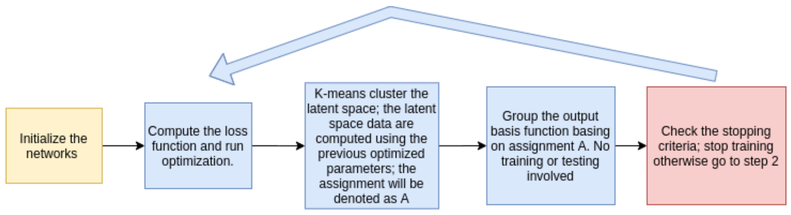
Initialize the networks
Compute the loss function and run optimization.
K-means cluster the latent space; the latent space data are computed using the previous optimized parameters; the assignment will be denoted as A
Group the output basis function basing on assignment A. No training or testing involved
Check the stopping criteria; stop training otherwise go to step 2

Figure 5. Deep learning algorithm.

Steps illustrated in Figure 5:

1. Initialize the networks and clustering the output basis function.
2. Compute the loss function L (defined later) and run optimization.
3. Cluster the latent space by K-means algorithm (reduced dimension space, which is a middle layer of the cluster network); the latent space data are computed using the previous optimized parameters; the assignment will be denoted as A.
4. Basis functions whose corresponding inputs are in the same cluster (basing on assignment A) will be grouped together. No training or fitting-in involved in this step.
5. Repeat step 2 to step 4 until the stopping criteria is met.

3.2. Loss Functions

Loss function is the key to the deep learning. Our loss function is consisted of cluster loss and the reconstruction loss.

1. Clustering loss $C(\theta_F, \theta_G)$: this is the mean standard deviation of all clusters of the learned basis and θ is the parameters we need to optimize. It should be noted that the loss here is computed using the learned basis instead of the input of the network. This loss controls the clustering process, i.e., the smaller the loss, the better the clustering in the sense of clustering the multiscale basis. Let us denote κ_{ij} as jth realization in ith cluster; $G(F(\kappa_{ij})) \in \mathbb{R}^d$ will then be jth learned basis in cluster i and let θ_G and θ_F be the parameters associated with G and F, the loss is then defined as follow,

$$C(\theta_F, \theta_G) = \frac{1}{|A|}\sum_i^{|A|}\sum_j^{A_i}\frac{1}{A_i}\|G(F(\kappa_{ij})) - \bar{\phi}_i\|_2^2, \tag{11}$$

where $|A|$ is the number of clusters which is a hyper parameter and A_i denotes the number of elements in cluster i; $\bar{\phi}_i \in \mathbb{R}^d$ is the mean of cluster i. This loss clearly serves the purpose of clustering the solution instead of the input heterogeneous fields; however, in order to guarantee the learned basis are closed to the pre-computed multiscale basis, we need to define the reconstruction loss which measures the difference between the learned basis and the pre-computed basis.
2. Reconstruction loss $R(\theta_F, \theta_G)$: this is the mean square error of multiscale basis $Y \in \mathbb{R}^{m,d}$, where m is the number of samples. This loss controls the construction process, i.e., if the loss is small,

the learned basis are close to the real multiscale basis. This loss will supervise the learning of the cluster. It is defined as follow:

$$R(\theta_F, \theta_G) = \frac{1}{m}\sum_{i}^{m} \|G(F(\kappa_i)) - \phi_i\|_2^2, \tag{12}$$

where $G(F(\kappa_i)) \in \mathbb{R}^d$ and $\phi_i \in \mathbb{R}^d$ are learned and pre-computed multiscale basis of ith sample κ_i separately.

The entire loss function is then defined as $L(\theta_F, \theta_G) = \lambda_1 C + \lambda_2 R$, where λ_1, λ_2 are predefined weights. We are going to solve the following optimization problem:

$$\min_{\theta_G, \theta_F} L(\theta_F, \theta_G) \tag{13}$$

for the required training process.

