Transport in Porous Media (2021) 140:241–272
https://doi.org/10.1007/s11242-021-01617-y

Computationally Efficient Multiscale Neural Networks Applied to Fluid Flow in Complex 3D Porous Media

Javier E. Santos[1] · **Ying Yin**[2] · **Honggeun Jo**[1] · **Wen Pan**[1] · **Qinjun Kang**[3] · **Hari S. Viswanathan**[3] · **Maša Prodanović**[1] · **Michael J. Pyrcz**[1] · **Nicholas Lubbers**[3]

Received: 10 January 2021 / Accepted: 7 May 2021 / Published online: 29 May 2021

Abstract

The permeability of complex porous materials is of interest to many engineering disciplines. This quantity can be obtained via direct flow simulation, which provides the most accurate results, but is very computationally expensive. In particular, the simulation convergence time scales poorly as the simulation domains become less porous or more heterogeneous. Semi-analytical models that rely on averaged structural properties (i.e., porosity and tortuosity) have been proposed, but these features only partly summarize the domain, resulting in limited applicability. On the other hand, data-driven machine learning approaches have shown great promise for building more general models by virtue of accounting for the spatial arrangement of the domains' solid boundaries. However, prior approaches building on the convolutional neural network (ConvNet) literature concerning 2D image recognition problems do not scale well to the large 3D domains required to obtain a representative elementary volume (REV). As such, most prior work focused on homogeneous samples, where a small REV entails that the global nature of fluid flow could be mostly neglected, and accordingly, the memory bottleneck of addressing 3D domains with ConvNets was side-stepped. Therefore, important geometries such as fractures and vuggy domains could not be modeled properly. In this work, we address this limitation with a general multiscale deep learning model that is able to learn from porous media simulation data. By using a coupled set of neural networks that view the domain on different scales, we enable the evaluation of large ($> 512^3$) images in approximately one second on a single graphics processing unit. This model architecture opens up the possibility of modeling domain sizes that would not be feasible using traditional direct simulation tools on a desktop computer. We validate our method with a laminar fluid flow case using vuggy samples and fractures. As a result of viewing the entire domain at once, our model is able to perform accurate prediction on domains exhibiting a large degree of heterogeneity. We expect the methodology to be applicable to many other transport problems where complex geometries play a central role.

Keywords Convolutional neural networks · Multiscale · Machine learning · Permeability · Lattice–Boltzmann · Representative elementary volume (REV)

✉ Javier E. Santos
jesantos@utexas.edu

Extended author information available on the last page of the article

1 Introduction

In the last few decades, micro-tomographic imaging in conjunction with direct numerical simulations (digital rock technologies) have been developed extensively to act as a complementary tool for laboratory measurements of porous materials (Schepp et al. 2020). Many of these breakthroughs are partly thanks to advances in data storing and sharing (Prodanovic et al.), wider availability of imaging facilities (Cnudde and Boone 2013), and better technologies (hardware and software) to visualize fine-scale features of porous media (Wildenschild and Sheppard 2013). Nevertheless, characterization based on standalone images does not provide enough insight on how the small-scale structures affect the macroscopic behavior for a given phenomenon (i.e., fluid flow). A more robust way of understanding these (and potentially being able to upscale them) is through simulating the underlying physics of fluid flow.

The increase in speed and availability of computational resources (graphics processing units, supercomputer clusters, and cloud computing) has made it possible to develop direct simulation methods to obtain petrophysical properties based on 3D images (Pan et al. 2004; Tartakovsky and Meakin 2005; White et al. 2006; Jenny et al. 2003). However, solving these problems in time frames that could allow their industrial applicability requires thousands of computing cores. Furthermore, the most insight could be gained in repeated simulation with dynamically changing conditions. For example, to assess the influence of diagenetic processes, such as cementation and compaction, surface properties like roughness, or tuning the segmentation of a sample to match experimental measurements. These scenarios entail solving a forward physics model several times (in similar domains), which is prohibitively expensive in many cases. A model that could give fast and accurate approximations of a given petrophysical property is of great interest.

