Hindawi
Geofluids
Volume 2020, Article ID 6641642, 22 pages
https://doi.org/10.1155/2020/6641642

Research Article

Reconstruction of Three-Dimensional Porous Media Using Deep Transfer Learning

Yi Du,[1] Jie Chen,[2] and Ting Zhang[2]

[1]*College of Engineering, Shanghai Polytechnic University, Shanghai 201209, China*
[2]*College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China*

Correspondence should be addressed to Ting Zhang; tingzh@shiep.edu.cn

Received 26 October 2020; Revised 18 November 2020; Accepted 9 December 2020; Published 30 December 2020

Academic Editor: Zhiming Chen

The reconstruction of porous media is widely used in the study of fluid flows and engineering sciences. Some traditional reconstruction methods for porous media use the features extracted from real natural porous media and copy them to realize reconstructions. Currently, as one of the important branches of machine learning methods, the deep transfer learning (DTL) method has shown good performance in extracting features and transferring them to the predicted objects, which can be used for the reconstruction of porous media. Hence, a method for reconstructing porous media is presented by applying DTL to extract features from a training image (TI) of porous media to replace the process of scanning a TI for different patterns as in multiple-point statistical methods. The deep neural network is practically used to extract the complex features from the TI of porous media, and then, a reconstructed result can be obtained by transfer learning through copying these features. The proposed method was evaluated on shale and sandstone samples by comparing multiple-point connectivity functions, variogram curves, permeability, porosity, etc. The experimental results show that the proposed method is of high efficiency while preserving similar features with the target image, shortening reconstruction time, and reducing the burdens on CPU.

1. Introduction

The reconstruction of porous media plays a key role in many engineering disciplines. A model of porous media can be used to quantitatively study the effects of various microscopic factors (e.g., pore structures, wettability, and aqueous films) on the macroscopic properties of oil and gas reservoirs, showing its great significance for the study of the seepage mechanisms of oil and gas [1–3].

Different approaches are used to model the internal structures and features of porous media such as geological facies and petrophysical properties. Some typical physical experimental methods including the Serial Sections Tomography Method (SSTM) and the X-ray Computer Tomography Scanning Method (XRCTSM) [4–7] use experimental instruments to scan a sample of porous media to obtain a large number of two-dimensional cross-section images and then superimpose these images by a modeling program or software to form three-dimensional digital porous media. The SSTM is quite time-consuming, and the connectivity between slices sometimes is not satisfactory. The XRCTSM has the advantage of being fast and precise, but the experiments are very expensive. Generally speaking, physical methods (e.g., SSTM and XRCTSM) can obtain high-resolution real images of porous media but are constrained by high costs of equipment or experimental difficulties.

Different from physical experimental methods, numerical reconstruction methods such as the cross-correlation-based simulation (CCSIM) [8], the Sequential Indicator Simulation Method (SISM) [9], and the process-based method [10] are often based on a small number of two- or three-dimensional real images and reconstruct the three-dimensional porous media by stochastic simulation or sedimentary process simulation. Compared with physical experimental methods,

numerical methods are cost-effective and can reconstruct many kinds of porous media, but some of them are still quite time-consuming and cannot do well in the reconstruction of porous media with complex inner structures like shale [3].

Multiple-point statistics (MPS) [11, 12] is also considered a typical numerical method for the reconstruction of porous media, calculating the conditional cumulative probability function by replacing the variogram with the training images (TIs) using a sequential simulation process. However, the whole simulation is quite slow and memory-demanding, and every time, the probabilistic information from TIs will be scanned over repeatedly for a new simulation because the probabilistic information is stored in memory instead of a file, leading to a great waste of both time and hardware resources, e.g., single normal equation simulation (SNESIM) [12], pattern-based simulation (SIMPAT) [13], filter-based simulation (FILTERSIM) [14], distance-based pattern simulation (DISPAT) [15], and direct sampling [16, 17].

Currently, it is possible to find some new solutions from flourishing machine learning especially deep learning and transfer learning to achieve better reconstruction of porous media for efficiency and quality. Deep learning is an algorithm that extracts complex features by performing multiple nonlinear transformations on data through multiple layers and neural networks [18]. Neural networks originated in the 1950s, which were called perceptron at that time. A neural network has altogether three layers: an input layer, an output layer, and a hidden layer. The input feature vectors connect the output layer through the hidden layer, and the result is obtained at the output layer. The single-layer perceptron is based on the linear combination of the input vectors, and the result is calculated by a nonlinear function. As shown in Figure 1, the first layer of a perceptron is regarded as a linear combination of input vectors $x = (x_1, x_2, \cdots, x_k)$. The output a can be used as the input of the next-level network after a nonlinear transformation by an activation function $f(\cdot)$. Define the weight $w = (w_1, w_2, \cdots, w_k)$ and the bias b. If there are multiple single-layer perceptrons connected as multiple layers, a fully connected and discrete network called an artificial neural network (ANN) is obtained.

In 2006, Hinton and Salakhutdinov [18] proposed an effective way of using the restricted Boltzmann machines (RBMs) to learn features from datasets, establishing the framework of deep learning. As an algorithm based on learning data and characterizing inner features, deep learning uses many simple nonlinear features to transform raw data into more complex, higher-level, and more abstract representations [19]. Compared with the traditional ANN, deep learning has more hierarchical layers, so it has a stronger ability to abstract complex features.

Deep learning has also developed into a number of related branches, such as Convolutional Neural Networks (CNNs) [20, 21] and Generative Adversarial Networks (GANs) [22]. A CNN is a multilayer perceptron specially designed to recognize two-dimensional shapes. It uses spatial relationships to reduce the number of parameters that need to be learned to improve the general training performance of the back propagation (BP) algorithm. A GAN trains two deep networks (generators and discriminators) and then lets them perform adversarial

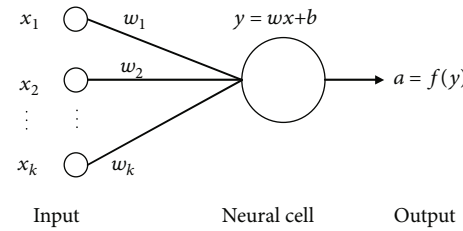


Figure 1: The structure of a perceptron.

learning, improving the capabilities of the generators and discriminators in continuous adversarial learning. GANs have been used for the reconstruction of porous media by using a discriminator to learn the TIs and a generator to reconstruct a new image of porous media [2].

In many machine learning algorithms, an important assumption is that the current training data and future training data must be in the same feature space and have the same distribution. However, in many realistic cases, this assumption may fail. Transfer learning is proposed to address this problem by using different transformations to bring the distances of different data distribution closer, extracting information from one or more source tasks, and then applying it to the target task. It is not necessary to retrain the TIs for every new reconstruction or simulation when using transfer learning, so it can save much time and reduce the necessary amount of training data [23].

Many different transfer learning methods have been developed. The distribution adaptation methods including Transfer Component Analysis (TCA) [24] and Deep Domain Confusion (DDC) [25] are typical transfer learning methods. The joint distribution adaptation method [26] improves transfer learning by reducing the distance between the joint probability distribution of the source and the target domains. Deep transfer learning (DTL) [27, 28] is based on transfer learning and currently is one of the most important technologies in deep learning, which can quickly transfer well-trained models from one dataset to another [29]. The network of DTL can simulate a new dataset well by borrowing the features from previously trained datasets [30].

This paper proposes a reconstruction method of porous media based on DTL, which uses deep learning to learn the features of porous media and then copies the learned features into the reconstructed results by transfer learning. Specifically, the method designs a deep learning model to learn the features of TIs and then to reconstruct porous media by transferring the features learned from TIs through a deep neural network (DNN). The reconstructed results have similar features (e.g., pore structures, connectivity, and permeability) hidden in the TIs.

There are two important tasks in the proposed method. The first one is to determine the proper parameters to learn TIs when training DNNs to achieve high accuracy. The second one is to transfer the features learned by DNNs into a new reconstruction. For the first task, the appropriate network structures (e.g., hidden layers and neural cells) and the optimization methods (e.g., the activation function and gradient descend algorithms) as well as some ways of tuning

for them will be considered. The second one can be solved by improving DTL and choosing appropriate transfer conditions.

The processing time of the proposed method can be largely shortened due to the GPU acceleration and optimization algorithms for deep learning compared to traditional simulation methods such as MPS. Experiments show that DTL can reconstruct the similar structures of porous media with the target images. Meanwhile, once the parameters for the reconstruction can be determined, they are stored after training and can be quickly used for new reconstructions, displaying the effectiveness of the proposed method in reconstruction quality, speed, and memory demands.

2. The Main Idea of the Proposed Method

In the proposed method, deep learning is used to learn the complex features of porous media, and then, these features are transferred into reconstruction by transfer learning, so the details for the reconstruction of porous media will be discussed in two sections: Deep Learning Phase and Transfer Learning Phase.

2.1. Deep Learning Phase. In the training process of deep learning, the errors or distances between the output and the expectation can often be obtained by calculating an objective function. These errors are reduced by modifying the internal tunable parameters of the model, which are often referred to as weights. In a typical deep learning system, there are millions of samples and weights for training a model. In order to adjust the weight vectors correctly, it is necessary to calculate the gradient vector of each weight and then adjust the weight vectors in the opposite direction of the gradient vector so that the overall errors are reduced to a reasonable interval. In practical applications, the gradient descent algorithm is used to iterate over the above process, which provides input vector samples, outputs, and errors by calculating the average gradient of these samples and then adjusting the weights [31]. This process is repeated by continuously inputting samples to train the network until the objective function is optimal. In contrast to other optimization techniques, the speed of gradient descent is faster and the generalization ability is stronger [23].

A typical DNN in deep learning is shown in Figure 2, in which a neuron, expressed as "○," is the basic node unit and the neurons constitute the hidden layers. $(x_1, x_2, \cdots, x_i)$ represents the input feature vectors, and "y" is the predicted output value of the output layer. Excluding the input layer and the output layer, the neural network has n layers. "$l\,(= 1, 2, \cdots, n)$" represents the serial number of the lth layer in a neural network. Eq. (1) is the output of each hidden layer, and Eq. (2) means the output activated by an activation function:

$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}, \tag{1}$$

$$a^{[l]} = f\left(z^{[l]}\right), \tag{2}$$

where z is the output of each hidden layer; a is the output of the activation function taking z as the only parameter; w and b are, respectively, the weight and the bias; and $f(\cdot)$ is the activation function which has several options including the sigmoid (Eq. (3)), tanh (Eq. (4)), and ReLu (Eq. (5)):

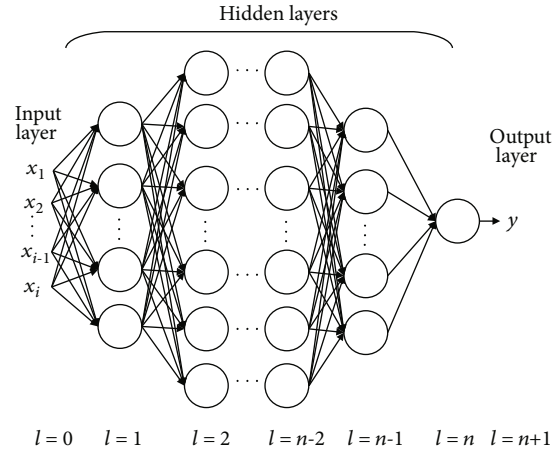


Figure 2: A typical DNN.

$$f(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \tag{3}$$

$$f(x) = \tanh\,(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{4}$$

$$f(x) = \text{ReLu}(x) = \begin{cases} 0, & \text{if } x \le 0, \\ x, & \text{if } x > 0. \end{cases} \tag{5}$$

In the training process, as the layer number of the neural network increases, the gradient of the training parameters will become smaller. It is a common issue called "gradient disappearance," which will prevent the parameters from changing their values and even break the training process. The ReLu can solve the problem of "gradient disappearance," so it is better than the sigmoid and the tanh functions [32, 33]. Hence, the ReLu is used in the proposed method as the activation function.