3.3. Adversary Network Severing as an Additional Loss

We have introduced the reconstruction loss which measures the similarity between the learned basis and the pre-computed basis in the previous section. It is the mean square error (MSE) of the learned and pre-computed basis. MSE is a smooth loss and easy to train but there is a well known fact about MSE that this loss will blur the image. In the area of image super-resolution and other low level computer vision tasks, the loss is not friendly to inputs with high contrast and the resulting generated images are usually over smooth. Our problem has multiscale nature and is similar with the low level computer vision task, i.e., this is a generative task; hence blurring and over smoothing should happen if the model is trained by MSE. To define a great reconstruction loss is important.

Motivated by some works about the successful application of deep fully convolutional network (FCN) in computer vision, we design a perceptual loss to measure the error. It is now clear that the lower layers in the FCN usually will extract some general features shared by all objects like the horizontal (vertical) curves, while the higher layers are usually more objects oriented. This gives people the insight to train the network using different layers. Johnson then proposed the perceptual loss [29] which is the combination of the MSE of selected layers of the VGG model [51]. The authors claim in their paper that the early layers tends to produce images that are visually indistinguishable from the input; however if reconstruct from higher layers, image content and overall spatial structure are preserved but color, texture, and exact shape are not.

We will adopt the perceptual loss idea and design an adversary network to compute an additional reconstruction loss. The network structure can be seen in Figure 6.

The adversary net is fully convolutional with input and output both pre-computed multiscale basis. The network has an auto encoder structure and is pre-trained; i.e., we are going to solve the following minimization problem:

$$\min_{\theta_A} \frac{1}{m}\sum_{i} \|f(\phi_i) - \phi_i)\|_2^2, \tag{14}$$

where ϕ_i is the multiscale basis and f is the adversary net associated with trainable parameter θ_A. Denote $f_j(\cdot)$ as the output of layer j of the adversary network. The additional reconstruction loss is then redefined as:

$$A(\theta_F, \theta_G) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j\in I} \|f_j(G(F(\kappa_i))) - f_j(\phi_i)\|_2^2, \tag{15}$$

where I is the index set which contains some layers of the adversary net. The complete optimization problem can be now formulated as:

$$\min_{\theta_G,\theta_F} \lambda_1 C + \lambda_2 R + \lambda_3 A. \tag{16}$$

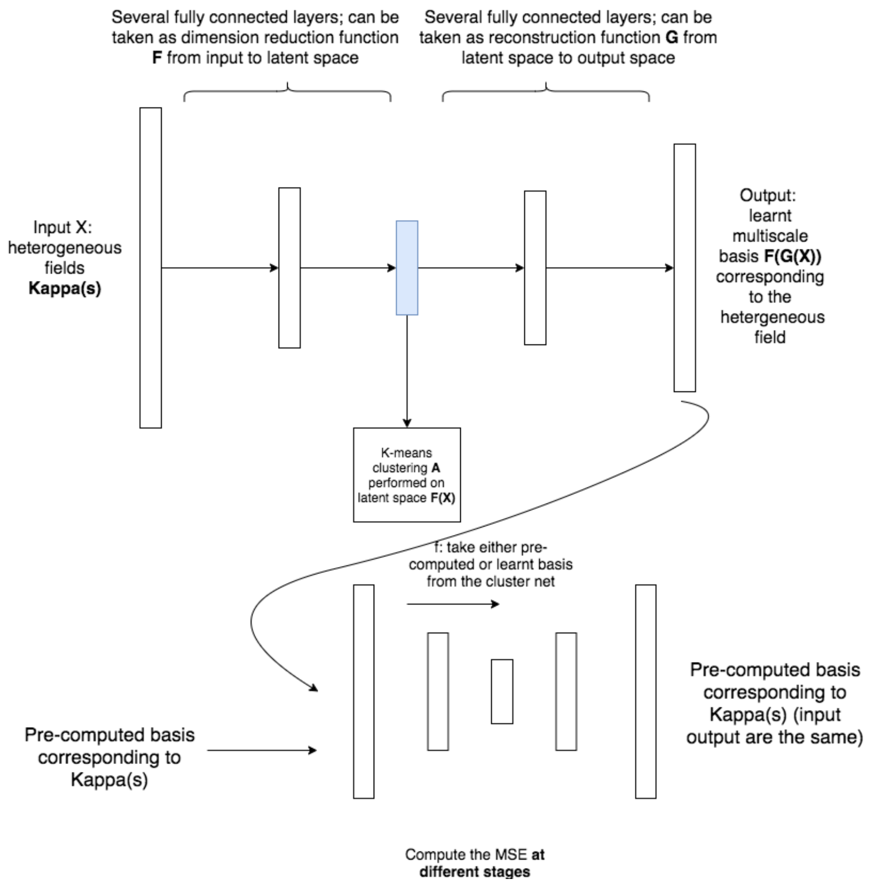
Several fully connected layers; can be taken as dimension reduction function F from input to latent space
Several fully connected layers; can be taken as reconstruction function G from latent space to output space
Input X: heterogeneous fields Kappa(s)
Output: learnt multiscale basis F(G(X)) corresponding to the hetergeneous field
K-means clustering A performed on latent space F(X)
f: take either pre-computed or learnt basis from the cluster net
Pre-computed basis corresponding to Kappa(s)
Pre-computed basis corresponding to Kappa(s) (input output are the same)
Compute the MSE at different stages

