Physics of Fluids ARTICLE scitation.org/journal/phf

Point-cloud deep learning of porous media for permeability prediction

Cite as: Phys. Fluids 33, 097109 (2021); doi: 10.1063/5.0063904
Submitted: 18 July 2021 · Accepted: 26 August 2021 ·
Published Online: 28 September 2021

View Online Export Citation CrossMark

Ali Kashefi[1,a] and Tapan Mukerji[2,b]

AFFILIATIONS

[1]Department of Civil and Environmental Engineering, Stanford University, Stanford, California 94305, USA
[2]Department of Energy Resources Engineering, Stanford University, Stanford, California 94305, USA

[a]Author to whom correspondence should be addressed: kashefi@stanford.edu
[b]Electronic mail: mukerji@stanford.edu

ABSTRACT

We propose a novel deep learning framework for predicting the permeability of porous media from their digital images. Unlike convolutional neural networks, instead of feeding the whole image volume as inputs to the network, we model the boundary between solid matrix and pore spaces as point clouds and feed them as inputs to a neural network based on the PointNet architecture. This approach overcomes the challenge of memory restriction of graphics processing units and its consequences on the choice of batch size and convergence. Compared to convolutional neural networks, the proposed deep learning methodology provides freedom to select larger batch sizes due to reducing significantly the size of network inputs. Specifically, we use the classification branch of PointNet and adjust it for a regression task. As a test case, two and three dimensional synthetic digital rock images are considered. We investigate the effect of different components of our neural network on its performance. We compare our deep learning strategy with a convolutional neural network from various perspectives, specifically for maximum possible batch size. We inspect the generalizability of our network by predicting the permeability of real-world rock samples as well as synthetic digital rocks that are statistically different from the samples used during training. The network predicts the permeability of digital rocks a few thousand times faster than a lattice Boltzmann solver with a high level of prediction accuracy.

I. INTRODUCTION AND MOTIVATION

The importance of study of porous media in a wide range of scientific and industrial fields such as digital rock physics,[1,2] membrane systems,[3] geological carbon storage,[4] and medicine[5] in the past decades has led to a growth in collection of pore-scale image data. Along with pore-scale imaging, there has been a growth in the use of numerical computation to assess physical and transport properties of porous media based on the image data. Such a revolution in the age of data has motivated the use of machine learning schemes as a data-driven strategy to accelerate the computations for understanding the physical properties of porous media. Among different machine learning techniques, deep learning has been widely used in various applications for the study of porous media. A few specific applications are rock image segmentation[6–8] and predicting physical properties and geometrical features such as permeability,[9–16] porosity,[9,17–20] effective diffusivity,[21] wave propagation velocities,[22] and fluid flow fields.[23,24] It is worth noting that arguments and ideas proposed in this article are general and usable for any desired porous media such as biological tissues and ceramics; however, we restrict ourselves to the applications of rocks in subsurface aquifers and petroleum reservoirs. Specifically, our focus in the present article is on deep learning frameworks for predicting permeability from digital rock images.

Convolutional neural networks (CNNs) have been used extensively to predict the permeability of digital rock images (see, e.g., Refs. 11, 12, and 14). In this setup, CNNs are trained on a set of labeled data to learn a mapping from two (2D) or three dimensional (3D) digital rock images to rock permeability. Generally speaking, a common challenge in using CNNs is the Graphics Processing Unit (GPU) memory required for training CNNs.[25] This challenge is magnified in large, deep, and three dimensional CNNs.[25] Limitation on the memory of available GPU memory might lead to a restriction on the "batch size" (see, e.g., Ref. 26 for the technical definition of "batch size"). Contrary to the technique of stochastic gradient descent, the mini batch gradient descent method accelerates the training procedure mainly by vectorization. However, the associated batch size affects the performance of deep neural networks.[27–30] Hence, it is vital to have freedom to choose

the optimal batch size. To overcome the above-mentioned challenge, we propose a new machine learning architecture, which is based on the deep learning of point cloud data. Next, we explain the key idea of our methodology.

Mathematically, the permeability of a porous medium is a function of the velocity fields in the pore-space of the medium. The solution of the governing equations of fluid flow in porous media (e.g., reservoir rocks) is a function of the geometry of the grain–pore boundary and the boundary conditions. Thus, if the geometry of the grain–pore boundary can be used as an input representation to the neural network, we do not need either the volume of the grain spaces or pore spaces anymore. To reach this goal, for a given porous medium, we only take the grain–pore boundary and represent it as a set of points, constructing a point cloud (see Fig. 1). Points on the surface (in three dimensional geometries) or on the edge (in two dimensional geometries) of this cloud represent the geometry of the grain–pore boundaries. Representing digital rocks as sparse point clouds instead of full two or three dimensional image pixels or voxels dramatically diminishes the size of memory required to be allocated on GPUs. Additionally, it provides users with more freedom to select the batch size.

Since we represent the boundary of the pore space as a point cloud, a point-cloud-based deep learning framework is required. From a computer science point of view, several architectures are available for this purpose (see, e.g., Refs. 31–33). Among these options, PointNet[32] has been widely used for deep learning of point cloud data for classification and segmentation of two and three dimensional objects in computer vision and computer graphics (see, e.g., Refs. 34 and 35). Qi *et al.*[32] first introduced PointNet in 2017, and the network has quickly become popular for both industrial and academic applications such as object detection,[35,36] shape reconstruction,[37] camera pose estimation,[37] and physical simulation.[38,39]

To the best of our knowledge, PointNet[32] has been already used twice for applications outside of the pure computer science areas. First, the performance of PointNet[32] for predicting the velocity and pressure fields of incompressible flows on irregular geometries has been examined by Kashefi *et al.*[38] Kashefi *et al.*[38] adjusted the PointNet architecture[32] to predict the flow fields around a cylinder with various shapes for its cross section and obtained an excellent to reasonable level of accuracy. Additionally, they[38] demonstrated the generalizability of their proposed neural network by predicting the velocity and pressure fields on unseen category data such as multiple objects and airfoils (see Figs. 13–19 of Ref. 38). Second, DeFever *et al.*[40] employed PointNet[32] to identify local structures in molecular simulations. These successes[38,40] motivate us to utilize the PointNet[32] architecture and modify it for our own application. To accomplish this task, we use the classification component of PointNet[32] and replace its cross-entropy cost function by the mean squared error to establish an end-to-end mapping from a point cloud (as input) to the corresponding permeability (as output) framed as a regression problem. It is important to mention that we utilize PointNet[32] for the first time for a regression problem. Although our focus in this article is on permeability prediction in porous media, our approach can be potentially used for any other machine learning problems, where the output of interest is a real number that is a function of the geometry of spatial domains. Further details of our neural network are described in Sec. II C 1.

We assess the prediction performance of the network, its sensitivity to different parameters and activation functions, and its computational efficiency in several ways. First, to evaluate prediction performance, we report the coefficient of determination as well as the maximum and minimum relative errors of the predicted permeability with reference to the permeability calculated from a numerical solver for a set of two and three dimensional porous medium geometries. Second, to assess the sensitivity to different parameters, we discuss the number of points in point clouds as a hyperparameter of the neural network proposed in this article. We evaluate the effect of input and feature transform blocks in PointNet[32] on the performance of the deep learning framework. Additionally, we explore the influence of different activation functions and different sizes of latent global feature on the accuracy of the predicted permeability, and test the neural network generalizability. Finally, we compute the speed-up factor obtained by the proposed neural network compared to a conventional numerical solver for flow simulation in pore spaces as well as compare the performance of the point-cloud neural network with a regular CNN.

The rest of this paper is structured as follows. We describe the governing equations of fluid flows in porous media and techniques for

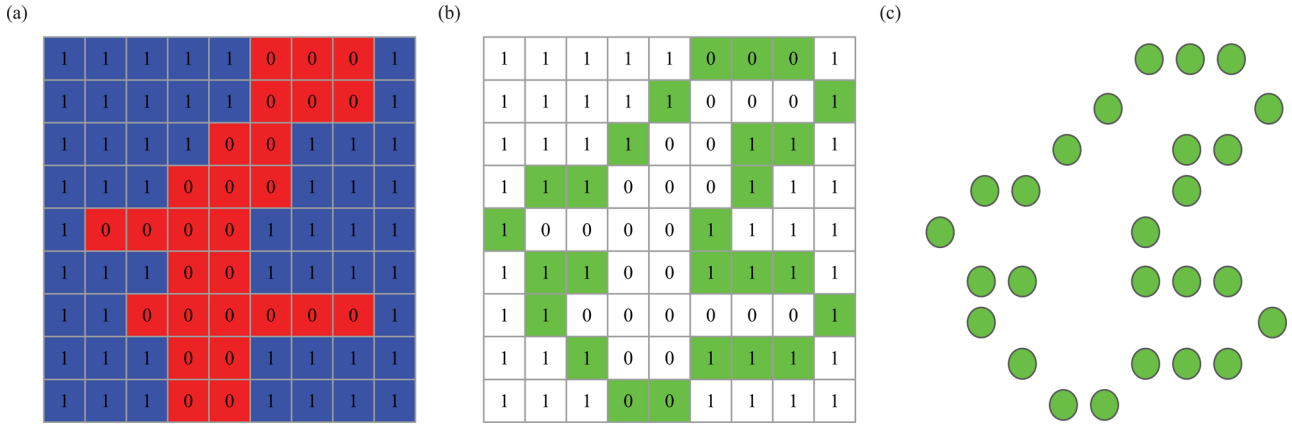

FIG. 1. Schematic illustration of the algorithm for constructing point clouds: (a) voxel representation, 0 (red) and 1 (blue) indicate, respectively, pore and grain spaces, (b) pore–grain space boundary identification, (c) point cloud representation; the green boundary is specified by a set of points.

permeability computations using numerical solvers in Sec. II A. Data generation for deep learning is explained in Sec. II B. We illustrate and compare the architecture of our neural network with a CNN in Sec. II C. Network training is illustrated in Sec. II D. An analysis of the network performance for two dimensional geometries is provided in Sec. III A. Prediction of the permeability in three dimensional porous media is investigated in Sec. III B. Alternative approaches for permeability prediction and potentials of our neural network in this regard are discussed in Sec. III C. Section IV summarizes and concludes the study.