In DNNs, the loss function measures the quality of simulation in a single training sample. For all the training samples, a cost function $J(w, b)$ and a loss function $L(y_i, y_i^{\text{pre}})$ need to be defined:

$$J(w, b) = \frac{1}{m} L\left(y_i, y_i^{\text{pre}}\right), \tag{6}$$

$$L\left(y_i, y_i^{\text{pre}}\right) = \sum_{i=1}^{n} \left(y_i - y_i^{\text{pre}}\right)^2, \tag{7}$$

where m, w, and b are the number of input samples, weights, and bias of neurons, respectively, and y_i and y_i^{pre}, respectively, represent the real and predicted values of the output.

The neural network is trained to iterate over parameters by forward propagation [34] and backward propagation [35]. As shown in Eqs. (1) and (2), forward propagation calculates the output z and a of each hidden layer from the input, the total output y, and the cost function $J(w, b)$ [36].

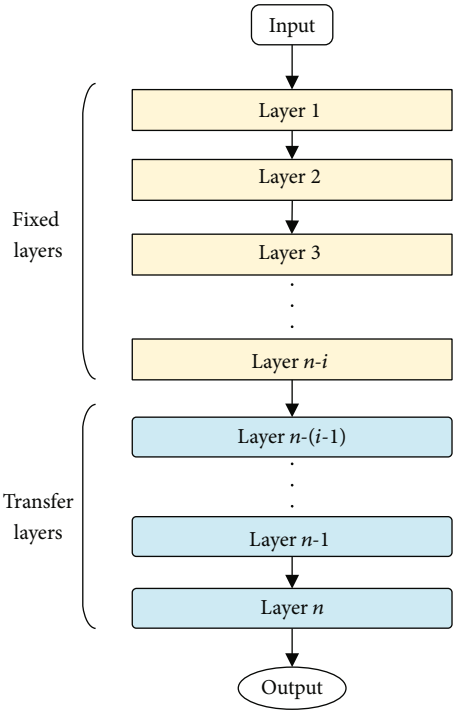


Figure 3: The structure of DTL.

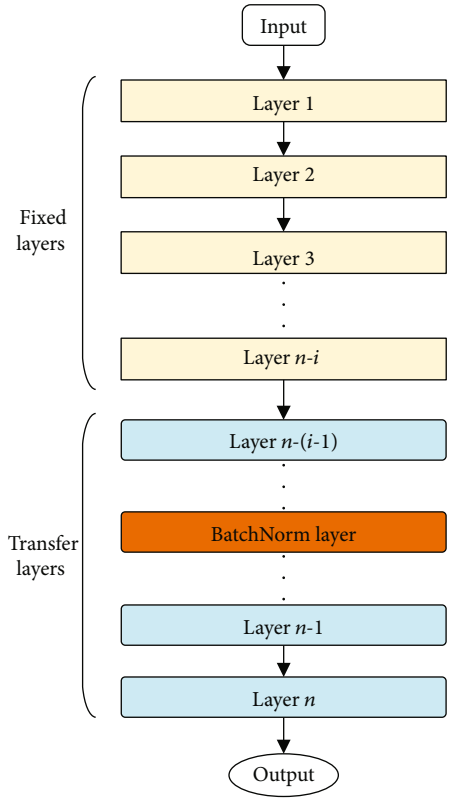


Figure 4: The structure of DTL using the AdaBN method.

Meanwhile, as shown in Eqs. (8) and (9), backward propagation calculates $dw^{[l]}$ and $db^{[l]}$ by the chain derivation rule [37] and then updates w, b in every forward propagation and backward propagation:

$$dw^{[l]} = \frac{\partial(J(w,b))}{\partial w^{[l]}}, \quad \text{where } w^{[l]} = w^{[l]} - \alpha \cdot dw^{[l]}, \tag{8}$$

$$db^{[l]} = \frac{\partial(J(w,b))}{\partial b^{[l]}}, \quad \text{where } b^{[l]} = b^{[l]} - \alpha \cdot db^{[l]}, \tag{9}$$

where α is the learning rate. The purpose of training is to find the appropriate w and b to minimize the total cost $J(w, b)$, which is calculated by forward propagation and backward propagation. The parameters (e.g., w, b) are derived through the chain derivation rule and updated in the whole network in the above computation.

When training and learning complex features, it is not necessary to have many nodes in each layer, but the number of layers is more important. For all these hidden layers, the first few layers can learn some lower-level simple features while the latter ones can combine them into more complex features.

2.2. Transfer Learning Phase. DTL needs to transfer some features into a new network, leading to a separation of these layers (an input layer, an output layer, and some hidden layers) into two categories: fixed layers and transfer layers, as shown in Figure 3. In transfer learning, the fixed layers will not change the parameters but the transfer layers will update the parameters, so it is very important to determine which layers should be transferred or fixed in DTL.

As mentioned above, fixed layers in DTL will not be changed and only learn simple features from raw data. On the contrary, transfer layers are changeable and learn complex features. The number of transfer layers is adjustable and determined according to the total number of layers in a DNN. A typical transfer learning method named Finetune [29, 30] is introduced in the proposed method, which has the following characteristics: (1) saving much time since it does not need to start a new task from the very beginning when learning and training data; (2) easily appending pretrained models to current datasets to extend training data; (3) simple implementation; and (4) permitting the distributions of training data and test data not necessarily to be completely identical.

In the structure of a neural network, Finetune can change the last layer (i.e., the output layer) and then perform a round of new training by inputting new data. For some more complex models, Finetune can be expanded to include several hidden layers before the output layer for better accuracy. However, Finetune has a poor effect when the distributions of training data and test data are quite different. There is a solution to address the issue by adding an adaptation layer into the layers of DTL, ensuring the accuracy of a transfer learning model. When using the adaptation method, there are two key points to be determined: (i) the layers that are added to the adaptation layer and (ii) the adaptative method. The first one determines the quality of the whole transfer learning, and the second one determines the generalization ability of the network.

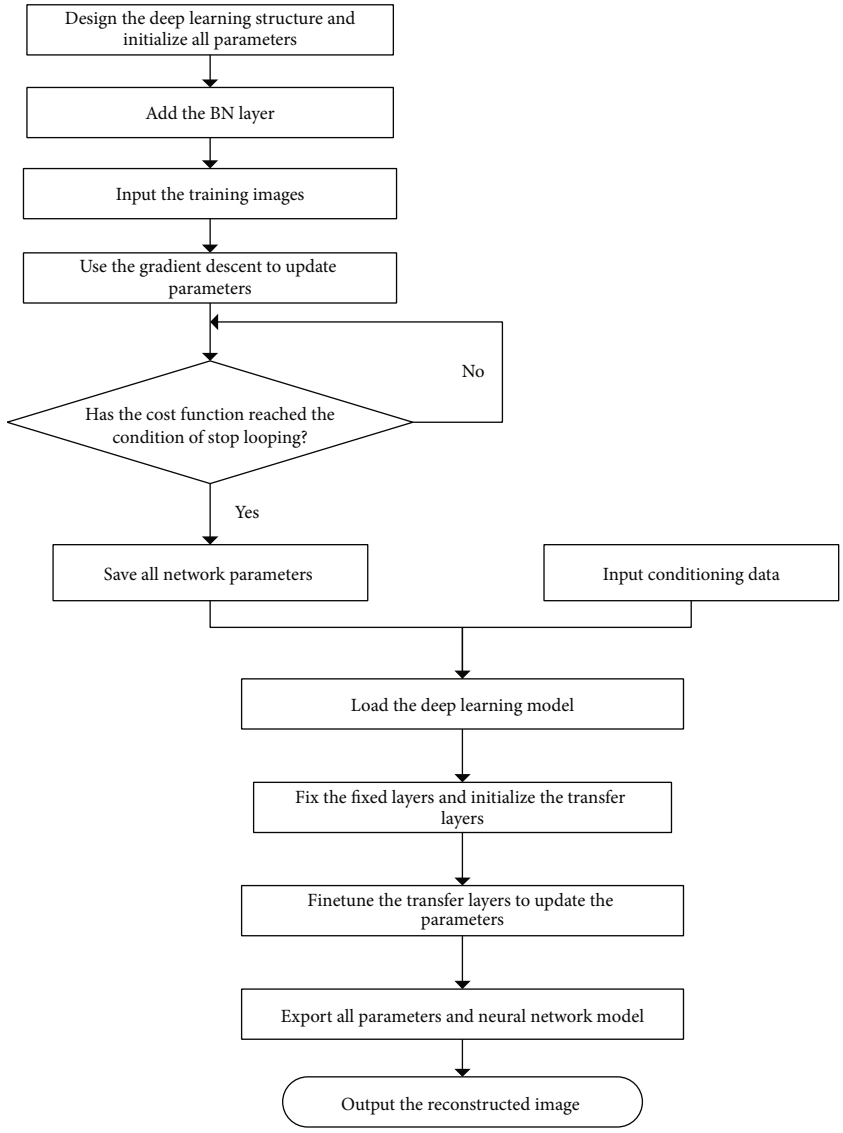


Figure 5: The procedures of the proposed method.

When adding an adaptive layer in DTL, there are altogether three steps: (1) determining the layers that are suitable for adaption, (2) adding an adaptive layer between the transfer layers and the fixed layers, and (3) using the data from the target domain to train the network by Finetune. The following loss function is defined in DTL:

$$l = l_c(D_s, y_s) + \lambda \cdot l_A(D_s, D_t), \tag{10}$$

where D_s and D_t are, respectively, the input of the source dataset (i.e., TIs) and the target dataset (e.g., known conditioning data), y_s is the output of the source dataset, l is the total loss of the network, $l_c(D_s, y_s)$ is the loss of the network and $l_A(D_s, D_t)$ is the adaptive loss of the network, and λ is the weight of the dataset.

The learning quality of DTL can be well improved by adding an adaptive layer, but the computational complexity is additionally increased, and the adaptive layer is also difficult to select. In order to simplify the deep network adaptation, the adaptive layer is replaced by a BatchNorm (BN) layer [38] for normalization to incorporate the adaptation of statistical features, which is called an Adaptive Batch Normalization (AdaBN) method. The structure of DTL using the AdaBN method is shown in Figure 4.

As shown in Figure 4, transfer layers consist of some hidden layers and the BN layer. The latter can reduce the differences of distribution from the source dataset and the target dataset. The goal of data normalization for each layer in AdaBN is to make each layer receive data from a similar distribution to mitigate the problem of dataset shift. AdaBN normalizes samples from the source dataset to a zero-mean and the same variance. The BN layer is defined in Eq. (11) for the jth neuron:

$$y_j = w_j \frac{\left(x_j - \mu_j^t\right)}{\sigma_j^t} + b_j, \tag{11}$$

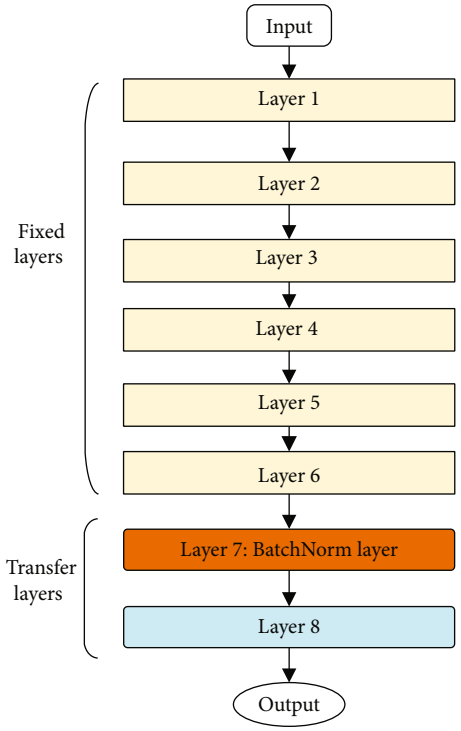


Figure 6: Architecture of the DTL network used in our tests.

where x_j and y_j are the input and output of the jth neuron; w_j and b_j are, respectively, the weight and the bias; μ_j and σ_j are the mean and standard deviation; and t means the tth iteration.