Figure 6. The complete network.

4. Numerical Experiments

In this section, we will demonstrate a series of experiments. We are going to apply our method on problems with high contrast including moving background and moving channels. he experiments are related to subsurface simulations. The moving background and moving channels simulate some important characteristics in the field. We numerically generate heterogeneous fields which contain moving channels and varying well rates. In Section 4.1, we first demonstrate a set of simulated heterogeneous oil fields to be used to train and solve the PDE modeling the reservoirs simulation; the deep learning model settings are also detailed in this section. In Section 4.2, we simulate some other more complicated heterogeneous fields and conduct the experiments to demonstrate the power of clustering algorithm. This experiments can show that our method is robust to handle complicated problems. In the last section, we will solve the PDE using the proposed method based on the heterogeneous field proposed in Section 4.1 and compute the relative error to demonstrate the accuracy of our method.

4.1. High Contrast Heterogeneous Fields with Moving Channels

We consider solving Equations (1)–(2) for a heterogeneous field with moving channels and changing background. Let us denote the heterogeneous field as $\kappa(x)$, where $x \in [0,1]^2$, then $\kappa(x) = 1000$ if x is in some channels which will be illustrated later and otherwise,

$$\kappa(x) = e^{\eta \cdot sin(7\pi x)sin(8\pi y)+sin(10\pi x)sin(12\pi y)},$$

where η follows discrete uniform distribution in $[0,1]$. The channels are moving and we include cases of the intersection of two channels and formation and dissipation of the channels in the fields. These simulate the realistic petroleum oil fields. In Figure 7, we demonstrate 20 heterogeneous fields.