II. PROBLEM FORMULATION AND METHODOLOGY

A. Permeability computation in porous media

To compute the permeability of a porous medium, first we obtain the velocity field of fluid flow in the pore space of the medium. The continuity and Navier–Stokes equations govern the dynamics of single-phase incompressible flow within the pores of a porous medium and are written as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \text{ in } V, \tag{1}$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p - \mu \Delta \boldsymbol{u} = \boldsymbol{f} \text{ in } V, \tag{2}$$

where $\boldsymbol{u}$ and p indicate, respectively, the velocity vector and absolute pressure in the space of V, the fluid-filled pore space. The fluid density and dynamic viscosity are shown by ρ and μ, respectively. The vector of external body force is indicated by $\boldsymbol{f}$. We consider the porous medium domains to be squares (in two dimensional spaces) or cubes (in three dimensional spaces) with length L along each principal axis, porosity of ϕ, and spatial correlation length of l_c. A pressure gradient in the x direction (dp/dx) is applied to stimulate the flow in the medium. No flow boundary condition is enforced at the top and bottom of the medium on the y–z planes. Periodic boundary conditions are applied at the inflow and outflow velocity boundaries parallel to the pressure gradient direction. A numerical solver based on the Lattice Boltzmann Method (LBM) is used to obtain the steady state solution to the governing equations. More details of the analysis are discussed in Ref. 41. After calculating the flow velocity in the pore space of the porous medium, the permeability in x direction (k) is obtained from Darcy's law,[42]

$$k = -\frac{\mu \bar{U}}{dp/dx}, \tag{3}$$

where $\bar{U}$ is the mean velocity in the entire porous medium including grain spaces. Note that Eq. (3) is only valid for low Reynolds numbers (see, e.g., Ref. 43).

B. Data generation

To have a robust control on training data and investigate the effect of different geometrical parameters such as porosity (ϕ) and spatial correlation length (l_c) in porous media, we synthetically generate our data set such that it represents a range of heterogeneity of reservoir rocks. Similar approaches have been taken by Wu *et al.*[12] and Da Wang *et al.*[23] To generate a synthetic binary (pore–grain) medium with a targeted porosity (ϕ) and spatial correlation (l_c), a straightforward algorithm of truncated Gaussian simulation[44,45] is used by truncating spatially correlated Gaussian random fields created by a moving average filter applied to random uncorrelated Gaussian noise. The algorithm is implemented as follows. First, we consider two and three dimensional arrays, respectively, with the size of n^2 and n^3. Next, uncorrelated random variables with the standard normal distribution are assigned to the array elements. Afterwards, Gaussian kernels with different kernel sizes are applied as a filter introducing spatial correlation. In the next stage, the numerical values of the arrays are normalized in the range of [0, 1], and thresholded to give binary arrays with desired ranges of porosity (i.e., see Figs. 2 and 3). Arrays with no correlated fields are discarded. In this work, we set $L = n \times \delta x$, where δx is the size of each pixel side and equal to 0.003 m. Concerning two dimensional porous media, we set $n = 128$ and synthetically generate data with three representative spatial correlation lengths (kernel of the Gaussian filter) of 9, 17, and 33 pixels while considering the porosity (ϕ) in the range of [0.125, 0.25). We use 2600 data samples with a spatial correlation length (l_c) of 9 for training, validation, and test purposes, while data with spatial correlation lengths (l_c) of 17 and 33 are used for the investigation of neural network generalizability. The mean (and standard deviations) for the porosity and permeability of the training and test set of the two dimensional porous media are as follows: training set porosity, 0.181 (0.03); test set porosity, 0.185 (0.04); training set permeability, 121.62 mD (18.42 mD); test set permeability, 128.52 mD (20.95 mD). Concerning three dimensional porous media, we generate data with $n = 64$ and spatial correlation length (kernel of the Gaussian filter) of 17 pixels, while the porosity (ϕ) in the range of [0.125, 0.20) is selected. A total of 2175 samples are generated for use in training, validation, and testing. The mean (and standard deviations) for the porosity and permeability of the training and test set of the three dimensional porous media are as follows: training set porosity, 0.146 (0.04); test set porosity, 0.151 (0.05); training set permeability, 67.12 mD (58.03 mD); test set permeability, 69.83 mD (61.12 mD). We consider a real 3D CT-scan image of a rock sample to carry out the generalizability level of our neural network. A set of Python codes and batch files automates the process of generating synthetic data. The LBM solver is run on all of the generated synthetic porous media to get the corresponding permeabilities, thus creating a labeled dataset.

The next step is to define the neural network domain (V_{NN}). Indicating the grain-pore boundary by ∂V, then mathematically, $V_{NN} \subset \partial V$. In fact, V_{NN} must represent the geometry of the grain–-pore boundary. Note that by keeping the physical properties of the fluid (i.e., viscosity and density) and the boundary conditions fixed over all the generated data, the solution of the governing equations is only a function of the geometry of the boundary of the pore space ∂V, and consequently V_{NN}. V_{NN} contains N points. The challenge is that the number of pixels located on ∂V varies from one data sample to another. Thus, N is a hyperparameter in our deep learning framework. We discuss the effect of the choice of N on our neural network performance in Sec. III A. Figures 2 and 3 depict, respectively, digital porous medium images and their resulting point clouds (V_{NN}) for two and three dimensions. Note that our deep learning methodology is not limited to constructing V_{NN} from digital images. One may use scattered data obtained on unstructured finite element or finite volume grids to establish V_{NN}.

To accelerate the convergence of our neural network training and equalize the contribution of each input component to the training of neural network parameters (e.g., weights and bias), the input and

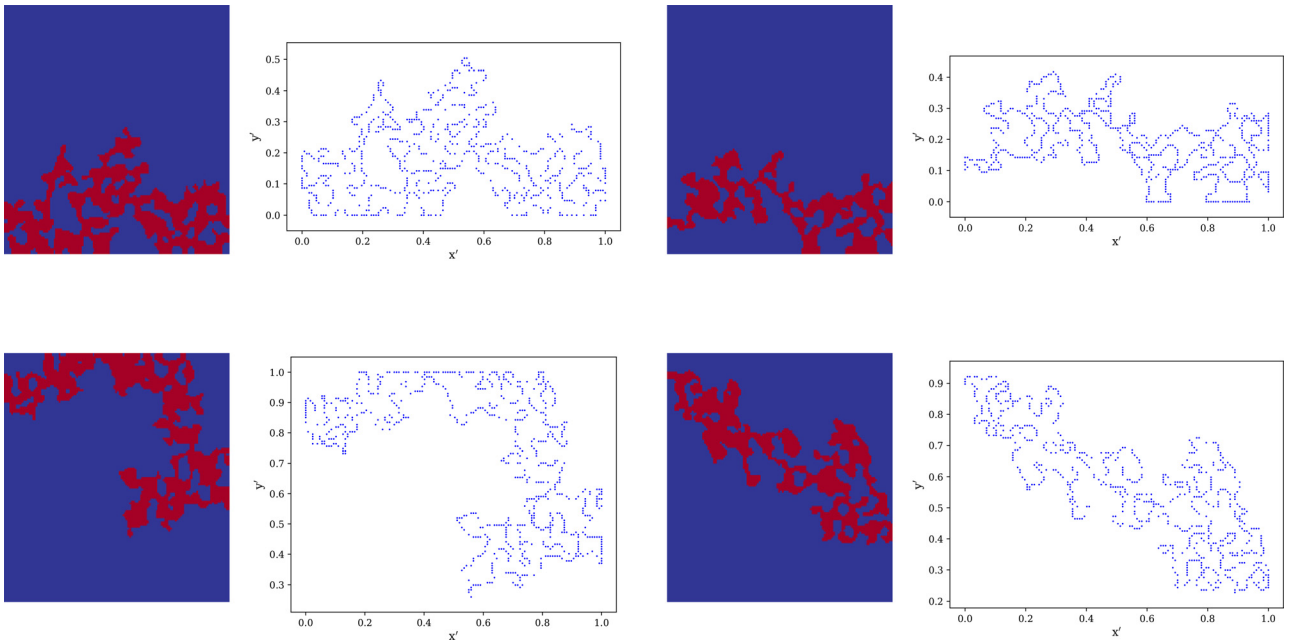

FIG. 2. Two dimensional digital porous medium images and their corresponding point-cloud representations; digital images and point clouds are used to train CNN and the point-cloud neural network, respectively.

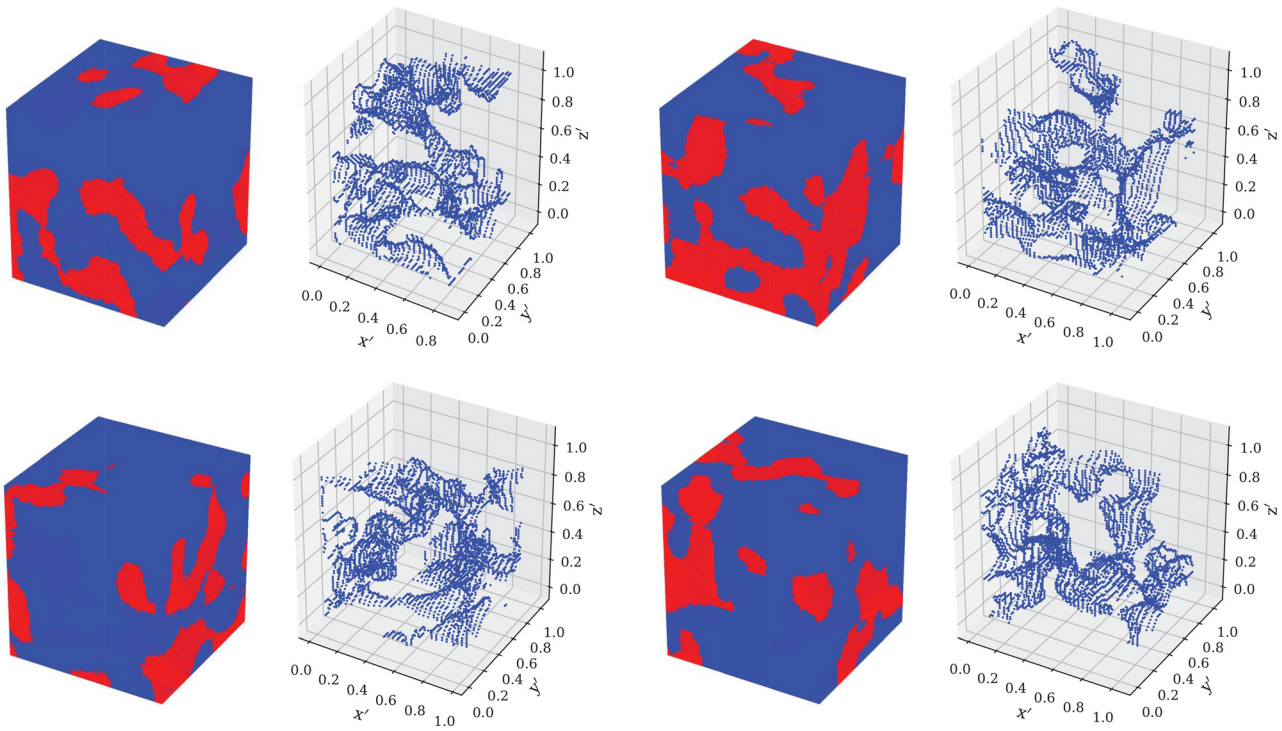

FIG. 3. Three dimensional digital porous medium images and their corresponding point-cloud representations; digital images and point clouds are used to train CNN and the point-cloud neural network, respectively.

output data are scaled in the range of [0, 1] using the maximum and minimum values of each set of k, x, y, and z. We indicate the scaled set by k', x', y', and z'. As an example, k' is computed as follows:

$$k' = \frac{k - \min(k)}{\max(k) - \min(k)}. \tag{4}$$

x', y', and z' are computed similarly. Obviously, k', x', y', and z' are dimensionless.