For the input of m samples at the nth iteration, suppose the input mean and input variance are μ and σ^2, respectively. The calculated mean μ_j and the variance ${\sigma^2}_j$ for the jth neuron can be updated as follows:

$$\begin{aligned} d &= \mu - \mu_j, \\ \mu_j &\longleftarrow \mu_j + \frac{dm}{n_j}, \\ \sigma_j^2 &\longleftarrow \frac{\sigma_j^2 n_j}{n_j + m} + \frac{\sigma^2 m}{n_j + m} + \frac{d^2 n_j m}{(n_j + m)^2}, \\ n_j &\longleftarrow n_j + m, \end{aligned} \tag{12}$$

where d is the difference between the input mean μ at the nth iteration and the calculated mean μ_j at the $(n-1)$th iteration and n_j is the sum of all the samples of the previous $n-1$ iterations, i.e., the cumulative number of samples. The variables on the left of the operator "$\longleftarrow$" are updated at each iteration. At the first iteration, μ_j and ${\sigma^2}_j$ are initialized to 0 and 1, respectively.

2.3. Hyperparameters in DTL. In DTL, some parameters called hyperparameters can be set artificially before training. Hyperparameters do not need to be adjusted and are often divided into three categories: network parameters, optimization parameters, and regularization parameters.

Network parameters include the interaction mode (addition, multiplication, concatenation, etc.) between network layers, the number of network layers (also called depth), and the activation functions. Optimization parameters include the learning rate, batch sizes, parameters of different optimizers, and adjustable parameters of some loss functions. Regularization parameters include the coefficient of weight attenuation and dropout.

At the beginning of the training process, a large learning rate can speed up training. Learning rate α is defined as

$$\alpha = \frac{k}{\sqrt{t}} \cdot \alpha_0, \tag{13}$$

where α_0 is the initial learning rate, k is the number of rounds of gradient descent, and t is the training time. With the growth of training times, the learning rate will gradually decrease, ensuring that Eq. (13) converges. Since the DTL model is complex and possibly prone to overfitting in calibration and underfitting in validation, the generalization ability of the model should be considered. As for overfitting problems, a regularization method can be used to reduce overfitting by adding a regular term Ω to the loss function [39]:

$$\Omega = \lambda \sum \|w\|^2, \tag{14}$$

$$L\left(y_i, y_i^{\text{pre}}\right) = \sum_{i=1}^{n} \left(y_i - y_i^{\text{pre}}\right)^2 + \Omega, \tag{15}$$

where λ is the weight of regularization and Ω reduces the number of feature vectors and complexity of the model to prevent overfitting.

Dropout is another regularization method for overfitting [40], which discards the values of some neuron nodes during each round of training by randomly setting some points to 0 (i.e., these neural cells are considered "dropped out"), improving the generalization ability of the model. The method of adding a regular term and dropout are both used in the proposed method to prevent overfitting. For underfitting in validation, choosing an appropriate network, adjusting hyperparameters, and training more times should be used, which will be discussed with some other hyperparameters such as the number of hidden layers and the BN layer in Section 4.4.

3. Procedures of the Proposed Method

As shown in Figure 5, the procedures of the proposed method are as follows.

Step 1. Design the deep learning structure (number of layers, activation functions, etc.) and add the BN layer into the hidden layers.

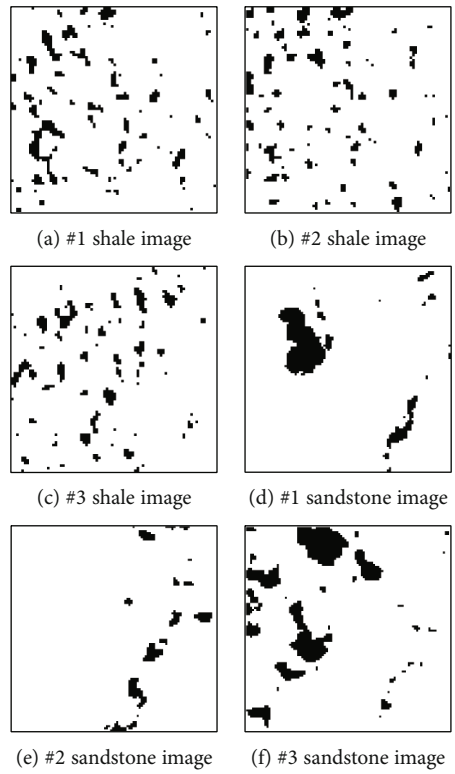


Figure 7: Some shale and sandstone cross-section images from real volume data.

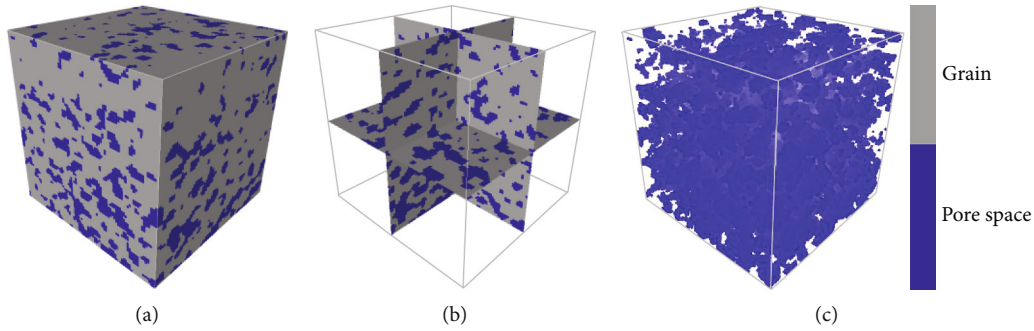


Figure 8: The TI (training image) of shale: (a) exterior; (b) cross-section ($X = 40$, $Y = 40$, and $Z = 40$); (c) pore space.

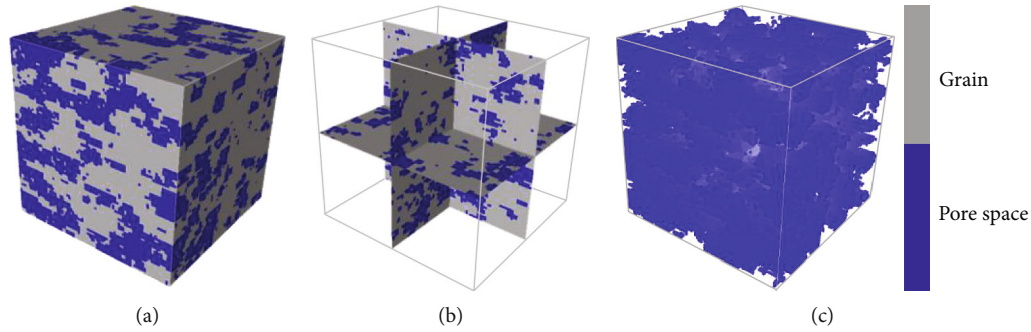


Figure 9: The target image of shale: (a) exterior; (b) cross-section ($X = 40$, $Y = 40$, and $Z = 40$); (c) pore space.

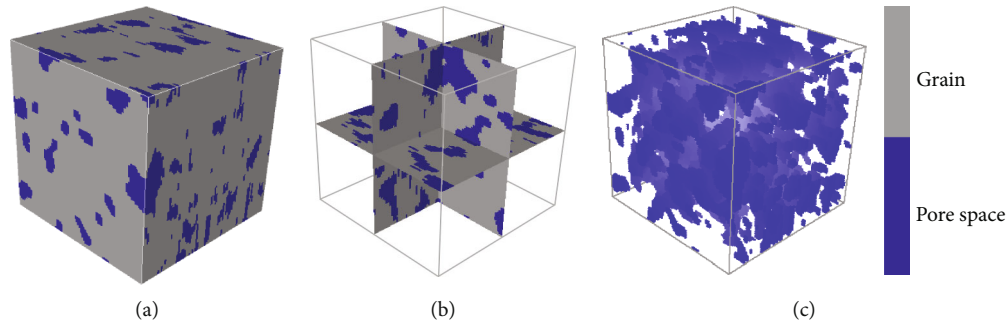


Figure 10: The TI of sandstone: (a) exterior; (b) cross-section ($X = 40$, $Y = 40$, and $Z = 40$); (c) pore space.

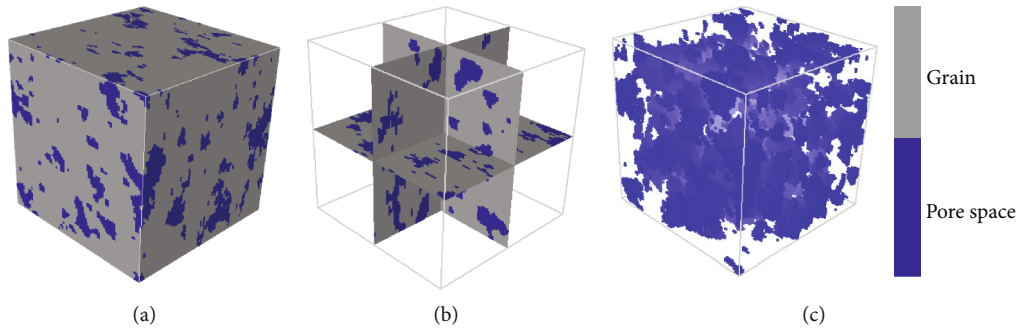


Figure 11: The target image of sandstone: (a) exterior; (b) cross-section ($X = 40$, $Y = 40$, and $Z = 40$); (c) pore space.

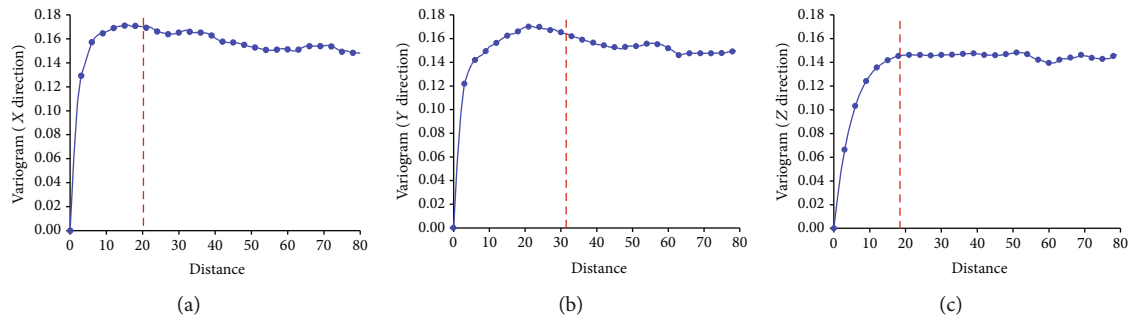


Figure 12: Variogram curves of original volume data (shale) in three directions: (a) X direction; (b) Y direction; (c) Z direction.

Step 2. Input 2D or 3D TIs of porous media, then use the gradient descent to update the parameters and set the adaptive learning rate.

Step 3. Save all the network parameters (w, b, the layer number, and the number of neurons in the layer) and hyperparameters (e.g., learning rate α and the activation function).

Step 4. Load the model and parameters and then take the conditioning data from the new dataset as input.

Step 5. Set the fixed layers and initialize the transfer layers, then update the parameters of transfer layers using Finetune.

Step 6. Export the neural network model including the structure and corresponding parameters to a file.

Step 7. Output 2D or 3D images of porous media using the above model and parameters.

Multiple source datasets (i.e., TIs) can be used in the proposed method. Besides, reconstructing porous media with any different sizes of 2D or 3D images can be realized according to the features extracted from TIs. Since TIs are real 3D images, the reconstruction retains the features of porous media in the real world.

4. Experimental Results and Analyses

Since DTL can be run by using a tensorflow-gpu framework [41] accelerated by GPU, the following tests were performed based on a tensorflow-gpu framework with a CPU of Intel Core i7-8700 (3.2 GHz), a memory of 8 GB, and a GPU of GeForce GTX 1070 (6 GB memory). As mentioned previously, the proposed method uses deep learning to extract all features from TIs and then save the trained model and corresponding parameters. The number of layers often can be determined according to some trials and experiences. Generally, the more complicated the porous media are, the more

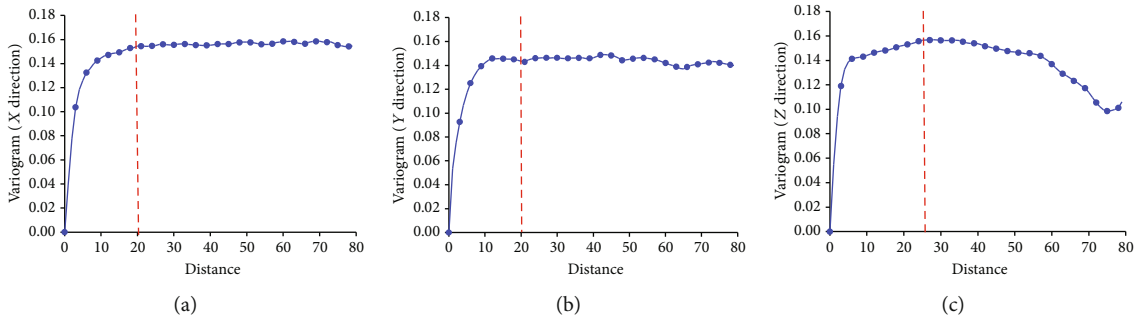


Figure 13: Variogram curves of original volume data (sandstone) in three directions: (a) X direction; (b) Y direction; (c) Z direction.