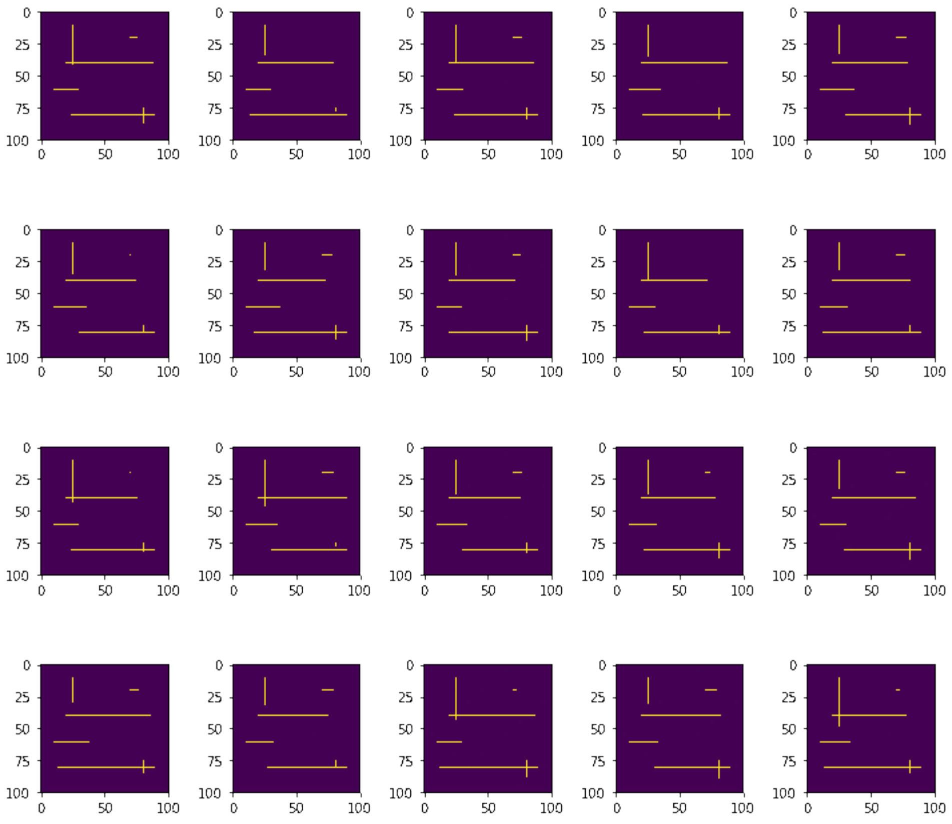

Figure 7. Heterogeneous fields, the yellow strips are the channels.

It can be observed from the images that, vertical channel (at around $x = 30$) (not always) intersects with horizontal channels (at around $y = 40$); and the channel at $x = 75, y = 25$ demonstrates the case of generation and degeneration of a channel.

We train the network using 600 samples using the Adam gradient descent. We find that the cluster assignment of 600 realizations in uncertain space is stable(fixed) when the gradient descent epoch reaches a certain number, so we set the stopping criteria to be: the assignment does not change for 100 iteration epochs; and the maximum number of iteration epochs is set to be 1000. We also find that the coefficients in Equation (16) can affect the training result. We set $\lambda_1 = \lambda_2 = \lambda_3 = 1$.

It should be noted that we train the network locally in each coarse neighborhood. The fine mesh element has size $1/100$ and 5 fine elements are merged into one coarse element.

The dimension reduction network F contains 4 fully connected layers to reduce the size of local coarse elements from 100 to $60, 40, 30, 20$ gradually. The K-means clustering is conducted in space $F(x)$ of dimension 20; the reconstruction net G is designed symmetrically with the reduction network F. The adversary net is fully convolutional. All convolution layers except the last layer have kernels of size 3 by 3 with stride 1; we use 1 by 1 convolution in the last layer to reduce the number of channels to 1. The number of channels is doubled if the spatial dimension is reduced and half-ed if the spatial dimension is increased. Max pooling of size 2 by 2 is used to reduce the spatial dimension in the encoder; and to increase the dimension in the decoder, we perform the nearest neighbor resize followed by convolution [52].

4.2. Results

We will present the numerical results of the proposed method in this section. We are going to show the cluster assignment experiment first, followed by two other experiments which demonstrate the error of the method.

4.2.1. Cluster Assignment in a Local Coarse Element

Before diving into the error analysis, we will show some of the cluster results in a local neighborhood. In this neighborhood, we manually created the cases such as: the extraction of a channel (longer), the expansion of a channel(wider), the discontinuity of a channel, the diagonal channels, the intersection of channels, and so on. In Figure 8, the number on top of each image is the cluster assignment ID number.