C. Neural network architectures

1. Point-cloud neural network

Our neural network is mainly based on the PointNet[32] architecture. In this subsection, we briefly describe the point-cloud neural network. One may refer to Ref. 32 for further explanations. In this subsection, the vectors and matrices of machine learning components are shown by bold letters but not italic. This is to distinguish the machine learning vectors and matrices from the physics-based ones. The two main components are Multilayer Perceptron (MLP) and Fully Connected (FC) layer. An MLP is constructed by several sequential FC layers. We use notation in the form of (A_1, A_2) to show an MLP with two layers, where A_1 and A_2 are, respectively, the size of the first and second layer. Notations in the form of (A_1, A_2, A_3) are similarly defined. In the current study, the point-cloud neural network is restricted to MLPs with two and three layers. We parameterize each FC layer by a weight matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$. The size of an FC layer indicates the number of rows in the corresponding matrix $\mathbf{W}$. Mathematically, a recursive function connects the input of ith FC layer $\mathbf{a}_i$ to the output of $i-1$th FC layer $\mathbf{a}_{i-1}$ such that

$$\mathbf{a}_i = \sigma(\mathbf{W}_i\mathbf{a}_{i-1} + \mathbf{b}_i), \tag{5}$$

where σ is a nonlinear activation function. The activation function is applied elementwise to each vector component.

Consider two sets $\mathcal{X}$ and $\mathcal{Y}$, the network inputs and the desired target, respectively, with $\mathcal{X} = \{\mathrm{x}_i \in \mathbb{R}^d\}_{i=1}^{N}$ and $\mathcal{Y} = \{\mathrm{y}_i \in \mathbb{R}\}_{i=1}^{n_p}$, where d corresponds to the spatial dimension (2 or 3) and n_p is the number of desired physical or geometrical quantities of interest as the targets of the network. When predicting permeability alone, n_p is 1. We wish to design a neural network to map $\mathcal{X}$ to $\mathcal{Y}$ by an operator f such that $\mathcal{Y} = f(\mathcal{X})$. Two fundamental concepts need to be considered in the design. First, the output set $\mathcal{Y}$ is a function of the geometrical features of the input set $\mathcal{X}$. Thus, the operator f must be able to capture the geometrical features. Second, since the input set $\mathcal{X}$ essentially represents an unstructured and unordered point cloud, the operator f must be invariant with respect to the order of input points x_i of the set $\mathcal{X}$. The PointNet[32] architecture provides these two critical features. Hence, we approximate the operator f by a PointNet-based neural network that learns the mapping from $\mathcal{X}$ to $\mathcal{Y}$ through a set of labeled data described in Sec. II B.

The structure of the point-cloud neural network is depicted in Fig. 4. As can be seen in Fig. 4, the network has two main branches: one before and another after the global feature. The first branch encodes the geometrical feature of the input set $\mathcal{X}$ in a latent global feature with a vector of size 1024. The second branch decodes the latent global feature to predict the permeability. Two Transform Nets (T-Nets) exist in the first branch. The first T-Net transforms the input set $\mathcal{X}$ into an implicit canonical space, while the second T-Net is used for an affine transformation for alignment in the input set $\mathcal{X}$. From a machine learning perspective, T-Nets can be viewed as mini PointNets that consist of an MLP component (64, 128, 1024) followed by a max pooling operator to extract the underlying features. The feature can then be decoded by two MLP components each with two layers (512, 256). One may refer to Ref. 32 for further descriptions of T-Nets. In addition to T-Nets, two MLP components contribute to the construction of the first branch: the first (64, 64) and the second (64, 128, 1024). Mathematically, PointNet[32] encodes the geometrical features of the point set such that the latent code is independent of ordering over the set of points. In other words, to aggregate information over the input set $\mathcal{X}$, a permutation invariant function such as maximum, minimum, average, and summation is necessary. PointNet[32] uses the "max" function to handle it. We represent all the mathematical operations carried out on the input set $\mathcal{X}$ just before the max pooling operator by a function h. Thus, the latent global feature is established on the input set $\mathcal{X}$ by a function g such that

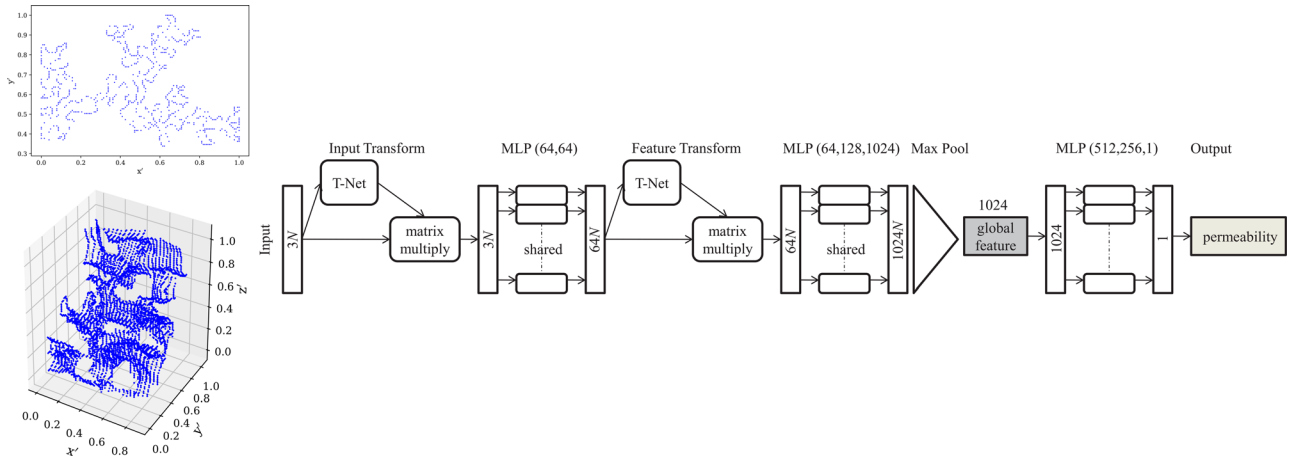

FIG. 4. Structure of the proposed point-cloud neural network based on PointNet;[32] the network input is the point cloud representing the boundary line or boundary surface of grain–pore spaces, respectively, for two and three dimensional porous media.

$$g(\mathcal{X}) \approx \max(h(\mathrm{x}_1), \ldots, h(\mathrm{x}_N)). \tag{6}$$

As can be observed in Fig. 4, an MLP component with three layers (512, 256, 1) in the second branch is used to predict the permeability. Note that all the MLP components in the first branch have shared weights, while this is not the case for the MLP component in the second branch. This is another key feature of PointNet[32] to handle unordered points in the set $\mathcal{X}$, and this is why we use the single function h for all the input points x_i in Eq. (6). In fact, it does not matter how the input set $\mathcal{X}$ is constructed for feeding it to our neural network as PointNet[32] treats all x_i in a same manner due to the shared weights of MLPs in the first branch. After each FC layer, a batch normalization[46] operator is used. The activation function used for all the layers is the Rectified Linear Unit (ReLU) defined as

$$\sigma(\gamma) = \max(0, \gamma), \tag{7}$$

except for the last layer where we employ a sigmoid function expressed as

$$\sigma(\gamma) = \frac{1}{1 + e^{-\gamma}}. \tag{8}$$

To close this subsection, we address a few points. First, we set $d = 3$ for the permeability prediction in two dimensional pore spaces by assigning zero values to the third axis. Alternatively, one may set $d = 2$ for two dimensional porous media and adapt the size of network layers accordingly. Second, we restrict our current study to the prediction of the permeability (i.e., $n_p = 1$); however, one may adjust n_p for the prediction of other quantities of interest such as porosity, average pore size, and specific surface area (see, e.g., Ref. 17).

2. Convolutional neural networks

We briefly explain the architecture of the CNNs designed for predicting permeability from two and three dimensional digital rock images. We skip describing the technical details used in this subsection, and we encourage potential audiences with interest in use of CNNs for permeability prediction to read Sec. 2.3 of Ref. 11. Similar to PointNet,[32] we need an encoder to extract the image features and a decoder, which maps the learned features to the corresponding permeability. We employ the encoder structure of DCGAN,[47] which is a highly cited and successful unsupervised generative adversarial network. Accordingly, ReLU activation function is used for all layers, and no pooling layer is utilized. The number of filters starts with 16 and doubles at each convolution layer, sequentially. All convolution layers are set with no padding, a stride size of 2, and kernel size of (2, 2) and (2, 2, 2), respectively, for the two and three dimensional CNNs, except in the last layer of the three dimensional CNN, which has a kernel size of (1, 1, 1). This is to enforce a latent global feature with the size of 1024 (see further discussions in Sec. II C 3). We use the PointNet[32] decoder for both the two dimensional (2D-CNN) and three dimensional CNN (3D-CNN). As an example, Fig. 5 depicts the architecture of 2D-CNN used in this study.

3. Comparison between PointNet and CNNs

A fair comparison between PointNet[32] and a CNN is not straightforward. First, each of them is based on different underlying mathematical and computational theories. Second, PointNet[32] has a unique structure, whereas we can find many neural networks, which are based on the convolution operation (see, e.g., Refs. 48 and 49) and fall in the category of CNNs. Moreover, neural networks with the convolution operation are usually combined with other functions such as max-pooling (see, e.g., Ref. 50), upsampling (see, e.g., Ref. 51), and skip connection (see, e.g., Ref. 52). Thus, a variety of CNNs with different performance can be implemented. With these in mind, we have enforced two conditions for designing the CNN introduced in Sec. II C 2 to make it similar to the PointNet[32] architecture as much as possible. First, the size of latent global feature of both 2D-CNN and PointNet[32] is equal to 1024. Second, both networks use the same decoder. This makes it easier to have a consistent comparison.