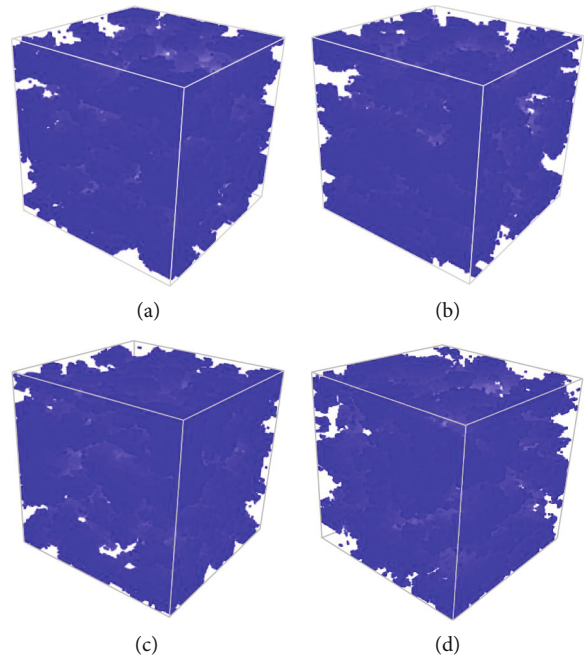


Figure 14: Reconstructed pore spaces of shale using (a) DTL; (b) SNESIM; (c) FILTERSIM; (d) DISPAT.

hidden layers should be applied. In the practical tests, the reconstruction of porous media is based on the model displayed in Figure 6, in which the network has 8 hidden layers using the ReLu function as the activation function [33]. The details about the architecture shown in Figure 6 will be discussed in Section 4.4.

4.1. Training Data and the Representative Elementary Volume. To evaluate the effects and applicability of the proposed method in the reconstruction of porous media, the real shale volume data with the resolution of 64 nanometers obtained by nano-CT and the real sandstone volume data with the resolution of 10.9 micrometers obtained by micro-CT were used as the test data for the following tests. Figure 7 shows some cross-sections of the volume data of shale and sandstone with two facies: grains (white) and pores (black).

4.1.1. 3D Experimental Shale Images. Before applying any reconstruction methods, two 3D cubes with $80 \times 80 \times 80$ voxels were extracted from different parts of the original shale volume data: one is used as a TI and the other is a target image. A target image can be a judge for comparing the reconstructed results when extracting the features from a TI and then reproducing them using the proposed method conditioned to some conditioning data from the target image. Figures 8(a)–8(c) are the exterior ($80 \times 80 \times 80$ voxels), cross-sections ($X = 40$, $Y = 40$, and $Z = 40$), and pore spaces of the TI (porosity = 0.1860). Similarly, the exterior ($80 \times 80 \times 80$ voxels), cross-sections ($X = 40$, $Y = 40$, and $Z = 40$), and pore spaces of the target image (porosity = 0.2812) are shown in Figures 9(a)–9(c).

4.1.2. 3D Experimental Sandstone Images. Similarly, two 3D sandstone cubes with $80 \times 80 \times 80$ voxels were, respectively, used as a TI and a target image. Figures 10(a)–10(c) are the exterior ($80 \times 80 \times 80$ voxels), cross-sections ($X = 40$, $Y = 40$, and $Z = 40$), and pore spaces of the TI (porosity = 0.1705). Figures 11(a)–11(c) display the exterior ($80 \times 80 \times 80$ voxels), cross-sections ($X = 40$, $Y = 40$, and $Z = 40$), and pore spaces of the target image (porosity = 0.1121). The porosity values of

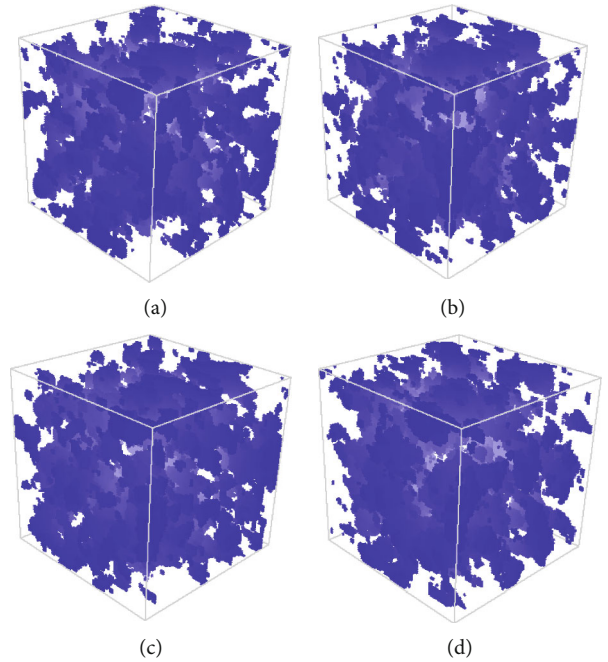


Figure 15: Reconstructed pore spaces of sandstone using (a) DTL; (b) SNESIM; (c) FILTERSIM; (d) DISPAT.

Table 1: Porosity of the TIs, the target images, and the reconstructed images using DTL, SNESIM, FILTERSIM, and DISPAT.

	TI	Target image	DTL	SNESIM	FILTERSIM	DISPAT
Shale	0.1860	0.2812	0.2734	0.2932	0.2495	0.2652
Sandstone	0.1705	0.1121	0.1082	0.1302	0.0953	0.0971

Table 2: Porosity of 10 reconstructed shale and sandstone images using SNESIM, FILTERSIM, DISPAT, and DTL.

Reconstruction	SNESIM		FILTERSIM		DISPAT		DTL	
	Shale	Sandstone	Shale	Sandstone	Shale	Sandstone	Shale	Sandstone
#1	0.2989	0.1243	0.2407	0.0897	0.2732	0.0947	0.2797	0.1033
#2	0.3056	0.1358	0.2721	0.1003	0.2608	0.0976	0.2701	0.1002
#3	0.3098	0.1196	0.2487	0.1215	0.2559	0.1107	0.2711	0.1137
#4	0.2754	0.1332	0.3011	0.0941	0.2717	0.1023	0.2675	0.1089
#5	0.2921	0.1265	0.2316	0.0956	0.2649	0.1142	0.2508	0.1074
#6	0.2646	0.0945	0.2724	0.0975	0.3012	0.1013	0.2721	0.1113
#7	0.3006	0.1101	0.2679	0.1132	0.2601	0.1045	0.2801	0.1069
#8	0.2946	0.1403	0.2433	0.1088	0.2590	0.0989	0.2721	0.1109
#9	0.3026	0.0921	0.2518	0.1102	0.2543	0.1012	0.2739	0.1026
#10	0.2886	0.1285	0.2298	0.0875	0.2772	0.0991	0.2766	0.1152
Average	0.2933	0.1205	0.2559	0.1018	0.2678	0.1025	0.2714	0.1080

the target images and TIs for both shale and sandstone are deliberately quite different to test the applicability of the proposed method.

4.1.3. Representative Elementary Volume of Samples. Before performing the tests, the representative elementary volume (REV) [42, 43] of the training data should be determined first. It is important to observe the influence which the scale of the studied sample exerts when reconstructing porous media, i.e., a key point is to determine the minimum size of a studied sample in which the features tend to be substantially stable and can be independent of the size of the sample. When the size of the sample is less than an REV, the features possibly will change easily with the different sizes of the sample, showing obvious fluctuations in the features. On the contrary, when the samples are bigger than the REV, they have

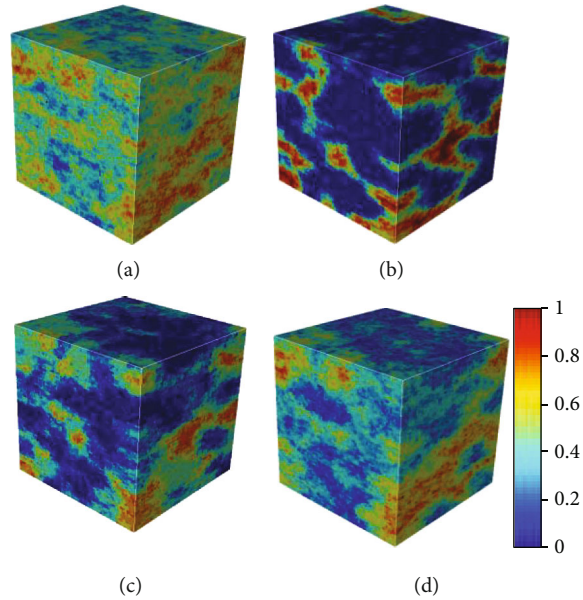


Figure 16: Averages of 10 reconstructed shale images using (a) SNESIM; (b) DTL; (c) FILTERSIM; (d) DISPAT.

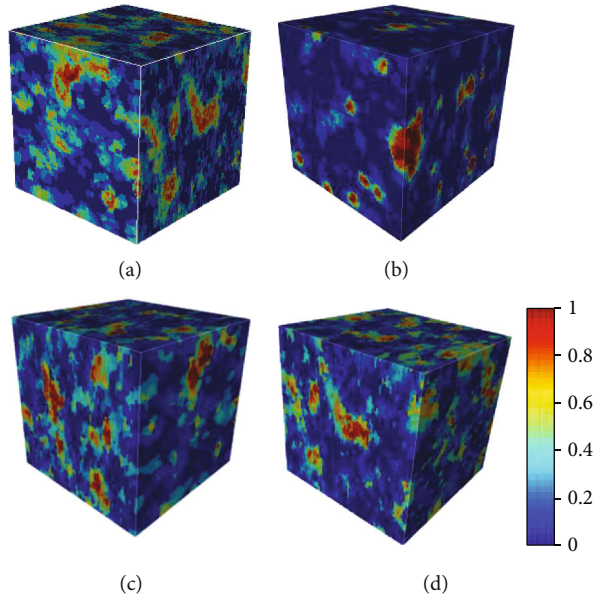


Figure 17: Averages of 10 reconstructed sandstone images using (a) SNESIM; (b) DTL; (c) FILTERSIM; (d) DISPAT.

almost the same features no matter where they are located in the original data.

According to the definition of an REV, all the samples of porous media bigger than an REV have the same statistical distribution as long as they are taken from the original data. The influence of different locations in the original data does not need to be considered. The REV actually conforms to the concepts of ergodicity and stationarity in statistics because a sample bigger than an REV has fixed statistical distribution, meaning the experimental samples bigger than an REV also have ergodicity and stationarity when they meet the REV requirements.