A particular physical framework of interest in digital rocks physics is to describe how a fluid flows through a given material driven by a pressure difference. This is relevant to characterize how easy it is for a fluid to travel through a specific sample, and it can also reveal the presence of preferential fluid paths and potential bottlenecks for flow. By understanding how a fluid behaves in a representative sample, it is possible to use this data to inform larger scale processes about the effect of the microstructure. For example, a strong understanding of representative flow can provide macroscopic simulations with constitutive relations. The simplest and most important way to summarize the microstructural effects on flow is with the permeability, which is a volume-average property derived from the fluid velocity which describes how well a fluid can advance through its connected void-space. Knowing the permeability is of interest for not only for petroleum engineering (Sun et al. 2017), but for carbon capture/sequestration (Bond et al. 2017), and aquifer exploitation (Cunningham and Sukop 2011), but also in geothermal engineering (Molina et al. 2020), membrane design, and fuel cell applications (Holley and Faghri 2006).

Despite the fact that there are many published analytical solutions and computational algorithms to obtain the permeability in a faster manner, they do not work well in the presence of strong heterogeneities associated with important geometries such as fractures (Tokan-Lawal et al. 2015). This is partly due to the fact that most of these proposed solutions are computed based on averaged properties of the solid structure, such as the porosity and the tortuosity of the sample (Carman 1939, 1997; Kozeny 1927; Bear 1972). The main issue is that samples with very similar average structural values could have widely different volumetric flux behaviors, for example, when fractures or vugs are present. For instance, a certain porous structure could have permeability values spanning three orders of magnitude depending whether the domain is not fractured, or if it hosts a fracture parallel or perpendicular to the direction of flow. While these situations significantly affect permeability, the porosity remains relatively unchanged; there is no known route for characterizing the relationship between flow and complex microstructure in terms of small number of variables. While there are theoretical approaches based on correlation functions (Torquato 2020), they are mostly applied to statistically homogenous microstructures, and it remains unclear if such an approach can apply to the complex multiscale porous media found in many natural samples.

To obtain a measure of the permeability of a sample taking into account the 3D microstructure, a fluid flow simulation can be carried out with a wide variety of iterative numerical methods that approximate the solution of the Navier–Stokes equation (Saxena et al. 2017). One of the most prominent is the Lattice–Boltzmann method (LBM). Although these simulations are performed at a much smaller scale relatively to a natural reservoir, they provide the critical parameters to enable the upscaling of hard-data (such as cores coming from wells or outcrops) into field scale simulators. Although it would be desirable to simulate bigger computational volumes that contain more information about the reservoir of interest (since imaging technology can provide volumes that are 2000^3 voxels or larger), it is computationally expensive, making it very difficult to perform routinely or repeatedly.

A representative elementary volume (REV) has to be ensured to reliably utilize these properties in large scale (field) simulations (and thus upscale). An REV is defined as the size of a window where measurements are scale-independent, and that accurately represents the system (Bachmat and Bear 1987). Notwithstanding, having an REV, i.e., porosity (which is easily determined from a segmented image), does not guarantee that this window size would have a representative response in a flow property like permeability. As shown in Costanza–Robinson et al. (2011), for fairly homogeneous samples, the side length of the window to obtain an REV in a dynamic property is at least five times what is its for the structure (porosity). This is one of the reasons why porosity alone is a poor estimate for permeability: Even when the pore structures are similar, the flow patterns that they host could be very different due to the global nature of the steady-state solution. In the context of single fractures, it is still unclear if an REV exists (Santos et al. 2018a; Guiltinan et al. 2020b, a). This puts into prominence the need for more advanced methods that can provide accurate solutions on large samples that take in account all the complexities of the domain.

In the last decade, convolutional neural networks (ConvNets) (Fukushima 1980) have become a prominent tool in the field of image analysis. These have taken over traditional tools for computer vision tasks such as image recognition and semantic segmentation, as result of being easily trained to create spatially-aware relationships between inputs and outputs. This is accomplished with learnable *kernels* which can be identified with small windows that have the same dimensionality as the input data (i.e., 2D for images, 3D for volumes). They have been successfully applied in many tasks regarding earth science disciplines (Alakeely and Horne 2020; Jo et al. 2020; Pan et al. 2020), and particularly in the field of digital rocks (Guiltinan et al. 2020a; Mosser et al. 2017a, b; Chung et al. 2020; Bihani et al. 2021; Santos et al. 2018b). These data-driven models have also been useful for solving flow (Santos et al. 2020b; Wang et al. 2021a, b; Alqahtani et al. 2021), successfully modeling the relationship between 3D microstructure and flow response much more accurately than empirical formulas that depend only on averaged properties.