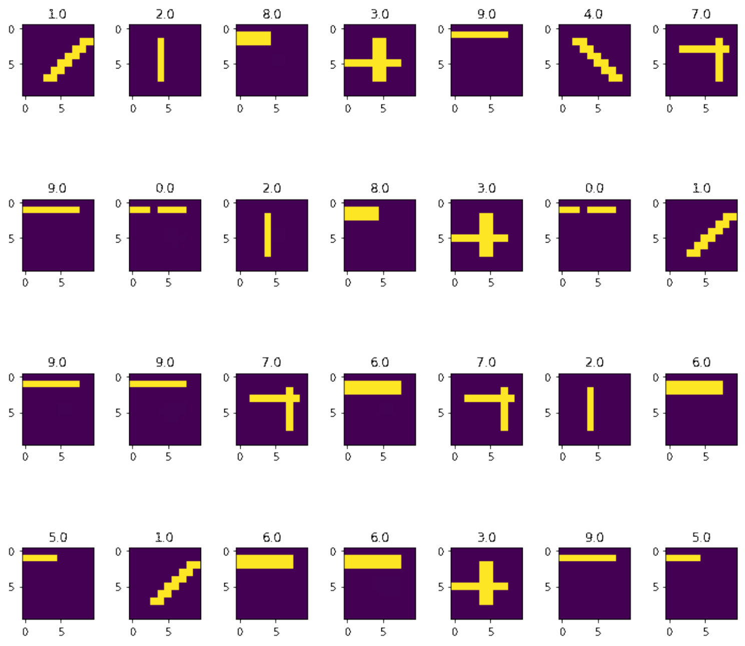

Figure 8. Cluster results of 28 samples, images shown are heterogeneous fields, the number on top of each image is the cluster assignment ID number.

We also demonstrate the clustering result in Figure 9 of another neighborhood which is around $(25, 45)$ in Figure 7. From the results in both Figures 8 and 9, we observe that our proposed clustering algorithm based on deep learning is able create a good clustering of the parameter space. That is, heterogeneous coefficients with similar spatial structures are grouped in the same cluster.

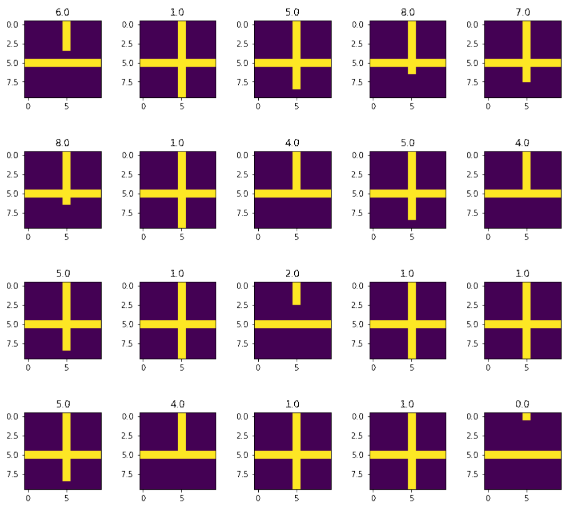

Figure 9. Cluster results of 20 samples, images shown are heterogeneous fields, the number on top of each image is the cluster assignment ID number.

4.2.2. Relation of Error and the Number of Clusters

In this section, we will demonstrate the error change when the hyperparamter—the number of clusters—increases. Given a new realization $\kappa(x, \hat{s})$ where $\hat{s}$ denotes the parameter and a fixed neighborhood, suppose the neighborhood of this realization will be fitted into cluster S_i by the model trained. We compute $\bar{\kappa}_i = \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} \kappa_{ij}$ where $|S_i|$ is the number of points in this cluster S_i. The GMsFEM basis of this neighborhood can then be derived using $\bar{\kappa}_i$. We finally construct the solution using the GMsFEM basis pre-computed in all neighborhoods. We define the l_2 relative error as :

$$ratio = \frac{\int_D (u - u_H)^2 dx}{\int_D u^2 dx}, \tag{17}$$

where u is the exact solution computed by finite element method with fine enough mesh and u_H is the solution of the proposed method. We test the network on newly generated 300 samples and take the average of the errors.

In this experiment, we calculate the l_2 relative error with the number of clusters increases. The number of clusters ranges from 5 to 11; and for each case, we will train the model and compute the l_2 relative error. The result can be seen in Figure 10 and it can be observed from the picture that, the error is decreasing with the number of cluster increases.

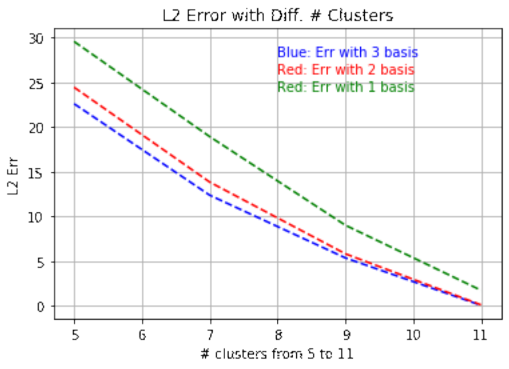

Figure 10. The l_2 error when the number of clusters changes, colors represent the number of GMsFEM basis.