There are two major methods for determining an REV. The first one is porosity widely used in soil science and material science, regardless of the macroscale parameters; the second one determines an REV based on some macroscale parameters without considering the microscale parameters of a sample, often used in engineering mechanics [44, 45]. For homogeneous porous media, the method of using porosity for REV may be effective, but it will not work for

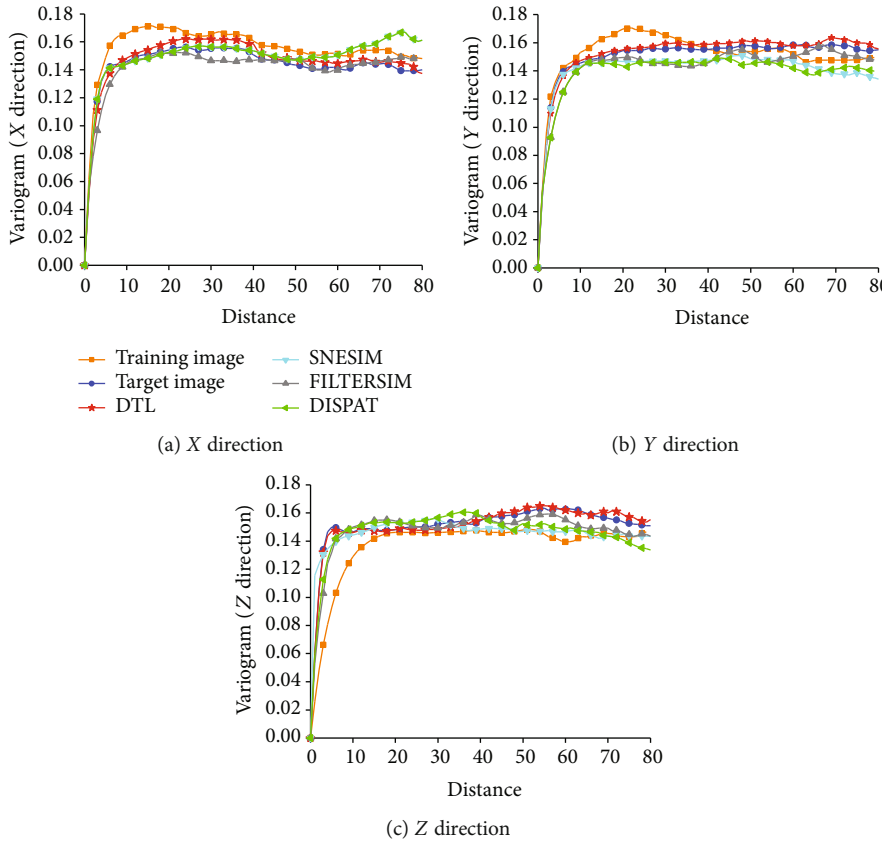


FIGURE 18: Variogram curves of shale images from the TI, the target image, and reconstructed images using SNESIM, FILTERSIM, DISPAT, and DTL in three directions.

heterogeneous porous media since porosity tends to vary largely with different sizes. Therefore, some other methods should be considered to determine an REV for heterogeneous porous media to ensure that their features will not change largely with different sizes. In our tests, the size of REV was determined by the variogram $\gamma(h)$, which is often used to represent the correlation and variability of spatial structural changes in a certain direction and defined in

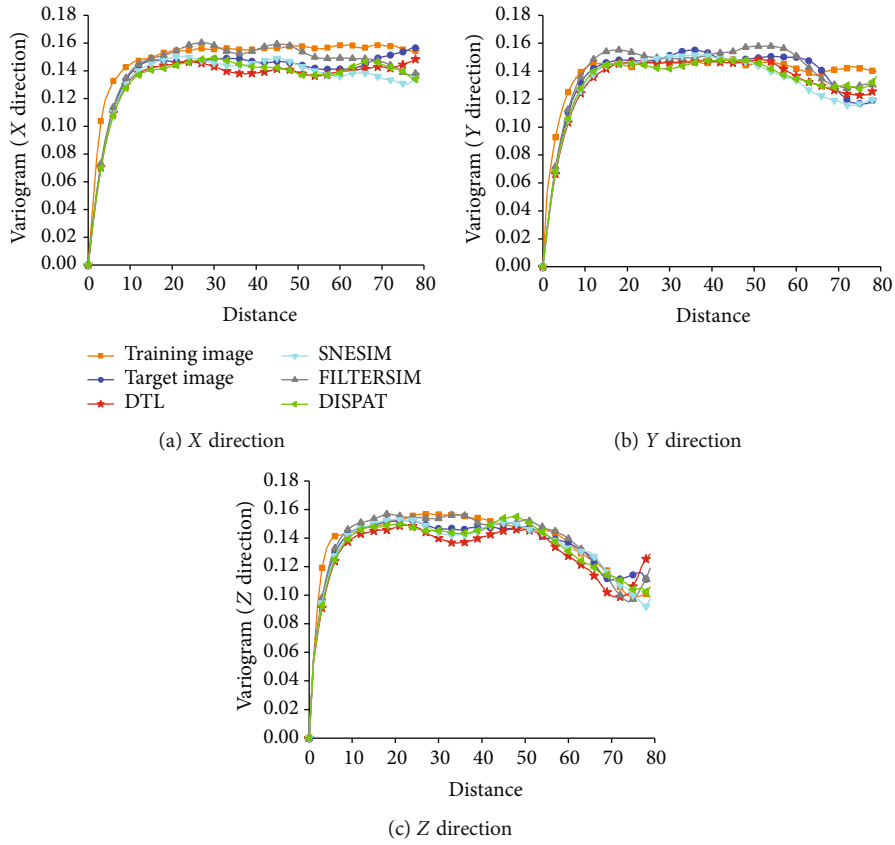
$$\gamma(h) = \frac{1}{2}E\{[Z(x+h) - Z(x)]^2\}, \tag{16}$$

where E means the mathematical expectation, $Z(x)$ is a variable value at the location x, and h is the lag between two locations x and $x + h$.

The specific procedure of determining an REV is the following: firstly, the variogram curves of pores in the X, Y, and Z directions are plotted, respectively; secondly, when the variogram curves at all three directions begin to become stable, the corresponding size of porous media is an REV. The variogram curves computed from the original volume data of shale and sandstone in three directions (X, Y, and Z) are shown in Figures 12 and 13. The abscissa indicates the spatial distance h (unit: voxel), and the ordinate indicates the value of the variogram. In Figure 12, the variogram curves in three directions tend to be stable at distance = 20, 31, and 19 voxels, respectively (indicated with red dashed lines), so the REV can be at least $20 * 31 * 19$ voxels for shale; similarly, the REV can be at least $20 * 20 * 26$ voxels for sandstone inferred from Figure 13. Since the TIs and the target images of shale and sandstone in our tests are $80 \times 80 \times 80$ voxels, their sizes are bigger than the REV and meet the experimental requirements.

4.2. Reconstructions and Comparisons with Other Methods. Some sample points extracted from the target images of shale and sandstone were, respectively, used as conditioning data of shale and sandstone reconstruction, accounting for 1% of total voxels of the target images, in which the pore voxels and the grain voxels have the same number (meaning the pore and grain voxels, respectively, account for 50% in the conditioning data). The proportion of conditioning data is deliberately quite different from the porosity of the target image to prove the applicability of the proposed method. Suppose the pore value is 1 and the grain value is 0. For convenience, the proposed method is called DTL hereafter. Reconstructions of porous media (shale and sandstone) were performed using DTL and some typical MPS methods (SNESIM, FILTERSIM, and DISPAT) with conditioning data and TIs.

4.2.1. Reconstructed Pore Spaces of Shale. Figure 14 is the reconstructed 3D pore spaces of shale, respectively, using DTL, SNESIM, FILTERSIM, and DISPAT. It is seen that all



(a) X direction

(b) Y direction

(c) Z direction

Figure 19: Variogram curves of sandstone images from the TI, the target image, and reconstructed images using SNESIM, FILTERSIM, DISPAT, and DTL in three directions.

the reconstructed pore spaces by the four methods have similar structures with the target image (Figure 9(c)), and the long connectivity of the pore spaces is well reproduced.

4.2.2. Reconstructed Pore Spaces of Sandstone. Figure 15 is the reconstructed 3D pore spaces of sandstone, respectively, using DTL, SNESIM, FILTERSIM, and DISPAT. The reconstructed images by the four methods also have similar structures with the target image (Figure 11(c)), and the long connectivity of pore spaces is well reproduced, too.

4.2.3. Averages of Reconstructions. For convenience, the porosity values of shale and sandstone reconstructions are put together. The porosity values of TIs, the target images, and the reconstructed images are shown in Table 1.

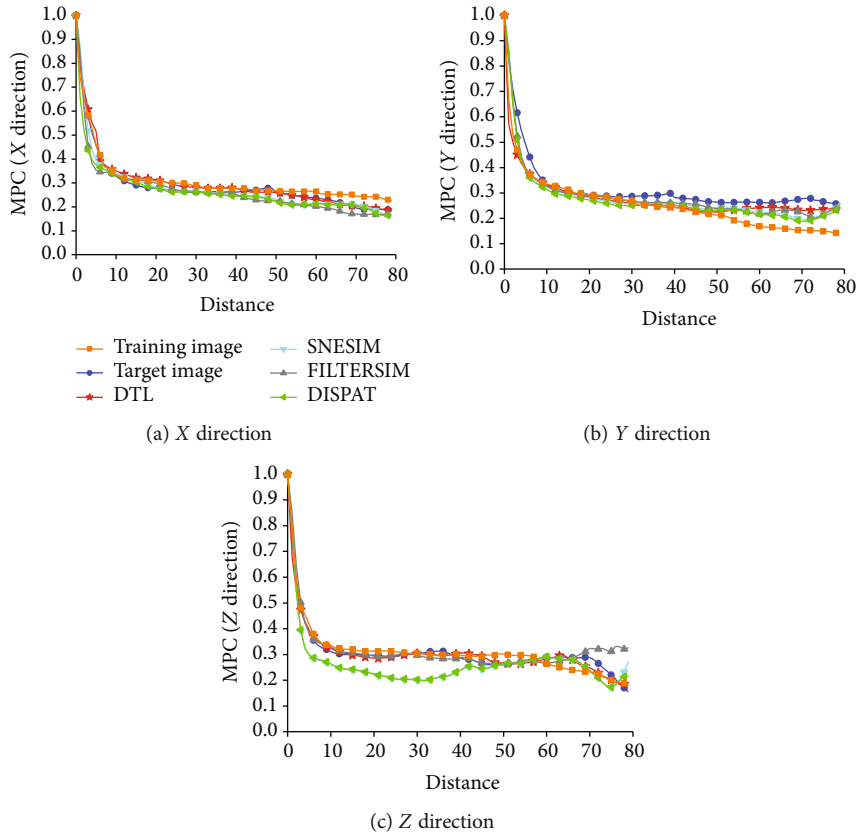
To get the average performance, another 10 shale and sandstone reconstructions using SNESIM, FILTERSIM, DISPAT, and DTL were performed, as shown in Table 2. All the reconstructed images are cubes with the same size of $80 \times 80 \times 80$ voxels. Since each voxel within the cube has its fixed location and value, the average values of these ten reconstructions can be computed and constitute an "average cube" with the size of $80 \times 80 \times 80$ voxels for each method (SNESIM, FILTERSIM, DISPAT, and DTL), as shown in Figures 16 and 17. The attribute values of each voxel in the "average cube" are the average of the voxel at the same location in reconstructed images. It seems that the reconstructed shale image using DTL has a clearer distinction between the pore spaces and grains, showing that DTL has relatively fixed reconstructed results compared with the other three methods.

4.2.4. Variogram and Multiple-Point Connectivity. Variogram depends on the independent variable h, and variogram curves can represent the spatial variability of two points in one direction. In the reconstruction of porous media, a variogram is also used for evaluation. The variogram curves of the TIs, the target images, and the reconstructions of SNESIM, FILTERSIM, DISPAT, and DTL were computed in the directions of X, Y, and Z, as shown in Figures 18 and 19. It is seen that the variogram curves of the DTL method, in all three directions, are quite close to those of the target images.

Multiple-point connectivity (MPC) [12, 46] can measure the joint connectivity between multiple points in one direction, which is defined as

$$\begin{aligned} \mathrm{MPC}(n) &= E\{S(u) \cdot S(u+1) \cdot \cdots \cdot S(u+(n-1) \cdot h)\} \\ &= E\left\{\prod_{i=0}^{n-1} S(u+i \cdot h)\right\}, \end{aligned} \tag{17}$$

where $S(u)$ is the attribute value at the position u, h is the lag distance, n is the number of nodes or points in one



(a) X direction (b) Y direction

(c) Z direction

Figure 20: MPC curves (shale) of the TI, the target image, and reconstructed images using SNESIM, FILTERSIM, DISPAT, and DTL.