D. Training

The first step in the training process is to select an appropriate cost function (or loss function). The mean squared error function has been widely used in the area of deep learning of computational mechanics (see, e.g., Ref. 53) as well as in porous media applications (see, e.g., Refs. 11–13, 23, and 24). In the current study, we utilize this function defined as

$$\mathcal{C} = \frac{1}{M} \sum_{i=1}^{M} (k_i' - \tilde{k}_i')^2, \tag{9}$$

where M is the number of data in our training set. We label the permeability (k') obtained by the LBM solver as the "ground truth," while we denote the predicted permeability by $\tilde{k}'$. After training, we rescale the predicted permeability ($\tilde{k}'$) back into to the physical domain ($\tilde{k}'$) for a

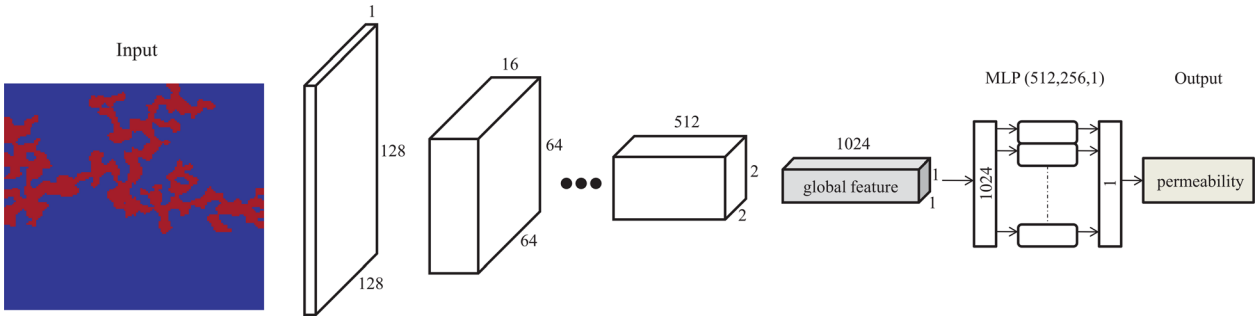

FIG. 5. Structure of the 2D-CNN used for learning two dimensional porous media.

post-processing analysis. We use the Adam[54] optimizer with hyperparameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\hat{\epsilon} = 10^{-6}$. To understand the mathematical definition of β_1, β_2, and $\hat{\epsilon}$, one may refer to Ref. 54. For two dimensional cases, our generated data are categorized into three sets for training (2300 data), validation (150 data), and test (150 data) through a random selection process. Similarly for three dimensional cases, we have three sets of training (1745 data), validation (215 data), and test (215 data). The validation data set is mainly used to track the convergence rate of the training process and to avoid overfitting. A systematic procedure through a grid search is undertaken to determine the network hyperparameters. Accordingly, the learning rates of $\alpha = 0.07$ for two dimensional cases and $\alpha = 0.1$ for three dimensional cases with an exponential decay with the rate of 0.1 provide the optimal choice based on the test cost ($\mathcal{C}$). Using high learning rates (α) in neural networks for the permeability prediction has been reported by other researchers as well (e.g., see Fig. 10 of Ref. 14). We use NVIDIA Tesla V100 graphics card with the memory clock rate of 1.41 GHz and 24 Gigabytes of RAM for the training process. This procedure takes approximately 30 min and 2 h, respectively, for two and three dimensional porous media. Note that we only provided the details of training the point-cloud neural network in this subsection. A similar procedure has been taken to obtain the highest possible performance for the CNN discussed in Sec. II C 2.

III. RESULTS AND DISCUSSION

A. Two dimensional porous media

The coefficient of determination, namely, R^2 score, is a commonly used metric to evaluate the performance of predicting the permeability (see, e.g., Refs. 11, 12, 14, and 15) and is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{P} (k_i - \tilde{k}_i)^2}{\sum_{i=1}^{P} (k_i - \bar{k})^2}, \tag{10}$$

where P is the number of data in the test set and $\bar{k}$ denotes the mean of the $\{k_i\}_{i=1}^{P}$ set. We use this metric in the current work.

The first step in the analysis of our results is to discuss the choice of N. As pointed out in Sec. II C 1, N is a hyperparameter of our deep learning framework. There is no restriction on N, and users of our machine learning platform can tune it to search for the highest achievable performance. For our current digital rock images, the number of points located on the pore–grain boundaries varies between $N_{\min} = 673$ and $N_{\max} = 1698$. To construct $\mathcal{X}$ when $N = N_{\min}$, we randomly select $N_{\min}$ points from each point cloud of our data set. Similarly, to establish $\mathcal{X}$ when $N = N_{\max}$, we add some extra points to point clouds by randomly repeating some of their own points to fill them up to $N_{\max}$. Additionally, one may select N such that $N_{\min} < N < N_{\max}$. Furthermore, there is no restriction on the selections of $N_{\max} < N$ or $N < N_{\min}$ although they do not seem reasonable choices, unless one intends to reduce the network size due to a memory limitation by the choice of $N < N_{\min}$. Figures 6(a) and 6(b), respectively, depict the examples of point cloud illustrations for the choices of $N_{\min}$ and $N_{\max}$ and their corresponding R^2 score plots. As can be realized from Figs. 6(a) and 6(b), selection of $N = N_{\min}$ results in a higher R^2 score compared to $N = N_{\max}$ (0.962 22 vs 0.925 36). From the above described algorithms, we argue that because the $\mathcal{X}$ set contains redundant data in the case of $N = N_{\max}$, it might cause a deviation in the path of network learning for finding the minimum in the space of the cost function. We also find the R^2 scores of 0.904 92 and 0.874 669 for $N = 1000$ and $N = 1300$, respectively.

Figure 6(c) exhibits the resulting prediction of permeability using the 2D-CNN introduced in Sec. II C 2. Compared to our deep learning strategy with $N = N_{\min}$, a 6.714% decrease in the R^2 score is observed (0.962 22 vs 0.897 61). A comprehensive comparison between the point-cloud neural network and 2D-CNN is made in Table I. Based on the information tabulated in Table I, 2D-CNN experiences higher minimum and maximum relative errors compared to the PointNet based network. Additionally, the size of input vector in CNN increases approximately by a factor of 9, leading to a higher GPU memory requirement. More importantly, the maximum possible batch size on our computational facilities for 2D-CNN is 1024, whereas the point-cloud neural network is able to load all 2300 training data in one epoch. It is conjectured that this is the main reason for a lower performance of 2D-CNN compared to the point-cloud neural network.

Figures 7(a) and 7(b) illustrate the geometries with the minimum relative errors for the point-cloud neural network and 2D-CNN, while Figs. 7(c) and 7(d) exhibit the geometries with the maximum relative errors for these networks, respectively. As can be inferred from Fig. 7, these extremums happen in different geometries for these two networks. It means that each of these two networks has been optimized in two different minima in the high dimensional space of the cost function. Note that we usually do not deal with convex optimization problems in the field of machine learning.[55] However, because both the maximum and minimum relative errors of the point-cloud neural network are smaller than the corresponding errors of 2D-CNN (see Table I), we conclude that the point-cloud neural network is more successful than 2D-CNN to solve the associated optimization problem. Note that as discussed in Sec. II C, here we report the highest possible performance obtained for each network by a grid search on their hyperparameters. We emphasize on the fact that the goal of this research paper is not to prove that the proposed network can "definitely" gain a higher score than any existing CNN-based networks. For instance, one may argue that one can adjust the 2D-CNN proposed in Sec. II C 2 by making it deeper to reach a higher score compared to our new neural network. Instead, we claim that the PointNet based network with less training efforts and less memory allocations still can compete and outperform CNN-based networks in many cases.

As explained in Sec. II B, we normalize the permeability in the range of [0, 1] for training the network along with the sigmoid activation function [see Eq. (8)] in the last layer of the neural network to cover that range. Our primary motivation to use this approach is that Kashefi *et al.*[38] have taken the same procedure for predicting real continuous variables such as velocity and pressure. However, since the permeability is a positive real number, another option would be to keep the permeability in the physical domain and use the ReLU activation function [see Eq. (7)] in the last layer. This option has been used by several researchers such as Hong and Liu[11] and Tembely *et al.*[14] We implement the latter option to compare these two strategies. The outcome of using the ReLU function [see Eq. (7)] is illustrated in Fig. 8(a). A comparison between Figs. 8(a) and 6(a) indicates a higher R^2 score for our current approach (i.e., using the sigmoid activation function). Note that the scatter in Figs. 6 and 8 is quantified by the R^2 scores.

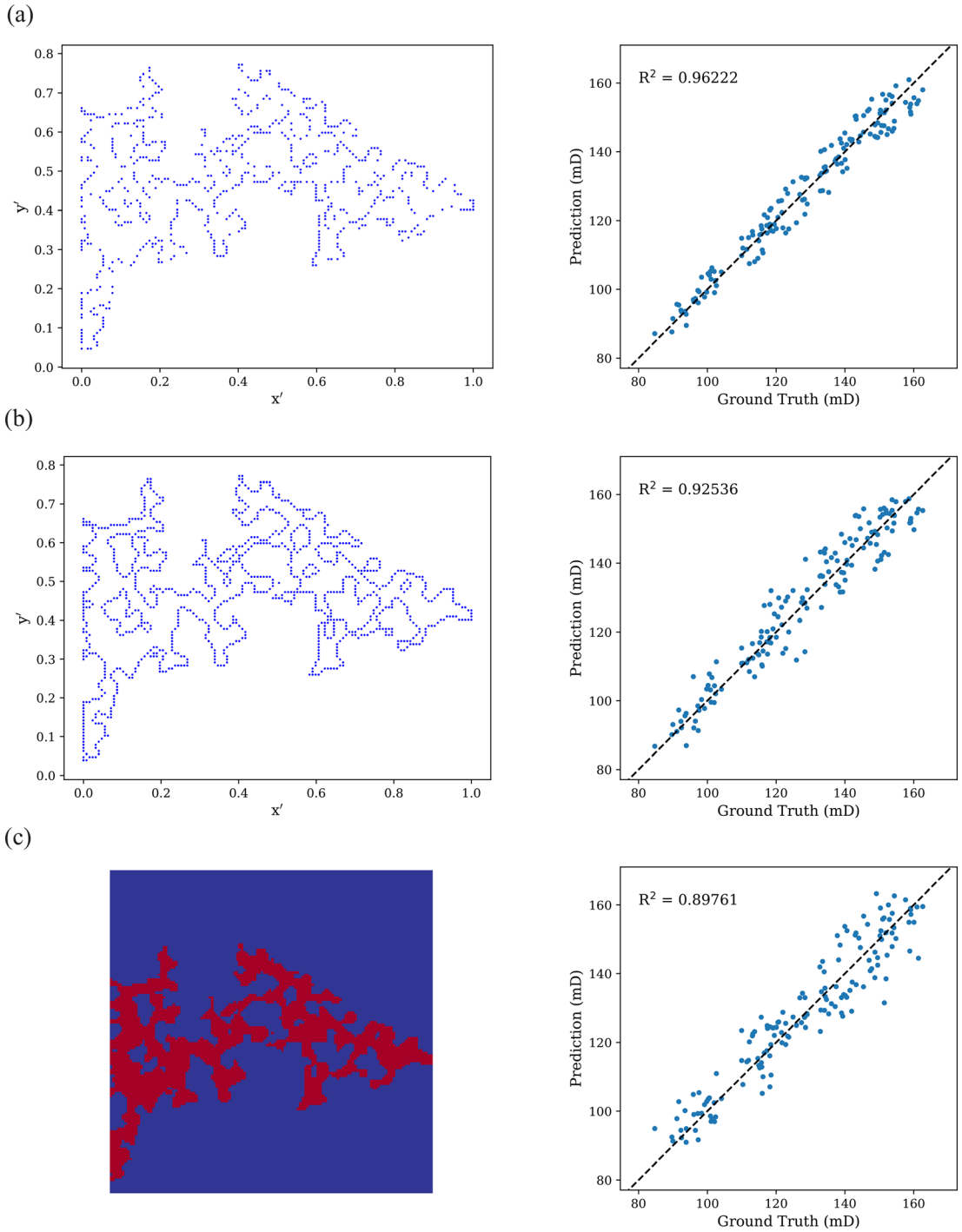

FIG. 6. Different input representations and their corresponding R^2 plots for (a) point-cloud neural network with $N = N_{min}$, (b) point-cloud neural network with $N = N_{max}$, and (c) 2D-CNN.