4.2.3. Comparison of Cluster-based Method with Tradition Method

In the second experiments, we first compute the l_2 relative error (defined in Equation (17) with u_H denoting the GMsFEM solution) of traditional GMsFEM method with given $\kappa(x, \hat{s})$. This means that the construct multiscale basis functions using the particular realization $\kappa(x, \hat{s})$. We then compare this error with the cluster method proposed (11 clusters). The comparison can be seen in Figure 11.

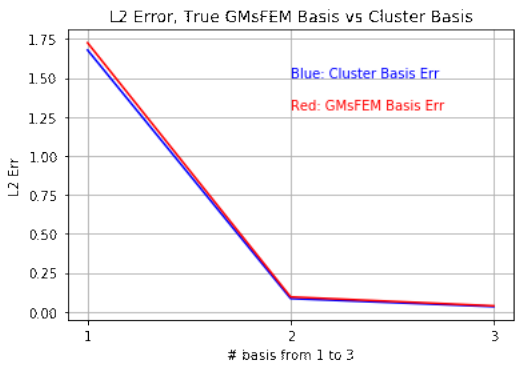

Figure 11. The l_2 error cluster solution (11 clusters) vs. solution by real $\kappa(x, \hat{s})$. Color represents number of basis.

It can be seen that the difference is negligible when the number of clusters reaches 11. We can then benefit from the deep learning; i.e., the fitting of $\kappa(x, \hat{s})$ into a cluster is fast; and since we will use the pre-computed basis, we also save time on computing the GMsFEM basis.

4.3. *Effect of the Adversary Net*

The target of this task is not the learning of multiscale basis; the multiscale basis in this work is just a supervision of learning the cluster. However, to demonstrate the effectiveness of the adversary network, we also test the the effect of the adversary net. There are many hyper-parameters like the number of clusters and coefficients of the loss function which can affect the result; so to reduce the influence from the clustering, we remove the clustering loss from the training, so this is a generative task which will generate the multiscale basis from the output of the first network in Figure 6. The loss function now can be defined as:

$$\min_{\theta_G, \theta_F} \lambda_1 R + \lambda_2 A, \tag{18}$$

where R and A are defined in Equations (12) and (15), separately; and λ_1 and λ_1 are both set to be 1. We compute the relative error with Equation (17) first by using the learned multiscale basis which is trained by Equation (18); and second by using the multiscale basis trained without the adversary loss Equation (15), i.e.,

$$\min_{\theta_G, \theta_F} A. \tag{19}$$

The l_2 relative error improves from 41.120439 to 36.760918 if we add one middle layer from the adversary net.

We also calculate the MSE difference of two learned basis (by loss Equation (18) and Equation (19), separately) and real multiscale basis, i.e., we calculate $\|B_{\text{learned basis}} - B_{\text{real basis}}\|_{MSE}$, where $B_{\text{learned basis}}$ refers to two basis trained with Equation (18) and Equation (19), separately and $B_{\text{real basis}}$ is the real multiscale basis formed using the input heterogeneous field. The MSE amazingly decreases from 0.9073400 to 0.748312 if we use basis trained with the adversary loss Equation (18). This can show the benefit from the adversary net.

5. Conclusions

We propose a deep learning clustering technique within GMsFEM to solve flows in heterogeneous media. The main idea is to cluster the uncertainty space such that we can reduce the number of multiscale basis functions for each coarse block across the uncertainty space. We propose the adversary loss motivated by the perceptual loss in the computer vision task. We use convolutional neural networks combined with some techniques in adversary neural networks, where the loss function is composed of several parts that includes terms related to clusters and reconstruction of basis functions. We present numerical results for channelized permeability fields in the examples of flows in porous media. In future, we would like to study the relation between convolutional layers and quantities related to multiscale basis functions. In addition, we are going to study the application of our method in the area of multiscale social network and other studies like extreme value prediction.