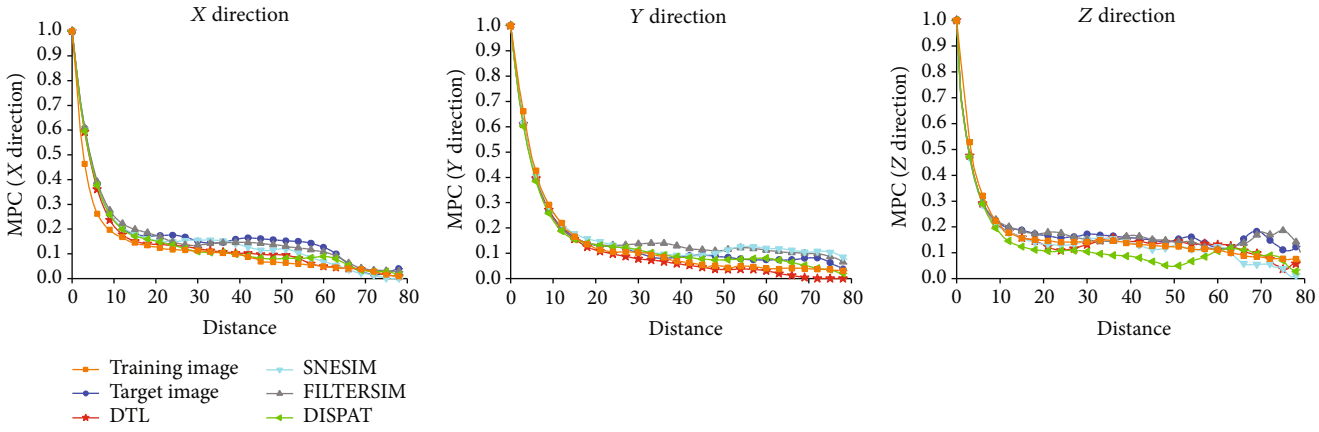


Figure 21: MPC curves (sandstone) of the TI, the target image, and reconstructed images using SNESIM, FILTERSIM, DISPAT, and DTL.

Table 3: Permeability (shale) of the TI, the target image, and average permeability of ten reconstructed images in three directions using DTL, SNESIM, FILTERSIM, and DISPAT.

Direction	Permeability (10^{-3} μm^2)					
	Train image	Target image	DTL	SNESIM	FILTERSIM	DISPAT
X	7.011	7.350	7.037	6.956	7.681	6.842
Y	6.904	7.656	7.165	7.241	7.312	6.775
Z	5.713	6.421	6.758	5.976	7.014	6.041

Table 4: Permeability (sandstone) of the TI, the target image, and average permeability of ten reconstructed images in three directions using DTL, SNESIM, FILTERSIM, and DISPAT.

Direction	Permeability (10^{-3} μm^2)					
	Train image	Target image	DTL	SNESIM	FILTERSIM	DISPAT
X	2.404	2.132	2.331	2.517	2.297	2.770
Y	2.497	2.319	2.746	2.638	2.261	2.466
Z	2.051	1.815	1.969	2.021	1.973	2.034

Table 5: The average numbers of pores in the TIs, the target images, and 10 reconstructed images using FILTERSIM, SNESIM, DISPAT, and DTL.

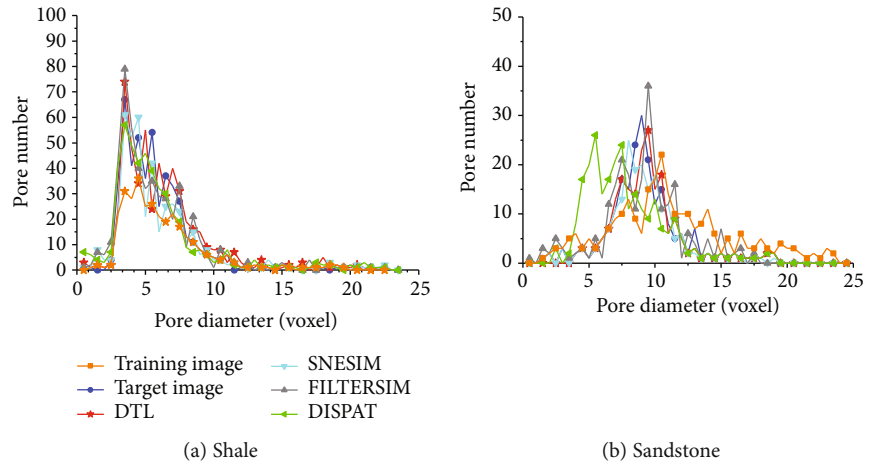
	Train image	Target image	FILTERSIM	SNESIM	DISPAT	DTL
Shale	326	480	513	445	498	507
Sandstone	265	217	250	199	259	205

Table 6: The average pore diameters (shale) in the TI, the target image, and the reconstructed images of SNESIM, FILTERSIM, DISPAT, and DTL.

	TI	Target image	SNESIM	FILTERSIM	DISPAT	DTL
Average diameter (voxel)	4.88	5.77	6.02	5.92	5.93	5.98
Maximum diameter (voxel)	19.21	22.51	22.10	20.26	22.56	21.16
Minimum diameter (voxel)	1.65	2.45	1.52	2.16	1.27	2.84

Table 7: The average pore diameters (sandstone) in the TI, the target image, and the reconstructed images of SNESIM, FILTERSIM, DISPAT, and DTL.

	TI	Target image	SNESIM	FILTERSIM	DISPAT	DTL
Average diameter (voxel)	9.85	9.30	8.07	9.12	7.93	9.51
Maximum diameter (voxel)	23.69	19.20	18.65	17.56	19.26	19.15
Minimum diameter (voxel)	1.97	3.56	1.10	1.51	2.59	3.95



(a) Shale (b) Sandstone

Figure 22: The distributions of pore diameters of the TIs, the target images, and the reconstructed images of DTL, SNESIM, FILTERSIM, and DISPAT.

direction, and E is the mathematical expectation. Suppose $S(u) = 1$ when u corresponds to pore space; otherwise, $S(u) = 0$. As shown in Figures 20 and 21, the MPC curves of the TIs, the target images, and reconstructions are very similar in X and Y directions, but in the Z direction (especially in Figure 20), DTL shows better performance since their curves are much closer to those of the target images.

Table 8: The average memory usage, CPU/GPU utilization, and running time of SNESIM, FILTERSIM, DISPAT, and DTL for 10 reconstructions (shale).

	SNESIM	FILTERSIM	DISPAT	DTL
Average memory usage	92%	90%	90%	35.5%
Average CPU utilization	93%	92%	90%	22%
Average GPU utilization	None	None	None	90%
First-round running time	11612 sec	12530 sec	13780 sec	1160 sec
Average running time (excluding the first running time)	8656 sec	9630 sec	9520 sec	496 sec

Table 9: The average memory usage, CPU/GPU utilization, and running time of SNESIM, FILTERSIM, DISPAT, and DTL for 10 reconstructions (sandstone).

	SNESIM	FILTERSIM	DISPAT	DTL
Average memory usage	90%	90%	90%	41%
Average CPU utilization	91%	90%	90%	25%
Average GPU utilization	None	None	None	90%
First-round running time	8820 sec	10100 sec	9860 sec	1310 sec
Average running time (excluding the first running time)	5060 sec	7300 sec	6700 sec	456 sec

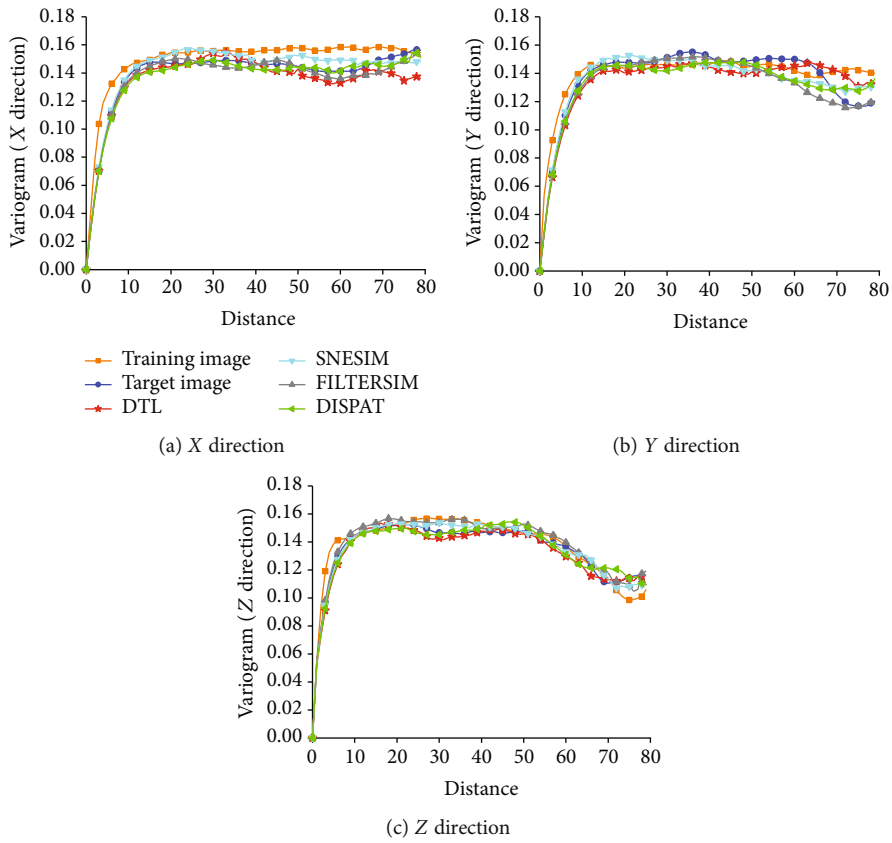
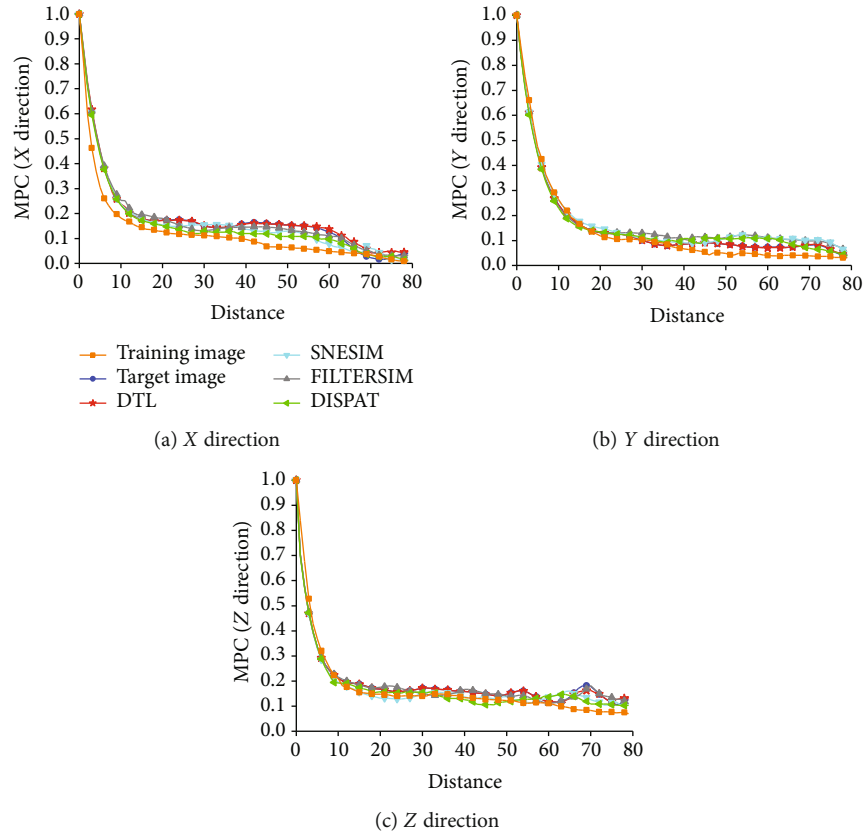


Figure 23: Variogram curves of reconstructed sandstone images by 5% conditioning data using SNESIM, FILTERSIM, DISPAT, and DTL in three directions.