However, ConvNets are expensive to scale to 3D volumes. This is due to the fact that these structures are memory intensive, so traditional networks used for computer vision tasks (i.e., *UNet* (Jha et al.) or the *ResNet* (He et al. 2016)) limit the input sizes to be around 100^3 (Santos et al. 2020b; Wang et al. 2021b). As shown in Santos et al. (2020b), one could subsample the domain into smaller pieces to use these architectures, where the subsample does not need to be an REV but it has to be accompanied by features that inform the model about the original location of this subdomain (i.e., tortuosity, pore-space connectivity, distance to the non-slip boundaries). This method provides accurate results, nevertheless, predictions stop being reliable in domains with large heterogeneities, such as a fracture or vugs. Alternatively, one can downsample the image into a more tractable size (Alqahtani et al. 2021), sacrificing detail. Wang et al. (2021a) states that one major point of focus for deep learning of porous media is the computational cost of working with 3D arrays, and the development of architectures that are better suited for porous materials instead of the usual architectures proposed for 2D classification/segmentation of standardized datasets (Deng et al. 2009). Here, we provide a strong approach to directly address these outstanding issues.

A multiscale approach that is able to capture large and small scale aspects of the microstructure simultaneously is an attractive proposal to overcome this limitation. Multiscale approaches have precedent in the ConvNet literature. Super-resolution ConvNets (Shi et al. 2016; Huang et al. 2017; Yang et al. 2018; Yamanaka et al. 2017) address images at two different scales, although the problem is not one of regression, but of interpolation; fine-scale input is not available to the model. Karra et al (2017) provide an explicitly multiscale training pipeline, which progressive grows a generative network to build a high-resolution model of image datasets. This is accomplished by starting with coarser, lower-resolution networks and gradually adding details using networks that work with finer scales. Perhaps the most relevant to our work is SinGAN (Shaham et al. 2019), another generative model that uses a linked set of networks to describe images at different scales; finer scales build upon the models for coarser scales.

We invoke similar principles to build a *MultiScale Network for hierarchical regression* (MS-Net), which performs regression based on a hierarchical principle: coarse inputs provide broad information about the data, and progressively finer-scale inputs can be used to refine this information. A schematic of the workflow is shown in Fig. 1. In this paper, we use MS-Net to learn relationships between pore structure and flow fields of steady-state solutions from LBM. Our model starts by analyzing a coarsened version of a porous domain (where the main heterogeneities affecting flow are present), and then proceeds to make a partial prediction of the velocity field. This is then passed subsequently to finer-scale networks to refine this coarse prediction until the entire flow field is recovered. This paradigm exhibits strong advantages over other ConvNet approaches such as Poreflow-Net (Santos et al. 2020b) with regard to the ability to learn on heterogeneous domains and in terms of the computational expense of the model. While applied here to fluid flow, we believe this hierarchical regression paradigm could be applied to many disciplines dealing with 3D volumes, not limited to the problems studied here.

The rest of this manuscript is organized as follows. In Sect. 2, we describe our methods, and in Sect. 3, we describe the data we have applied our methods to. In Sect. 4, we describe the results of training using two different data sources. We show the results on test data comprised of a variety of samples presenting a wide range of heterogeneities at different scales. In Sect. 5, we provide discussion, including comments on the memory-efficiency of the approach, and we conclude in Sect. 6.