Author Contributions: All authors have contributed to methodology and validation. Simulations are performed by Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: Eric Chung's work is partially supported by the Hong Kong RGC General Research Fund (Project numbers 14304719 and 14302018) and the CUHK Faculty of Science Direct Grant 2018-19. Y.E. would like to thank the partial support from NSF 1620318 and NSF Tripod 1934904. Y.E. would also like to acknowledge the support of Mega-grant of the Russian Federation Government (N 14.Y26.31.0013).

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

References

1. Tsiropoulou, E.; Koukas, K.; Papavassiliou, S. A socio-physical and mobility-aware coalition formation mechanism in public safety networks. *EAI Endorsed Trans. Future Internet* **2018**, *4*, 154176. [CrossRef]
2. Thai, M.T.; Wu, W.; Xiong, H. *Big Data in Complex and Social Networks*; CRC Press: Boca Raton, FL, USA, 2016.
3. Calo, V.M.; Efendiev, Y.; Galvis, J.; Ghommem, M. Multiscale empirical interpolation for solving nonlinear PDEs. *J. Comput. Phys.* **2014**, *278*, 204–220. [CrossRef]
4. Chung, E.T.; Efendiev, Y.; Leung, W.T. Residual-driven online generalized multiscale finite element methods. *J. Comput. Phys.* **2015**, *302*, 176–190. [CrossRef]
5. Chung, E.T.; Efendiev, Y.; Leung, W.T. An online generalized multiscale discontinuous Galerkin method (GMsDGM) for flows in heterogeneous media. *Commun. Comput. Phys.* **2017**, *21*, 401–422. [CrossRef]
6. Chung, E.T.; Efendiev, Y.; Li, G. An adaptive GMsFEM for high-contrast flow problems. *J. Comput. Phys.* **2014**, *273*, 54–76. [CrossRef]
7. Chung, E.T.; Efendiev, Y.; Li, G.; Vasilyeva, M. Generalized multiscale finite element methods for problems in perforated heterogeneous domains. *Appl. Anal.* **2016**, *95*, 2254–2279. [CrossRef]
8. Efendiev, Y.; Galvis, J.; Lazarov, R.; Moon, M.; Sarkis, M. Generalized multiscale finite element method. Symmetric interior penalty coupling. *J. Comput. Phys.* **2013**, *255*, 1–15. [CrossRef]
9. Efendiev, Y.; Galvis, J.; Li, G.; Presho, M. Generalized multiscale finite element methods: Oversampling strategies. *Int. J. Multiscale Comput. Eng.* **2014**, *12*, 465–484. [CrossRef]
10. 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]
11. Chung, E.; Efendiev, Y.; Fu, S. Generalized multiscale finite element method for elasticity equations. *Int. J. Geomath.* **2014**, *5*, 225–254. [CrossRef]
12. Chung, E.; Vasilyeva, M.; Wang, Y. A conservative local multiscale model reduction technique for Stokes flows in heterogeneous perforated domains. *J. Comput. Appl. Math.* **2017**, *321*, 389–405. [CrossRef]
13. Chung, E.T.; Efendiev, Y.; Leung, W.T.; Zhang, Z. Cluster-based generalized multiscale finite element method for elliptic PDEs with random coefficients. *J. Comput. Phys.* **2018**, *371*, 606–617. [CrossRef]
14. Karhunen, K. *Über Lineare Methoden in der Wahrscheinlichkeitsrechnung*; Suomalainen Tiedeakatemia: Helsinki, Finland 1947; Volume 37.
15. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
16. Wang, Y.; Cheung, S.W.; Chung, E.T.; Efendiev, Y.; Wang, M. Deep multiscale model learning. *J. Comput. Phys.* **2020**, *406*, 109071. [CrossRef]