4.2.5. Permeability Estimation Using the Lattice Boltzmann Method. Permeability means the ability to allow fluid to pass through porous media and often is related to porosity, geometry of pores in the direction of liquid penetration, and other factors [47]. The permeability of reconstructed porous media is computed using the Lattice Boltzmann Method (LBM) [3, 48]. The evolution equation is defined as follows:

$$f_i(\mathbf{x} + \Delta x, t + \Delta t) = f_i(\mathbf{x}, t) - \frac{1}{\tau}\left(f_i(\mathbf{x}, t) - f_i^{\mathrm{eq}}(\mathbf{x}, t)\right), \quad (18)$$



(a) X direction

(b) Y direction

(c) Z direction

FIGURE 24: MPC curves of reconstructed sandstone images by 5% conditioning data using SNESIM, FILTERSIM, DISPAT, and DTL in three directions.

where τ is the dimensionless relaxation time, $f_i(\mathbf{x}, t)$ is the density distribution function in the ith velocity direction at the lattice location $\mathbf{x}$ and the time t, and Δt and Δx are, respectively, increments of time and space. The equilibrium distribution function is

$$f_i^{\mathrm{eq}}(\mathbf{x}, t) = w_i \rho \left[1 + 3\frac{\mathbf{e}_i \cdot \mathbf{u}}{c^2} + 4.5\frac{(\mathbf{e}_i \cdot \mathbf{u})^2}{c^4} - 1.5\frac{\mathbf{u}^2}{c^2}\right], \quad (19)$$

where $c = \Delta x/\Delta t$ is the lattice speed and $\mathbf{u}$ is the fluid velocity. w_i means the weights whose values are $w_i = 1/3\ (i = 0)$, $w_i = 1/18\ (i = 1, 2, \cdots, 6)$, and $w_i = 1/36\ (i = 7, \cdots, 18)$, respectively. $\mathbf{e}_i$ is the discrete velocities:

$$\mathbf{e}_i = \begin{cases} (0, 0, 0), & i = 0, \\ (\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1), & i = 1, \cdots, 6, \\ (\pm 1, \pm 1, 0), (0, \pm 1, \pm 1), (\pm 1, 0, \pm 1), & i = 7, \cdots, 18. \end{cases} \quad (20)$$

The density of the momentum $\rho\mathbf{u}$ and the fluid ρ are

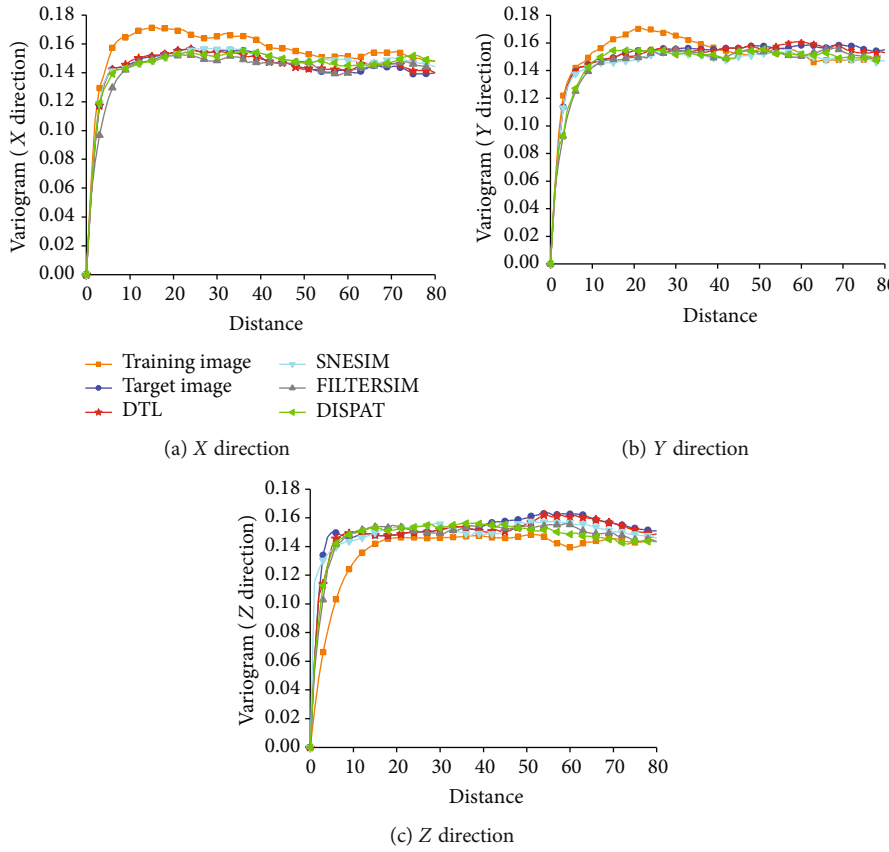
$$\rho = \sum_i f_i(\mathbf{x}, t), \quad (21)$$

$$\rho\mathbf{u} = \sum_i f_i(\mathbf{x}, t)\mathbf{e}_i. \quad (22)$$

Since the internal structures of reconstructed systems are very complicated, the bounce-back scheme is used to obtain no-slip velocity conditions. The inlet and outlet of models are computed by using pressure conditions. The data of the TIs, the target images, and the reconstructed images were, respectively, used as the input files of LBM simulation to calculate the permeability of those models with the size of $80 \times 80 \times 80$ voxels. Some parameters in LBM are defined as follows: $\Delta x = \Delta t = 1$, $\tau = 1$. Two faces of the reconstructed model, which are perpendicular to the flow direction, are left open while all other four faces are sealed with the matrix phase. When convergence is reached, the permeability along this flow direction can be computed according to Darcy's law.

As shown in Tables 3 and 4, the permeability of the TIs, the target images, and the average permeability of ten reconstructed images using DTL, SNESIM, FILTERSIM, and DISPAT in three directions were computed by LBM. The permeability values (especially in the Z direction) of the reconstructed images using DTL are quite close to those of the target images, displaying good reconstruction quality of DTL.

4.2.6. Distribution and Numbers of Pores. Analyses of pore structures were performed by the software Avizo [49] through importing the TIs, the target images, and the reconstructed images of DTL, SNESIM, FILTERSIM, and DISPAT.



(a) X direction

(b) Y direction

(c) Z direction

FIGURE 25: Variogram curves of reconstructed shale images by 5% conditioning data using SNESIM, FILTERSIM, DISPAT, and DTL in three directions.

The pore structures including the number of pores, the pore sizes, and the diameters of each pore were calculated. The diameter of pores is approximately defined as

$$\text{Diameter} = \sqrt[3]{\frac{6V}{\pi}}, \tag{23}$$

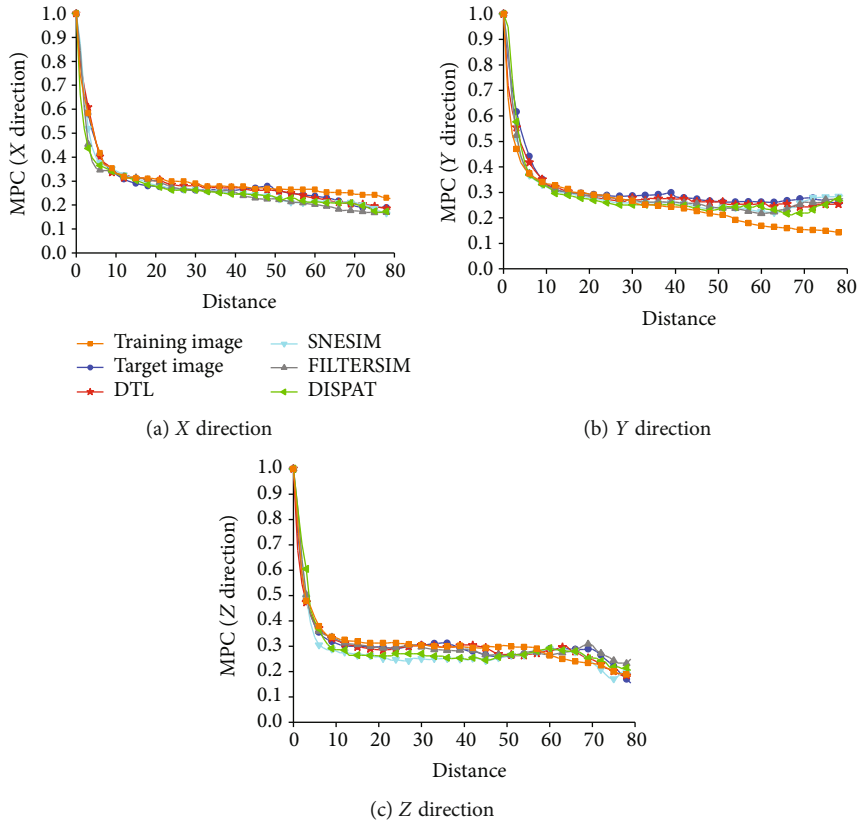
where V is the volume of each pore.

Table 5 shows the average numbers of pores in the TIs, the target images, and 10 reconstructed images using FILTERSIM, SNESIM, DISPAT, and DTL. The diameters of pores in the TIs, the target images, and the reconstructed images of SNESIM, FILTERSIM, DISPAT, and DTL are displayed in Table 6 (shale) and Table 7 (sandstone). Figure 22 shows the distributions of pore diameters of the TIs, the target images, and the reconstructed images of SNESIM, FILTERSIM, DISPAT, and DTL. DTL is not the best in all individual items, but it has shown good quality in the overall performance, judged from Tables 5–7 and Figure 22.

The average memory usage, CPU/GPU utilization, and running time of SNESIM, DISPAT, FILTERSIM, and DTL of ten reconstructions (shale and sandstone) are shown in Tables 8 and 9. The reason for splitting the "first-round running time" from the overall running time is that all four methods cost more time in their first-round running: SNESIM, DISPAT, and FILTERSIM scan all points or patterns in TIs, and DTL needs to use deep learning to train a basic model to learn all the structural features in TIs. However, after the "first-round running time," the time consumption of all four methods will be largely reduced because the training or learning processes have finished in the first round. It is seen that DTL is much better than other methods in the reconstruction time and memory demands due to the use of GPU.

4.3. Comparisons Using Different Conditioning Data. The results using four methods with 1% conditioning data (i.e., accounting for 1% of total voxels of the target image) are compared in Section 4.2. Since the porosity values of the target image and the training image are quite different, more conditioning data extracted from the target image can prove the applicability of the proposed method better. Hence, 5% conditioning data (i.e., accounting for 5% of total voxels of the target image) extracted from the target image for reconstruction were used to demonstrate the applicability of the proposed method. The pore voxels and grain voxels in the conditioning data have the same number. Variogram and MPC curves (average of ten reconstructions) of sandstone and shale using the conditioning data (5% of the total voxels) are shown in Figures 23–26.

It is seen that when there are more conditioning data (5% of total voxels), the variogram and MPC curves of the



(a) X direction

(b) Y direction

(c) Z direction

Figure 26: MPC curves of reconstructed shale images by 5% conditioning data using SNESIM, FILTERSIM, DISPAT, and DTL in three directions.

reconstructed DTL images are closer to those of the target image, proving the applicability of the proposed method.

4.4. Parameters Used for DTL. The practically used DTL network and some parameters will be discussed in this section. When using the DTL method, the network structure needs to be designed, including the number of hidden layers, the number of neural cells in the hidden layers, the optimization method, the activation function, and some other parameters, which will largely affect the training time, learning ability, accuracy, generalization ability, etc.

When designing the structure of DTL, the number of hidden layers can be determined by the learning effect, which is measured by the accuracy during the deep learning phase. The accuracy is calculated by comparing voxel values of the results from the training model with those of the TI during the deep learning phase. The accuracy (range is from 0 to 100%) is actually the similarity between the results in the deep learning phase with the TI, measuring whether the network structure can learn the structural features of training data quickly and accurately. High accuracy in the deep learning phase means that the network learns the features of training data well. However, high accuracy has little effect on the generalization ability of DTL and does not mean overfitting of the reconstruction because during the transfer phase conditioning data will be added for the constraints and the model will be trained again. The effect of overfitting on the whole DTL is small during the deep learning phase, but conditioning data have a significant impact on the generalization ability of DTL during the transfer learning phase. Therefore, the principle of selecting network parameters is to choose a network with a simple structure while ensuring accuracy.