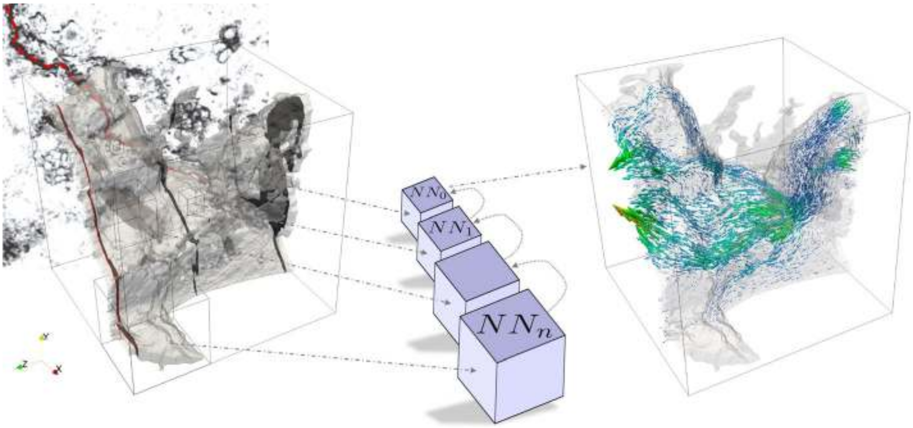


Fig. 1 Overview of our multiscale network approach. Starting from a 3D μCT image of a fractured carbonate (a gray-scale cross-section of the domain is shown in the back, where unconnected microporosity can be observed, and the main fracture of the domain is shown in dashed red lines), we predict the single-phase flow field of the image by combining predictions made over multiple resolutions of the binary input (shown to the left, where only the surface between the pore-space and the solid is shown in gray). Each neural network (depicted as purple boxes) scans the image with different window sizes that increase exponentially (depicted with gray outline cubes in the domain). The different sizes of the neural networks depict higher model capacity. The predictions of the networks are all linked together (via the arrow connections) to provide an approximation of the Navier–Stokes solution (shown in the right, where the higher velocity values get brighter colors). All these elements are explained in detail in Sects. 2 and Fig. 3

2 Multiscale Neural Network Description

Our end goal is to build a model which can estimate flow characteristics based on the microstructure found in a variety of complex samples, with constant fluid properties and time independent driving force. We aim to capture the steady-state fluid flow and associated statistics thereof, such as the permeability, but emphasize that other field quantities at steady-state could be addressed with the same framework. The main requirement for our approach is to have a domain (constituting a 3D binary array), and a matching response (simulation or experiment) across that domain. Additional information would be needed to capture more complex situations such as time-dependent (unsteady) flow. We choose to model this relationship with system of deep neural networks.

The task of learning the physics of transport in porous media requires a model that can learn complex relationships (like the one between structure and fluid velocity), and that has capacity to generalize many possible domains (outside of the ones used for training) for its broad applicability. One approach is to obtain massive amounts of data. This is expensive for multiscale porous media, which need to be modeled at very high resolutions. Moreover, model training time scales with the evaluation speed of the model. The common deep learning approach to building complex, well-generalizing models is to use networks that are (1) very deep, having a large number of layers or (2) very wide, having a large number of neurons at each layer. Although these strategies typically results in higher accuracy, they always result in a larger computation required to train and use the model. The memory cost is proportional not only to the width and depth of the network, but also with the volume of the sample that needs to be analyzed.

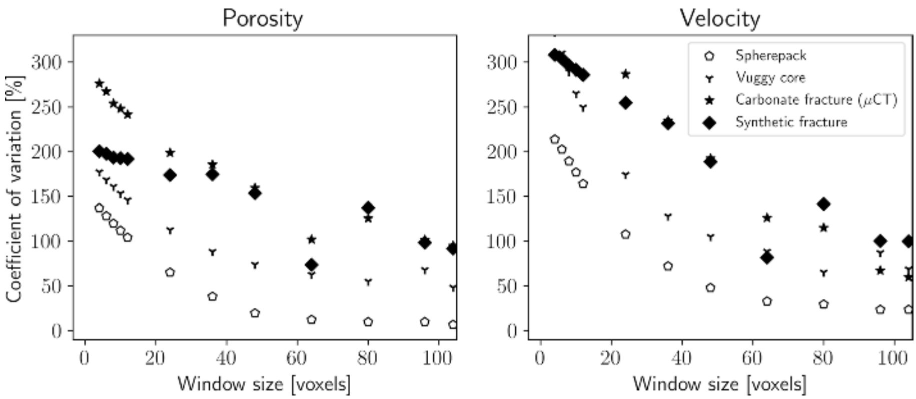


Fig. 2 Coefficient of variation (ratio between standard deviation and the mean) of the porosity and fluid flow field for domains subsampled using increasingly larger window sizes. We show four examples: a sphere pack, a vuggy core, an imaged carbonate fracture (from Fig. 1) and a synthetic fracture (from Sect. 4.2). For samples presenting large heterogeneities (like the fractures), very large windows (>100 voxels) are necessary to capture representative porosity and flow patterns

In practice, this has limited the volume with which 3D data can be evaluated on a single GPU to sizes on the order of 80^3 (Santos et al. 2020b).