Our next machine learning experiment addresses the effect of input and feature transforms (see Fig. 4) on the accuracy of predicted permeability. From a computer vision point of view, the input and output transforms have two significant contributions to the shape classification problems. Here, we briefly describe these two contributions at a high level. One may refer to the original PointNet[32] article for a deeper discussion. First, these two transforms enhance the network performance to identify rotated objects. For instance, a rotated

TABLE I. Comparison between the performance of the point-cloud neural network and 2D-CNN for learning the permeability of two dimensional porous media.

	Point-cloud neural network	2D-CNN
R^2 score	0.962 22	0.897 61
Minimum relative error	0.002%	0.014%
Maximum relative error	5.365%	13.199%
Input vector size	2019 ($N_{\min} \times 3$)	16 384 (128×128 images)
Number of trainable parameters	2 415 763	3 459 601
Maximum possible batch size (increasing by a factor of 2)	Able to load all 2300 training data in one epoch	1024

cat still needs to be classified as a cat by PointNet.[32] Second, using these two transforms, the input point clouds are aligned to a canonical space, and it leads to a more efficient global feature extraction using the max-pooling operator (see Fig. 4). Concerning the application considered in this research article, the first contribution mentioned above is not useful. It is mainly because of the fact that we do not rotate our training data for data augmentation purposes. In other words, we are interested in permeability along the x-axis [see Eq. (3)] and by a rigid transformation of the digital rock, the corresponding permeability changes in this direction. However, there would be a hope that the second contribution of the transforms to the computer vision application improves our results as well. To answer this question, we remove the input and feature transform blocks from the point-cloud neural network (see Fig. 4) to investigate its usefulness. Figure 8(b) shows the R^2 plot as a consequence of this modification. As can be observed in Fig. 8(b), the R^2 score is reduced to 0.915 27. Hence, we conclude that the existence of these two transforms increases the network ability for

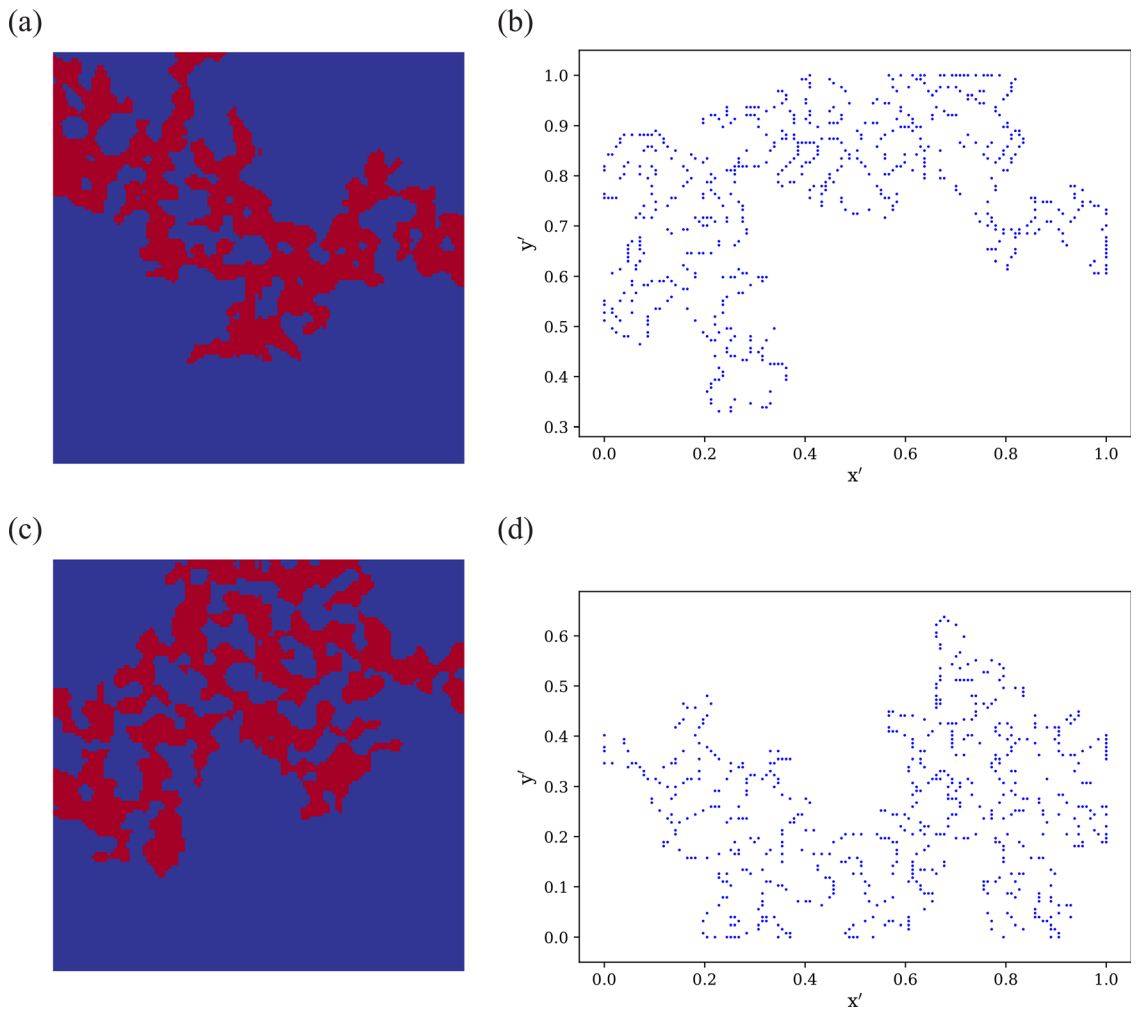

FIG. 7. Geometries with (a) minimum relative error for 2D-CNN, (b) minimum relative error for the point-cloud neural network, (c) maximum relative error for 2D-CNN, and (d) maximum relative error for the point-cloud neural network.

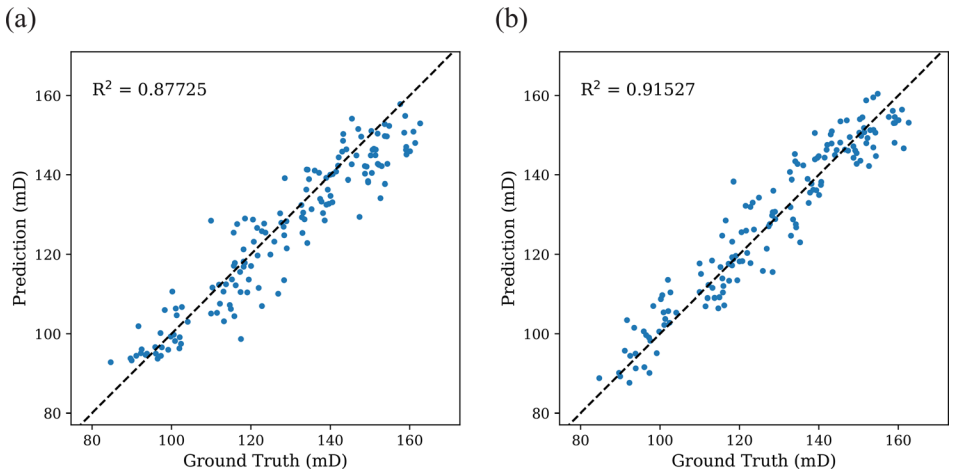

FIG. 8. R^2 scores obtained (a) with the ReLU activation function [see Eq. (7)] in the last layer of the point-cloud neural network and (b) without using the input and feature transforms in the neural network architecture (see Fig. 4).

permeability prediction although the network still performs with a relatively high level of accuracy in their absence.

The size of the latent global feature is a critical parameter of PointNet.[32] Qi *et al.*[32] discussed the effect of this parameter on the PointNet[32] performance for the object classification and segmentation task. Kashefi *et al.*[38] also investigated the influence of the global feature size for prediction of the velocity and pressure fields on unstructured grids (see Table II of Ref. 38). It is important to mention that the original PointNet[32] is designed with the global feature size of 1024 (see Fig. 2 of Ref. 32). Table II collects the R^2 scores of various global feature sizes for the point-cloud neural network. According to Table II, the highest performance is obtained for the size of 1024. Similar results are reported by Qi *et al.*[32] and Kashefi *et al.*[38] Note that by changing the global feature size, an adjustment in the size of the MLP right after the latent global feature is necessary to maintain the global feature as the main information bottleneck of the network structure.

We test the generalizability of the point-cloud neural network by prediction of the permeability of synthetic digital rock images (150 data) with the spatial correlation length of $l_c = 17$ and $l_c = 33$, while the network has only seen rock images with the correlation length of $l_c = 9$ in the training process. Note that from a computer science point of view, the generalizability of a neural network should be examined on unseen data from unseen categories. Thus, measuring the network performance on the test set cannot be interpreted as an indication of network generalizability because the test set contains unseen data but from seen categories. Table III demonstrates the outcome of this test. For $l_c = 17$, we only obtain a reasonable level of accuracy with the R^2 score of 0.673 02. However, for $l_c = 33$, the R^2 score takes a negative value with a maximum relative error of approximately 30%. The negative R^2 score indicates models that have worse predictions than a baseline based on just the mean value. A similar investigation is conducted for 2D-CNN, and a similar trend is experienced. However, a great reduction in the R^2 score is observed according to Table III. Compared to the point-cloud neural network performance, 2D-CNN experiences higher maximum relative errors and lower minimum relative errors based on the data tabulated in Table III. This observation demonstrates that 2D-CNN has relatively high bias on some cases (those predicted by low relative errors) and relatively high variance on other cases (those predicted by high relative errors). From this observation, we can also conclude that 2D-CNN is less generalizable in comparison with the point-cloud neural network. Similar results have been reported by Hong and Liu.[11] Although their network[11] trained on Coconino Sandstone achieved the R^2 score of 0.872 for the test set of permeability in the x-direction, they could only obtain the R^2 score of 0.6623 for predicting the same quantity for Bentheim Sandstone.