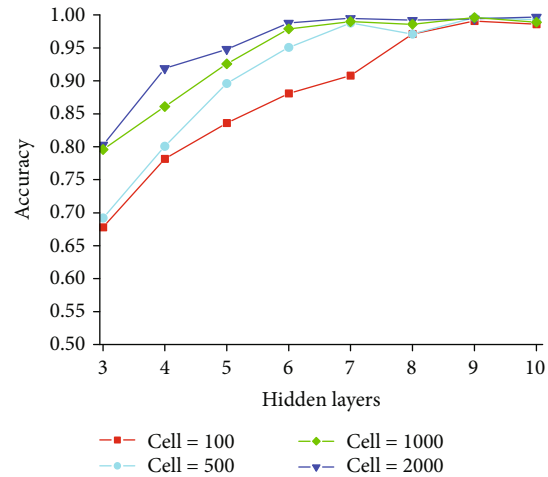


Figure 27: The accuracy using different numbers of hidden layers and neural cells.

As shown in Figure 27 and Table 10, more hidden layers and neural cells make the accuracy higher but need more training time. In our real experiments, the number of hidden

Table 10: The training time using different numbers of hidden layers and neural cells.

Number of hidden layers	3	4	5	6	7	8	9	10
Average training time (sec)								
Number of neural cells								
100	70	102	187	412	595	1106	2150	4010
500	76	126	198	468	650	1160	2695	4221
1000	82	139	229	512	786	1380	2960	4613
2000	91	151	312	566	960	1645	3319	4950

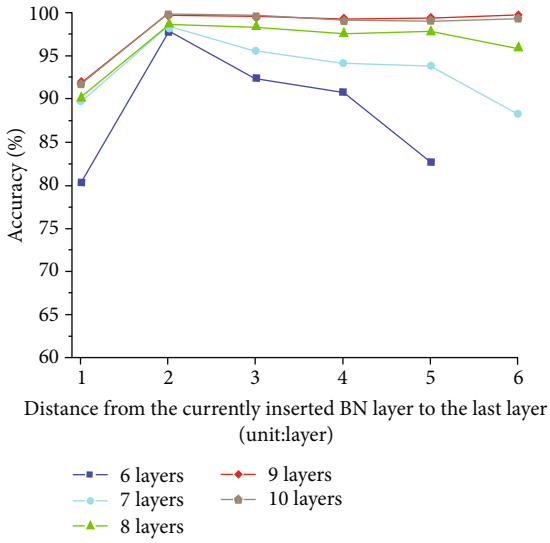


Figure 28: The accuracy when the BN layer is inserted in different locations of hidden layers.

layers is 8 and the number of neural cells is 500 to balance the training time and accuracy.

A BN layer will be inserted in transfer layers, but the specific location should be determined by experiments. A BN layer was inserted in the different locations of a DTL network to, respectively, form a 6-, 7-, 8-, 9-, and 10-layer network for a test. The accuracy during the deep learning phase is also, respectively, calculated. As shown in Figure 28, the abscissa ranging from 1 to 6 means the number of layers from the location of the inserted BN layer to the last layer, considered the "distance" (unit: layer) from the currently inserted BN layer to the last layer of the DTL network. For example, the accuracy is 80% when the inserted BN layer is next to the last layer (distance = 1) for a 6-layer DTL network in Figure 28; the accuracy is 98% when distance = 2 for an 8-layer DTL network. Since the number of hidden layers was 8 in the real experiments (see discussions about Figure 27 and Table 10) and the accuracy was close to 100% when distance = 2, the BN layer was added to the penultimate layer (i.e., distance = 2).

As for the optimization algorithm, the Adam-gradient descent method was used [40], which has faster convergence speed and more effective learning effect in practical applications. The weight w and the bias b should be randomly initialized. Learning rate α can be set to 0.5 at the beginning and will be adjusted to a smaller level after some training processes. The loss function in our experiments is the cross-entropy loss function for speeding up the training process and improving accuracy [13], which is defined as

$$L\left(y_i, y_i^{\text{pre}}\right) = -y_i \cdot \log y_i^{\text{pre}} - (1 - y_i) \cdot \log\left(1 - y_i^{\text{pre}}\right). \quad (24)$$

5. Conclusions

Statistical methods represented by MPS and some other methods are widely used for the reconstruction of high-resolution 3D porous media. However, the applicability of these methods is limited due to their large CPU cost and memory requirements. Meantime, the models established by them are not deterministic but a series of stochastic implementations with equal probabilities, so it is necessary to reconstruct models many times to obtain an average result.

Due to the rapid development of various machine learning technologies, it has become feasible to use deep learning to solve the problem in the reconstruction of 3D porous media. In this paper, a reconstruction method based on DTL is proposed, which is considered a combination of deep learning and transfer learning. Deep learning is used to learn the structural features of porous media, and then, transfer learning reproduces the features in new reconstructions.

Instead of reconstructing the unknown regions pixel by pixel or pattern by pattern, the proposed method learns the features and relationships between TIs and conditioning data first. The modeling process of the neural networks is a process of iterative optimization, making the errors gradually smaller and the results more certain rather than a series of equal probability results, which is also the reason that DTL has more stable results. This method has lower CPU costs and memory demands than the traditional MPS-like methods. The experiments of reconstructing the shale and sandstone images have proved the advantages of the proposed method in reconstruction quality, time consumption, and CPU utilization.

Data Availability

The data used to support this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 41672114 and 41702148).

References

[1] S. Bakke and P. E. Øren, "3D pore-scale modelling of sandstones and flow simulations in the pore networks," *SPE Journal*, vol. 2, no. 2, pp. 136–149, 2013.

[2] L. Mosser, O. Dubrule, and M. J. Blunt, "Reconstruction of three-dimensional porous media using generative adversarial neural networks," *Physical Review E*, vol. 96, no. 4, article 043309, 2017.

[3] T. Zhang, Y. Du, T. Huang, J. Yang, F. Lu, and X. Li, "Reconstruction of porous media using ISOMAP-based MPS," *Stochastic Environmental Research and Risk Assessment*, vol. 30, no. 1, pp. 395–412, 2016.

[4] C. H. Arns, F. Bauget, A. Limaye et al., "Pore scale characterization of carbonates using X-ray microtomography," *SPE Journal*, vol. 10, no. 4, pp. 475–484, 2005.

[5] M. J. Blunt, B. Bijeljic, H. Dong et al., "Pore-scale imaging and modelling," *Advances in Water Resources*, vol. 51, pp. 197–216, 2013.

[6] F. Mees, R. Swennen, M. V. Geet, and P. Jacobs, "Applications of X-ray computed tomography in the geosciences," *Geological Society, London, Special Publications*, vol. 215, no. 1, pp. 1–6, 2003.

[7] T. Zhang, Y. Du, T. Huang, and X. Li, "GPU-accelerated 3D reconstruction of porous media using multiple-point statistics," *Computational Geosciences*, vol. 19, no. 1, pp. 79–98, 2015.

[8] P. Tahmasebi, A. Hezarkhani, and M. Sahimi, "Multiple-point geostatistical modeling based on the crosscorrelation functions," *Computational Geosciences*, vol. 16, no. 3, pp. 779–797, 2012.

[9] J. Caers, "Direct sequential indicator simulation," in *Proceedings of the 6th International Geostatistics Congress*, pp. 39–48, Cape Town, South Africa, 2000.

[10] P. E. Øren and S. Bakke, "Process based reconstruction of sandstones and prediction of transport properties," *Transport in Porous Media*, vol. 46, no. 2/3, pp. 311–343, 2002.

[11] H. Okabe and M. J. Blunt, "Pore space reconstruction using multiple-point statistics," *Journal of Petroleum Science and Engineering*, vol. 46, no. 1-2, pp. 121–137, 2005.

[12] S. B. Strebelle, "Conditional simulation of complex geological structures using multiple-point statistics," *Mathematical Geology*, vol. 34, no. 1, pp. 1–21, 2002.

[13] G. B. Arpat and J. Caers, "Conditional simulation with patterns," *Mathematical Geology*, vol. 39, no. 2, pp. 177–203, 2007.

[14] T. Zhang, P. Switzer, and A. Journel, "Filter-based classification of training image patterns for spatial simulation," *Mathematical Geology*, vol. 38, no. 1, pp. 63–80, 2006.

[15] M. Honarkhah and J. Caers, "Stochastic simulation of patterns using distance-based pattern modeling," *Mathematical Geosciences*, vol. 42, no. 5, pp. 487–517, 2010.

[16] G. Mariéthoz and P. Renard, "Reconstruction of incomplete data sets or images using direct sampling," *Mathematical Geosciences*, vol. 42, no. 3, pp. 245–268, 2010.

[17] P. Renard, G. Mariethoz, and J. Straubhaar, "The direct sampling: a new way of performing multiple-points simulations," *EGU General Assembly Conference Abstracts*, vol. 11, p. 12187, 2009.

[18] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[19] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, https://arxiv.org/abs/1503.02531.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, Boston, MA, USA, June 2015.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, pp. 2672–2680, 2014.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, 2016.

[24] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[25] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: maximizing for domain invariance," 2014, https://arxiv.org/abs/1412.3474.

[26] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *2013 IEEE International Conference on Computer Vision*, pp. 2200–2207, Sydney, NSW, Australia, December 2013.

[27] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 17–36, Bellevue, Washington, USA, 2012.

[28] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: a deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 513–520, Bellevue, Washington, USA, 2011.

[29] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," 2015, https://arxiv.org/abs/1502.02791.

[30] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3320–3328, 2014.

[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning - ICML '08*, pp. 160–167, Helsinki, Finland, 2008.

[33] D. Yarotsky, "Error bounds for approximations with deep ReLU networks," *Neural Networks*, vol. 94, pp. 103–114, 2017.

[34] A. Griewank, J. Utke, and A. Walther, "Evaluating higher derivative tensors by forward propagation of univariate Taylor series," *Mathematics of Computation*, vol. 69, no. 231, pp. 1117–1131, 2000.

[35] M. Courbariaux, Y. Bengio, and J. P. David, "Binaryconnect: training deep neural networks with binary weights during propagations," *Advances in Neural Information Processing Systems*, vol. 28, pp. 3123–3131, 2015.

[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings*

of the thirteenth international conference on artificial intelligence and statistics, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 2010.

[37] E. A. Wan and F. Beaufays, "Diagrammatic derivation of gradient algorithms for neural networks," *Neural Computation*, vol. 8, no. 1, pp. 182–201, 1996.

[38] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018.

[39] T. Van Laarhoven, "L2 regularization versus batch and weight normalization," 2017, https://arxiv.org/abs/1706.05350.

[40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, https://arxiv.org/abs/1207.0580.

[41] M. Abadi, P. Barham, J. Chen et al., "Tensorflow: a system for large-scale machine learning," in *12th Symposium on Operating Systems Design and Implementation*, pp. 265–283, Savannah, GA, USA, 2016.

[42] M. S. Costanza-Robinson, B. D. Estabrook, and D. F. Fouhey, "Representative elementary volume estimation for porosity, moisture saturation, and air-water interfacial areas in unsaturated porous media: data quality implications," *Water Resources Research*, vol. 47, no. 7, p. W07513, 2011.

[43] M. Oda, "A method for evaluating the representative elementary volume based on joint survey of rock masses," *Canadian Geotechnical Journal*, vol. 25, no. 3, pp. 440–447, 1988.

[44] R. al-Raoush and A. Papadopoulos, "Representative elementary volume analysis of porous media using X-ray computed tomography," *Powder Technology*, vol. 200, no. 1-2, pp. 69–77, 2010.

[45] K. Nordahl and P. S. Ringrose, "Identifying the representative elementary volume for permeability in heterolithic deposits using numerical rock models," *Mathematical Geosciences*, vol. 40, no. 7, pp. 753–771, 2008.

[46] S. Krishnan and A. G. Journel, "Spatial connectivity: from variograms to multiple-point measures," *Mathematical Geology*, vol. 35, no. 8, pp. 915–925, 2003.

[47] M. Singh and K. K. Mohanty, "Permeability of spatially correlated porous media," *Chemical Engineering Science*, vol. 55, no. 22, pp. 5393–5403, 2000.

[48] H. Okabe and M. J. Blunt, "Prediction of permeability for porous media reconstructed using multiple-point statistics," *Physical Review E*, vol. 70, no. 6, article 066135, 2004.

[49] C. FEI, *Avizo 9 User's Guide*, FEI Corporation, 2009.

[50] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, https://arxiv.org/abs/1609.04747.

[51] P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.