17. Wang, M.; Cheung, S.W.; Leung, W.T.; Chung, E.T.; Efendiev, Y.; Wheeler, M. Reduced-order deep learning for flow dynamics. The interplay between deep learning and model reduction. *J. Comput. Phys.* **2020**, *401*, 108939. [CrossRef]
18. Vasilyeva, M.; Leung, W.T.; Chung, E.T.; Efendiev, Y.; Wheeler, M. Learning macroscopic parameters in nonlinear multiscale simulations using nonlocal multicontinua upscaling techniques. *arXiv* **2019**, arXiv:1907.02921.
19. Cheung, S.W.; Chung, E.T.; Efendiev, Y.; Gildin, E.; Wang, Y.; Zhang, J. Deep global model reduction learning in porous media flow simulation. *Comput. Geosci.* **2020**, *24*, 261–274. [CrossRef]
20. Wang, M.; Cheung, S.W.; Chung, E.T.; Efendiev, Y.; Leung, W.T.; Wang, Y. Prediction of discretization of gmsfem using deep learning. *Mathematics* **2019**, *7*, 412. [CrossRef]
21. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
22. Caron, M.; Bojanowski, P.; Joulin, A.; Douze, M. Deep clustering for unsupervised learning of visual features. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 132–149.
23. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised deep embedding for clustering analysis. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 478–487.
24. Yang, B.; Fu, X.; Sidiropoulos, N.D.; Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR.org, Sydney, Australia, 6–11 August 2017; pp. 3861–3870.
25. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin, Germany, 2006.
26. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1125–1134.
27. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4681–4690.
28. Lai, W.S.; Huang, J.B.; Ahuja, N.; Yang, M.H. Deep laplacian pyramid networks for fast and accurate super-resolution. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 624–632.
29. Johnson, J.; Alahi, A.; Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin, Germany, 2016; pp. 694–711.
30. Dong, C.; Loy, C.C.; He, K.; Tang, X. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 295–307. [CrossRef]
31. Kim, J.; Kwon Lee, J.; Mu Lee, K. Accurate image super-resolution using very deep convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 1646–1654.
32. Zhang, Y.; Li, K.; Li, K.; Wang, L.; Zhong, B.; Fu, Y. Image super-resolution using very deep residual channel attention networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 286–301.
33. Zhang, Y.; Tian, Y.; Kong, Y.; Zhong, B.; Fu, Y. Residual dense network for image super-resolution. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2472–2481.
34. Tai, Y.; Yang, J.; Liu, X. Image super-resolution via deep recursive residual network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3147–3155.
35. Lim, B.; Son, S.; Kim, H.; Nah, S.; Mu Lee, K. Enhanced deep residual networks for single image super-resolution. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 136–144.
36. Tai, Y.; Yang, J.; Liu, X.; Xu, C. Memnet: A persistent memory network for image restoration. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 4539–4547.

37. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
38. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
39. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. *arXiv* **2018**, arXiv:1805.08318.
40. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA; 7–12 June 2015; pp. 3431–3440.
41. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
42. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2818–2826.
43. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
44. Noh, H.; Hong, S.; Han, B. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 11–18 December 2015; pp. 1520–1528.
45. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
46. Efendiev, Y.; Galvis, J.; Hou, T.Y. Generalized multiscale finite element methods (GMsFEM). *J. Comput. Phys.* **2013**, *251*, 116–135. [CrossRef]
47. Hou, T.Y.; Wu, X.H. A multiscale finite element method for elliptic problems in composite materials and porous media. *J. Comput. Phys.* **1997**, *134*, 169–189. [CrossRef]
48. Jenny, P.; Lee, S.; Tchelepi, H.A. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *J. Comput. Phys.* **2003**, *187*, 47–67. [CrossRef]
49. Calo, V.M.; Efendiev, Y.; Galvis, J.; Li, G. Randomized oversampling for generalized multiscale finite element methods. *Multiscale Model. Simul.* **2016**, *14*, 482–501. [CrossRef]
50. Vidal-Jordana, A.; Montalban, X. Multiple sclerosis: Epidemiologic, clinical, and therapeutic aspects. *Neuroimaging Clin.* **2017**, *27*, 195–204. [CrossRef]
51. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
52. Odena, A.; Dumoulin, V.; Olah, C. Deconvolution and checkerboard artifacts. *Distill* **2016**, *1*, e3. [CrossRef]