The next topic to discuss is the speedup factors achieved by our deep learning configuration. The point-cloud neural network estimates the permeability of the test set (150 data) in approximately 6 s on the GPU machine available in our computational resources. Computing the permeability of these 150 data using the LBM code written in C++ programing language takes on average 1350 s (approximately 23 min) on a single Intel(R) Core processor with the clock rate of 2.30 GHz. Consequently, the averaged achieved speedup factor is equal to 225 compared to the numerical simulation. Note that the factors

TABLE II. R^2 score as well as minimum and maximum relative errors for two dimensional porous media for different sizes of the global feature with the choice of $N = N_{\min}$; the FC size shows the size of different layers of the fully connected layer right after the global feature in the network (see Fig. 4).

Global feature size	128	256	512	1024	2048
FC size	(128, 128, 1)	(256, 128, 1)	(512, 256, 1)	(512, 256, 1)	(512, 256, 1)
R^2 score	0.897 99	0.916 51	0.915 41	0.962 22	0.918 45
Minimum relative error	0.030%	0.032%	0.004%	0.002%	0.038%
Maximum relative error	19.688%	13.276%	12.294%	5.365%	13.592%

TABLE III. Comparison between the generalizability of the point-cloud neural network and 2D-CNN, where they have never seen samples of porous media with spatial correlation lengths of $l_c = 17$ and $l_c = 33$ during the training procedure.

	Point-cloud neural network		2D-CNN	
	$l_c = 17$	$l_c = 33$	$l_c = 17$	$l_c = 33$
R^2 score	0.67302	−0.917 50	0.358 59	−1.231 18
Minimum relative error	0.439%	0.393%	0.048%	0.062%
Maximum relative error	13.178%	29.164%	66.971%	86.507%

reported here are not absolute and strongly depend on the efficiency of the LBM solver and the power of GPU and CPU (central processing unit) used. It should also be noted that the LBM solver is an in-house research code running on CPU alone. Modern commercial LBM codes taking advantage of GPUs are expected to be much faster than the code used in this work.

B. Three dimensional porous media

We construct the three dimensional point clouds (see Fig. 3) similar to the procedure explained in Sec. III A with a choice of $N_{\min} = 4003$. Figure 9 compares the R^2 score obtained by the point-cloud neural network with that achieved by 3D-CNN. Accordingly, the proposed deep learning technology gains 1.565% higher accuracy based on the metric of the coefficient of determination (0.991 51 vs 0.975 99). Table IV compares these two neural network types from other perspectives. As tabulated in Table IV, both the minimum and maximum relative errors of 3D-CNN are higher than those obtained by the neural network introduced in this study. In the point-cloud neural network, the minimum and maximum relative errors occur for porous media with permeability of 146.658 and 10.148 mD, respectively. The number of trainable parameters of 3D-CNN is slightly greater than the corresponding number in our deep learning

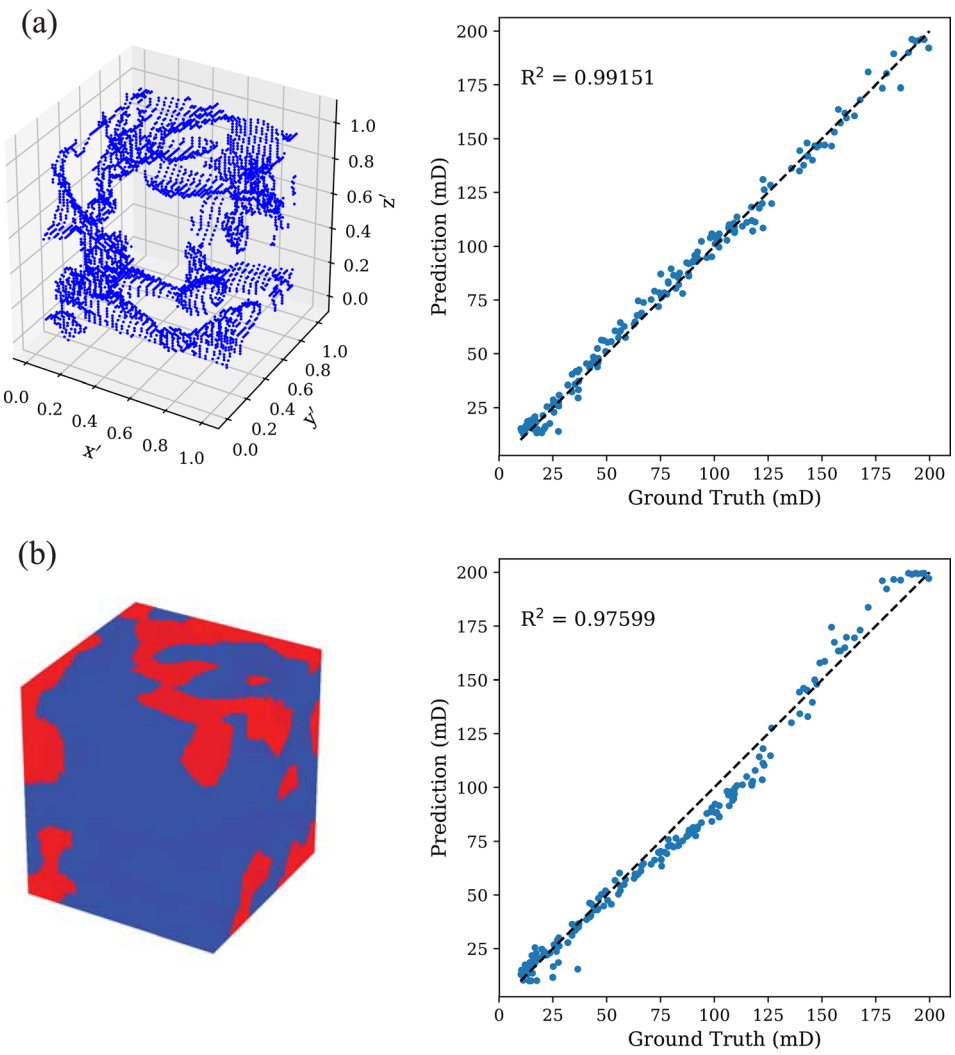

FIG. 9. Different input representations for three dimensional geometries and their corresponding R^2 plots for (a) point-cloud neural network and (b) 3D-CNN.

TABLE IV. Comparison between the performance of the point-cloud neural network and 3D-CNN for learning the permeability of three dimensional porous media.

	Point-cloud neural network	3D-CNN
R^2 score	0.991 51	0.975 99
Minimum relative error	0.030%	0.279%
Maximum relative error	50.487%	57.745%
Input vector size	12 009 ($N_{min} \times 3$)	262 144 ($64 \times 64 \times 64$ images)
Number of trainable parameters	2 415 763	2 585 169
Maximum possible batch size (increasing by a factor of 2)	2048	512

framework. Note that the number of trainable parameters of the point-cloud neural network is the same for both the two and three dimensional cases (see Tables I and IV) and in fact is independent of number of input points. This is simply because the MLP components in the first branch of the point-cloud neural network use shared weights as discussed in Sec. II C 1. Note that in contrast to the point-cloud neural network, the number of trainable parameters is a function of input size in CNN architectures.

Based on the information provided in Table IV, the input of the point-cloud neural network is a vector of size of 12009 ($N_{\min} \times 3$), while this parameter is approximately 22 times greater for 3D-CNN and is equal to 262 144 for $64 \times 64 \times 64$ images. This fact leads to the condition that maximum possible usable batch size of the point-cloud neural network is 2048, whereas it is equal to 512 for 3D-CNN. We further investigate the effect of batch size on the performance of the point-cloud neural network and 3D-CNN as reported, respectively, in Tables V and VI. As can be realized from Table V, the maximum R^2 score is obtained with the batch size of 1024 for the point-cloud neural network, while we are not able to run 3D-CNN with the batch size of 1024 and 2048 due to the lack of sufficient GPU memory. According to the data collected in Table VI, the batch size of 512 results in the maximum R^2 score of 3D-CNN, while the performance of 3D-CNN for the batch size of 1024 and 2048 is unknown to us. This is exactly the issue of 3D-CNN addressed in Sec. I. In fact, it would be completely possible that 3D-CNN with the batch size of 1024 or 2048 provides a higher R^2 score compared to when it is trained with the batch size of 512; however, memory restriction on GPU prevents us trying such experiments. Contrarily, such experiments are doable on the point-cloud neural network and eventually we obtain higher accuracy with the batch size of 1024 compared to 512. We observe how our new methodology overcomes the GPU memory limitation and ends in a higher level of prediction accuracy compared to the CNN methodology.

Concerning the achieved speedup factor, predicting the permeability of the test set (215 data) approximately takes 9 s by the point-cloud neural network, while the C++ LBM solver computes the permeability of this set in 38700 s (approximately 11 h). Hence, the point-cloud neural network, once trained, accelerates the permeability computations on average by a factor of 4300. Again as mentioned earlier, the LBM solver is an in-house research code running on CPU alone. Modern commercial LBM codes taking advantage of GPUs are expected to be much faster than the code used in this work.

To perform the generalizability of the point-cloud neural network for three dimensional porous media, we inspect the performance of the network on predicting the permeability of Berea sandstone samples (see, e.g., Ref. 2) as natural porous media, while the point-cloud neural network are solely trained on the synthetic data. Figure 10 exhibits the voxel and point cloud representations of one of these samples. The point-cloud neural network obtains R^2 score of 0.70437 with, respectively, the minimum and maximum relative errors of 2.735% and 35.578% over eight samples. We observe that only a reasonable accuracy level is gained because of two main reasons. First, although the permeability of the natural samples is in the range of training data, they have different spatial structure than data during the training procedure. Second, the number of points (N) in the clouds constructing the boundary of pore spaces in the natural samples vary between 4388 and 8696; this is while we set $N = 4003$ for the network and it ends in losing even further information about the correct structure of the natural samples and thus decreasing the R^2 score specifically for samples with large numbers of N (compared to 4003). It is concluded that to obtain higher R^2 scores, the network should be trained on similar natural samples or similar synthetic data from permeability, porosity, and spatial correlation length perspectives. Note that the goal of such an experiment is to test the generalizability of the network, meaning that we quantitatively investigate how a deviation from one of these three features negatively affects the accuracy of prediction; however, a decrease in the R^2 score would be expected in advance. As discussed in Sec. III A, we emphasize that the network must be asked to predict unseen data from unseen categories in a generalizability test.

At the end of this subsection, we address three points. First, our machine learning experiments shows that the contribution of input and feature transforms (see Fig. 4) to increasing the accuracy of predicting the permeability of three dimensional point clouds is insignificant (less than 0.1%). Thus, one may optionally remove these two transforms from the neural network to make it faster and lighter. Second, an important feature of the point-cloud neural network is its scalability. Depending on the number of points in the training point clouds, one may make the network smaller or larger. Alternatively,

TABLE V. Effect of batch size on the performance of the point-cloud neural network for predicting the permeability of three dimensional porous media.

Batch size	8	16	32	64	128	256	512	1024	2048
R^2 score	0.762 97	0.707 58	0.724 89	0.934 64	0.984 22	0.982 69	0.984 66	0.991 51	0.990 36
Minimum relative error (%)	0.013	0.032	0.060	0.224	0.019	0.058	0.066	0.030	0.031
Maximum relative error (%)	264.369	578.575	104.570	60.173	64.171	43.655	75.154	50.487	40.863

TABLE VI. Effect of batch size on the performance of 3D-CNN for predicting the permeability of three dimensional porous media; the cross symbol (×) indicates that training is impossible due to limitation of GPU memory.

Batch size	8	16	32	64	128	256	512	1024	2048
R^2 score	0.734 34	0.682 99	0.927 04	0.932 44	0.958 03	0.968 23	0.975 99	×	×
Minimum relative error (%)	0.096	0.300	0.079	0.320	0.007	0.097	0.279	×	×
Maximum relative error (%)	116.177	80.346	80.026	73.766	77.470	79.800	57.745	×	×

one may investigate the effect of the network size on its performance. For example, the size of network can be reduced by scaling its MLPs by a factor of 0.25, which leads to MPLs with sizes of (16, 16), (16, 32, 128), and (128, 64, 1), respectively, from left to right as shown in Fig. 4. In this case, the number of trainable parameters of the network without input and feature transforms decreases from 809601 to 51873. Third, training the point-cloud neural network on a data set containing porous media with a great range of spatial correlation lengths is challenging mainly because such set in practice leads to point-cloud subsets with a comparatively big difference between N_{max} and N_{min}. As discussed in Sec. III A, N is a hyper-parameter that needs to be tuned in the point-cloud neural network; however, when $N_{max} - N_{min}$ is very large, the training process becomes demanding. In fact, by selecting an N close to N_{min}, the corresponding geometries of point sets with $N \gg N_{min}$ are poorly represented. On the other hand, by choosing an N close to N_{max}, point sets with $N \ll N_{max}$ contain unreasonably a great number of repeated points, which eventually appear as redundant data to the point-cloud network and lead to decreasing the network performance. Moreover, our machine learning experiments show that a moderate N (e.g., arithmetic or geometric mean of N_{min} and N_{max}) is not an ideal option as well. Hence, one of our future study plans is to resolve these types of limitations from the proposed point-cloud deep learning framework. It is conjectured that such improvements could positively affect the generalizability of the network as well.

C. Potentials for fluid flow field predictions

In this article, our main focus is the prediction of permeability directly from the digital rock images. However, an alternative approach for the permeability prediction is to first predict the entire velocity fields in the pore spaces using a machine learning framework and then compute the permeability from the predicted velocity field. This approach has been so far taken by several researchers (see, e.g., Refs. 23 and 24). In this approach, a neural network is used as a replacement of conventional numerical solvers to provide an end-to-end mapping from the geometry of porous media to the fluid field of interest in the pore space. Due to high complexity in geometries of natural porous media, an efficient geometry representation in neural networks is necessary. CNNs as deep learning tools have been so far utilized for predicting the velocity fields in porous media. For instance, Santos *et al.*[24] used a three dimensional CNN but only to predict one component of the velocity field (parallel to the direction of the applied pressure gradient). Da Wang *et al.*[23] proposed a CNN based on U-Net[56] to predict the velocity fields in two and three dimensional porous media.

There are two common approaches to represent the geometry of porous media in the case of using CNNs. The first approach is to mask the pixels associated with the grains (see, e.g., Ref. 24). There are two major shortcomings with this approach. The first one is that for each numerical array (representing the porous media) a huge number of pixels have to be masked, specifically for porous media with low

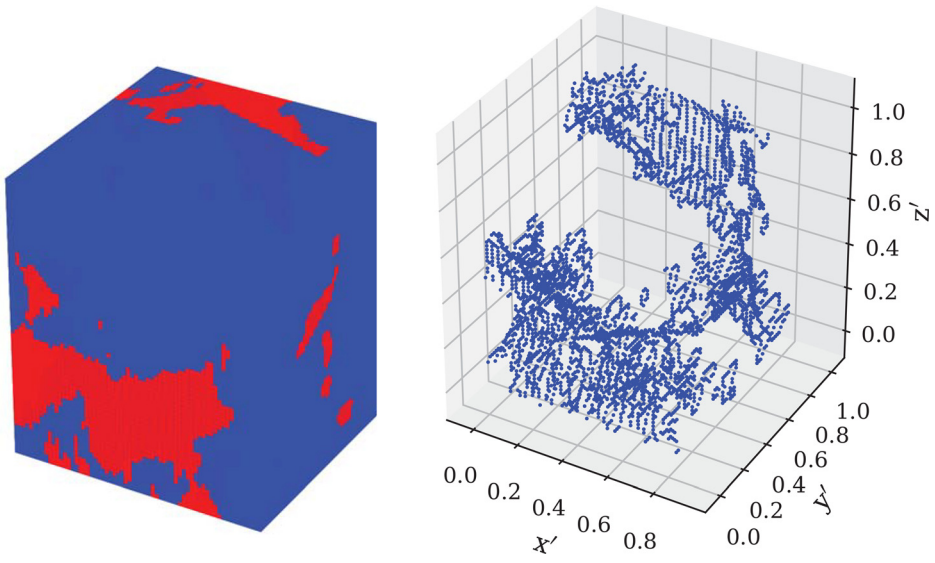

FIG. 10. Voxel and point cloud representations of one of Berea sandstone samples as a natural porous medium used for exploring the generalizability of our deep learning framework.

porosity. In fact, a considerable portion of computational capacity of CNNs is wasted by taking this strategy. The second one is that for digital rock images with high resolutions, a huge (and uncommon) memory size on Graphics Processing Units (GPUs) is necessary in order to load the entire domain of interest even with a batch size of one. This issue becomes highlighted specifically for prediction of flow fields in three dimensional porous media. For instance, this shortcoming has been reported by Santos *et al.*[24] when it was impossible for them to train their own CNN over the entire simulation domain. From a computer vision point of view, a possible solution would be breaking the entire domain into a few subdomains. Although it might resolve the memory issue, it would bring other concerns and constrains for CNNs training and the velocity field prediction. A full discussion on this issue can be found in Sec. 3.1 of Ref. 24. The minor issue of this approach is relevant to programing from a computer science point of view. An efficient function needs to be written to handle the pixel-masking procedure for a large number of training data, each one with a different pattern of fluid flow channels. In this approach, CNNs learn over training data that the fluid flow fields in pore spaces are a function of the number and pattern of the masked pixels. It is important to mention that this technique has been widely used in the area of fluid mechanics for deep learning flows over bluff bodies or airfoils (see, e.g., Ref. 57). The second approach is to label the input array using two different digit numbers: one representing the pore spaces and another representing the grain (solid) portions. Similarly, the corresponding pixels of pore spaces in the output array take the numerical value of the velocity field, whereas the corresponding pixels of grain spaces in the output array take the same number as input. In comparison, with the first approach, this scheme is easier to program and implement and also consumes less wall time over each epoch of training. On the other hand, the main difficulty with this approach is that CNNs not only have to learn the fluid flow fields in pore spaces, but also have to learn the geometry of grain spaces (see, e.g., Ref. 23). There are three major shortcomings with this approach. The first and second issues are similar to the first approach: wasting considerable computational resources of CNN frameworks and requiring an unreasonable memory size on GPUs even for batching one sample from data set. The third major shortcoming is that machine learning experiments showed that CNNs have troubles identifying the pore spaces vs the grain spaces, specifically when the geometry of flow cluster becomes highly complicated and the effective porosity of the rock decreases. For instance, CNNs proposed by Da Wang *et al.*[23] predicted the velocity fields (regardless the accuracy of its magnitude) in regions where the flow does not even exist (i.e., in grain spaces). Note that there are a few alternative methods to represent the pore spaces as the input of CNNs, such as using the Euclidean distance transform function (see, e.g., Refs. 23 and 24), instead of using a single number. Although these alternatives might increase the CNN performance, the fundamental issues addressed here remains unchanged. In summary, both of these two approaches suffer from modeling and involving the grain spaces in a CNN deep learning framework. To resolve these issues, we suggest only taking the pore space of a porous medium and representing it as a set of points that constructs a point cloud. Point clouds represent, respectively, the surface and volume of pore spaces of two and three dimensional porous media. Consequently, one may use the segmentation component of PointNet[32] to establish an end-to-end mapping between the spatial coordinates of each point of a cloud and the numerical values of the velocity vector at that point. It is conjectured that PointNet++[58] and Kpconv[33] would have higher performance in comparison with PointNet[32] because they pay more attention to local features of a given geometry.

Moreover, designing an efficient cost function in such problems is critical (see, e.g., Sec. 2.3.2 of Ref. 24). Cost functions so far used are mainly based on L^1 or L^2 norm error of the velocity fields.[23,24] To enhance the performance of such neural networks and accuracy of the velocity field prediction, we suggest two strategies. Both of these two strategies are based on adding information of the flow governing equations to neural network cost functions. The first strategy leads to a supervised deep learning approach, while the second one results in an unsupervised (or semi-supervised) methodology. The first approach is to add the residual of the continuity and Navier–Stokes equations [Eqs. (1) and (2)] to the cost function. This approach has been carried out in other research areas (see, e.g., Refs. 57 and 59). The second approach is to use the technology of the Physics Informed Neural Network (PINN).[60–62] In PINNs, the cost function is defined based on the governing equations of the problem of interest as well as the desired initial and boundary conditions, while there is no need of labeled data for training. One may refer to Refs. 60–62 for a deeper discussion on PINNs. Note that there is no mechanism in the current version of PINNs to capture the variations in the geometry of problem domains. In other words, the parameters (e.g., weights and bias) of PINNs are not a function of the geometry of physical domains. Thus, the combination of PointNet[32] (or other point-cloud based neural networks) with PINNs has the potential to resolve this issue.

IV. SUMMARY

In this study, we introduced a novel point-cloud based deep learning configuration for permeability predictions of digital porous media. We designed the architecture of this configuration according to the classification branch of PointNet.[32] Taking the advantages of the point-cloud based deep learning methodology, limitations on GPU memory requirements were relaxed and selecting higher batch sizes compared to CNNs became possible. It was mainly due to dramatically diminishing the size of network inputs by only taking the boundary of solid matrix and pore spaces in a porous medium via point cloud representations, rather than taking its whole volume via voxel representations. Freedom in the choice of batch size provided the chance of exploring a relatively wide range of batch sizes to obtain the highest possible accuracy of the permeability prediction. We concentrated on synthetic digital rocks as test cases. According to the metric of coefficient of determination, our deep learning technique achieved excellent accuracy for the predicted permeability of both two and three dimensional porous media. Compared to a numerical LBM solver, the point-cloud neural network predicted the permeability of test set a few thousand times faster. Finally, we discussed the generalizability of the point-cloud neural network by examining it over two unseen categories: real-world samples and synthetic samples but with unseen spatial correlation lengths.

ACKNOWLEDGMENTS

We acknowledge the sponsors of the Stanford Center for Earth Resources Forecasting (SCERF) and support from Professor Steve Graham, the Dean of the Stanford School of Earth, Energy and Environmental Sciences. The work was funded by Shell-Stanford

collaborative project on Digital Rock Physics and the Army Research Office Contract No. W911NF1810008. Some of the computing for this project was performed on the Sherlock cluster. We would like to thank Stanford University and the Stanford Research Computing Center for providing computational resources and support that contributed to these research results. Additionally, we wish to thank the reviewers for their insightful comments.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]H. Andrä, N. Combaret, J. Dvorkin, E. Glatt, J. Han, M. Kabel, Y. Keehm, F. Krzikalla, M. Lee, C. Madonna *et al.*, "Digital rock physics benchmarks—Part I: Imaging and segmentation," Comput. Geosci. **50**, 25–32 (2013).
[2]H. Andrä, N. Combaret, J. Dvorkin, E. Glatt, J. Han, M. Kabel, Y. Keehm, F. Krzikalla, M. Lee, C. Madonna *et al.*, "Digital rock physics benchmarks—Part II: Computing effective properties," Comput. Geosci. **50**, 33–43 (2013).
[3]M. Gruber, C. Johnson, C. Tang, M. Jensen, L. Yde, and C. Hélix-Nielsen, "Computational fluid dynamics simulations of flow and concentration polarization in forward osmosis membrane systems," J. Membr. Sci. **379**, 488–495 (2011).
[4]M. J. Blunt, B. Bijeljic, H. Dong, O. Gharbi, S. Iglauer, P. Mostaghimi, A. Paluszny, and C. Pentland, "Pore-scale imaging and modelling," Adv. Water Resour. **51**, 197–216 (2013).
[5]K. Khanafer, K. Cook, and A. Marafie, "The role of porous media in modeling fluid flow within hollow fiber membranes of the total artificial lung," J. Porous Media **15**, 113 (2012).
[6]S. Karimpouli and P. Tahmasebi, "Segmentation of digital rock images using deep convolutional autoencoder networks," Comput. Geosci. **126**, 142–150 (2019).
[7]Y. D. Wang, M. Shabaninejad, R. T. Armstrong, and P. Mostaghimi, "Deep neural networks for improving physical accuracy of 2D and 3D multi-mineral segmentation of rock micro-CT images," Appl. Soft Comput. **104**, 107185 (2021).
[8]Y. Niu, P. Mostaghimi, M. Shabaninejad, P. Swietojanski, and R. T. Armstrong, "Digital rock segmentation for petrophysical analysis with reduced user bias using convolutional neural networks," Water Resour. Res. **56**, e2019WR026597, https://doi.org/10.1029/2019WR026597 (2020).
[9]K. M. Graczyk and M. Matyka, "Predicting porosity, permeability, and tortuosity of porous media from images by deep learning," Sci. Rep. **10**, 21488 (2020).
[10]A. Bhatt, "Reservoir properties from well logs using neural networks," Ph.D. thesis (Norwegian University of Science and Technology, 2002).
[11]J. Hong and J. Liu, "Rapid estimation of permeability from digital rock using 3D convolutional neural network," Comput. Geosci. **24**, 1523–1539 (2020).
[12]J. Wu, X. Yin, and H. Xiao, "Seeing permeability from images: Fast prediction with convolutional neural networks," Sci. Bull. **63**, 1215–1222 (2018).
[13]M. Röding, Z. Ma, and S. Torquato, "Predicting permeability via statistical learning on higher-order microstructural information," Sci. Rep. **10**, 15239 (2020).
[14]M. Tembely, A. M. AlSumaiti, and W. Alameri, "A deep learning perspective on predicting permeability in porous media from network modeling to direct simulation," Comput. Geosci. **24**, 1541–1556 (2020).
[15]J. Tian, C. Qi, Y. Sun, Z. M. Yaseen, and B. T. Pham, "Permeability prediction of porous media using a combination of computational fluid dynamics and hybrid machine learning methods," Eng. Comput. **2020**, 1–17.
[16]A. Zolotukhin and A. Gayubov, "Machine learning in reservoir permeability prediction and modelling of fluid flow in porous media," IOP Conf. Ser.: Mater. Sci. Eng. **700**, 012023 (2019).
[17]N. Alqahtani, F. Alzubaidi, R. T. Armstrong, P. Swietojanski, and P. Mostaghimi, "Machine learning for predicting properties of porous media from 2D x-ray images," J. Pet. Sci. Eng. **184**, 106514 (2020).
[18]A. Rabbani, M. Babaei, R. Shams, Y. D. Wang, and T. Chung, "Deepore: A deep learning workflow for rapid and comprehensive characterization of porous materials," Adv. Water Resour. **146**, 103787 (2020).
[19]F. Bordignon, L. Figueiredo, R. Exterkoetter, B. B. Rodrigues, and M. Correia, "Deep learning for grain size and porosity distributions estimation on micro-CT images," in Proceedings of the 16th International Congress of the Brazilian Geophysical Society & Expogef (2019).
[20]V. Hébert, T. Porcher, V. Planes, M. Léger, A. Alperovich, B. Goldluecke, O. Rodriguez, and S. Youssef, "Digital core repository coupled with machine learning as a tool to classify and assess petrophysical rock properties," in E3S Web Conf. **146**, 01003 (2020).
[21]H. Wu, W.-Z. Fang, Q. Kang, W.-Q. Tao, and R. Qiao, "Predicting effective diffusivity of porous media from images by deep learning," Sci. Rep. **9**, 20387 (2019).
[22]S. Karimpouli and P. Tahmasebi, "Image-based velocity estimation of rock using convolutional neural networks," Neural Networks **111**, 89–97 (2019).
[23]Y. Da Wang, T. Chung, R. T. Armstrong, and P. Mostaghimi, "ML-LBM: Machine learning aided flow simulation in porous media," arXiv:2004.11675 (2020).
[24]J. E. Santos, D. Xu, H. Jo, C. J. Landry, M. Prodanović, and M. J. Pyrcz, "PoreFlow-Net: A 3D convolutional neural network to predict fluid flow through porous media," Adv. Water Resour. **138**, 103539 (2020).
[25]S. Li, X. Shen, Y. Dou, S. Ni, J. Xu, K. Yang, Q. Wang, and X. Niu, "A novel memory-scheduling strategy for large convolutional neural network on memory-limited devices," Comput. Intell. Neurosci. **2019**, 4328653.
[26]I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).
[27]N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," arXiv:1609.04836 (2016).
[28]I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," ICT Express **6**, 312–315 (2020).
[29]D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," arXiv:1804.07612 (2018).
[30]Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade* (Springer, 2012), pp. 437–478.
[31]Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," ACM Trans. Graph. **38**, 146 (2019).
[32]C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017), pp. 652–660.
[33]H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (IEEE, 2019), pp. 6411–6420.
[34]C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," in *proceedings of the IEEE/CVF International Conference on Computer Vision* (IEEE, 2019), pp. 9277–9286.
[35]C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointNets for 3D object detection from RGB-D Data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2018), pp. 918–927.
[36]X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2019), pp. 529–537.
[37]D. Rempe, T. Birdal, Y. Zhao, Z. Gojcic, S. Sridhar, and L. J. Guibas, "CaSPR: Learning canonical spatiotemporal point cloud representations," arXiv:2008.02792 (2020).
[38]A. Kashefi, D. Rempe, and L. J. Guibas, "A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries," Phys. Fluids **33**, 027104 (2021).
[39]D. Rempe, S. Sridhar, H. Wang, and L. J. Guibas, "Learning generalizable final-state dynamics of 3D rigid objects," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, Long Beach, CA, USA, 16–20 June, 2019* (Computer Vision Foundation/IEEE, 2019), pp. 17–20.
[40]R. S. DeFever, C. Targonski, S. W. Hall, M. C. Smith, and S. Sarupria, "A generalized deep learning approach for local structure identification in molecular simulations," Chem. Sci. **10**, 7503–7515 (2019).

[41]Y. Keehm, T. Mukerji, and A. Nur, "Permeability prediction from thin sections: 3D reconstruction and lattice-Boltzmann flow simulation," Geophys. Res. Lett. **31**, L04606, https://doi.org/10.1029/2003GL018761 (2004).
[42]H. Darcy, *Les Fontaines Publiques de la Ville de Dijon: Exposition et Application* (Victor Dalmont, 1856).
[43]A. Eshghinejadfard, L. Daróczy, G. Janiga, and D. Thévenin, "Calculation of the permeability in porous media using the lattice Boltzmann method," Int. J. Heat Fluid Flow **62**, 93–103 (2016).
[44]C. Lantuéjoul, *Geostatistical Simulation: Models and Algorithms* (Springer Science & Business Media, 2013).
[45]W. Xu, A. Journel *et al.*, "GTSIM: Gaussian truncated simulations reservoir units in a W. Texas carbonate field," SPE Paper No. 27412, 1993.
[46]S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning* (PMLR, 2015), pp. 448–456.
[47]Y. Yu, Z. Gong, P. Zhong, and J. Shan, "Unsupervised representation learning with deep convolutional neural network for remote sensing images," in *International Conference on Image and Graphics* (Springer, 2017), pp. 97–108.
[48]A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)* (IEEE, 2020), pp. 1–5.
[49]A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artif. Intell. Rev. **53**, 5455–5516 (2020).
[50]J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *IEEE International Conference on Signal and Image Processing Applications (ICSIPA)* (IEEE, 2011), pp. 342–347.
[51]P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding convolution for semantic segmentation," in *IEEE Winter Conference on Applications of Computer Vision (WACV)* (IEEE, 2018), pp. 1451–1460.
[52]J. Yamanaka, S. Kuwashima, and T. Kurita, "Fast and accurate image super resolution by deep CNN with skip connection and network in network," in *International Conference on Neural Information Processing* (Springer, 2017), pp. 217–225.
[53]V. Sekar, Q. Jiang, C. Shu, and B. C. Khoo, "Fast flow field prediction over airfoils using deep learning approach," Phys. Fluids **31**, 057103 (2019).
[54]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[55]P. Jain and P. Kar, "Non-convex optimization for machine learning," arXiv:1712.07897 (2017).
[56]O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2015), pp. 234–241.
[57]S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik, "Prediction of aerodynamic flow fields using convolutional neural networks," Comput. Mech. **64**, 525–545 (2019).
[58]C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December, 2017*, edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017), pp. 5099–5108.
[59]A. Subramaniam, M. L. Wong, R. D. Borker, S. Nimmagadda, and S. K. Lele, "Turbulence enrichment using physics-informed generative adversarial networks," arXiv:2003.01907 (2020).
[60]A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," Comput. Methods Appl. Mech. Eng. **365**, 113028 (2020).
[61]Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020).
[62]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).