One approach to address this limitation is to break large domains into sub-samples, and augment the feature set so that it contains hand-crafted information pertaining to the relative location of each sub-sample (Santos et al. 2020b). This can add information about the local and global boundaries surrounding the subsample. However, a clear limitation of this approach is its applicability for domains containing large-scale heterogeneities. Figure 2 shows the variation of properties as a function of window size for various data considered in this paper, and it is clear that in some cases the REV may be much larger than 80 voxels. If this data is split into sub-domains, the large-scale information about features is limited to the hand-crafted information fed to the model, and the power of machine learning to analyze the correlations in the 3D input space is limited to window sizes that are orders of magnitude smaller than the REV.

Instead of increasing the model size, one can design topologies for a network (*architectures*) which aim to capture aspects of human cognition or of domain knowledge. For example, self-attention units (Vaswani et al. 2017; Jaegle et al. 2021; Ho et al. 2019; Dosovitskiy et al. 2020) capture a notion of dynamic relative importance among features.

We propose an architecture designed to overcome the hurdles of training convolutional neural networks to small 3D volumes, the multiscale network (*MS-Net*), and show that MS-Net is a suitable system to model physics in complex porous materials. The MS-Net is a coupled system of convolutional neural networks that allows training to entire samples to understand the relationship between pore-structure and single-phase flow physics, including large-scale effects. This makes it possible to provide accurate flow field estimations in large domains, including large-scale heterogeneities, without complex feature engineering.

In the following sections, we first provide an overview of how convolutional neural networks work, and explain our proposed model, the MS-Net, a system of single-scale networks that work collectively to construct a prediction for a given sample. We then explain our loss function which couples these networks together. Finally, we explain the coarsening and refining operations used to move inputs and outputs, respectively, between different networks.

2.1 Overview of Convolutional Networks

Our model is comprised by individual, memory inexpensive neural networks which are described in Appendix A. All of the individual networks (one per scale) of our system are composed by identical fully convolutional neural networks (which means that the size of the 3D inputs are not modified along the way). The most important component of a convolutional network is the convolutional layer. This layer contains kernels (or *filters*) of size $(k_{\text{size}})^3$ that slide across the input to create feature maps via the convolution operation:

$$x_{\text{out}} = f\left(\sum_{i=1}^{F} x_{\text{in}} * k_i + b_i\right), \tag{1}$$

where F denotes the total number of kernels (or channels) of that layer, $*$ is the convolution operation, b a bias term, and f is a nonlinear *activation function*. A detailed explanation of these elements is provided in Goodfellow et al. (2016). The elements of these kernels are trainable parameters, and they operate on all the voxels of the domain. These parameters are optimized (or *learned*) during training. By stacking these convolutional layers, a convolutional network can build a model which is naturally *translationally covariant*, that is, a shift of the input image volume produces a shift in the output image volume (LeCun et al. 2015). In this work, we use $k_{\text{size}} = 3$, since it is the most efficient size for current hardware (Ding et al. 2021).

An important concept in convolutional neural networks is the *field of vision*. The field of vision (FoV) dictates to which extent parts of the input might affect sections of the output. The FoV of a single convolutional layer with a $k_{size} = 3$ is of 3 voxels. For the case of a fully convolutional neural network with no coarsening (pooling) inside its layers (the input size is equal to the output size), the FoV is given by the following relation:

$$\text{FoV} = L\left(k_{\text{size}} - 1\right) + 1, \tag{2}$$

where L is the number of convolutional layers of the network, and k_{size} the size of the kernel. For the case of the single scale network architecture used here (see Appendix A for details), the FoV is 11 voxels. This is much smaller than the REV of any of our samples (Fig. 2). It is worth noting that it is not possible to add more layers to increase the FoV and still train with samples that are 256^3 or larger in current GPUs. We now explain MS-Net, which addresses this limitation using a coupled set of convolutional networks.

2.2 Hierarchical Network Structure

The MS-Net is a system of interacting convolutional neural networks that operate at different scales. Each neural network takes as input the same domain at different resolutions through coarsening and refining operations, $\mathbb{C}$ and $\mathbb{R}_m$, which will be explained in detail in Sect. 2.4. Each network is responsible for capturing the fluid response at a certain resolution and to pass it to the next network. This process is visualized in Fig. 3.

What changes between the individual networks is the number of inputs that they receive. The coarsest network receives only the domain representation at the coarsest scale (Eq. 4),

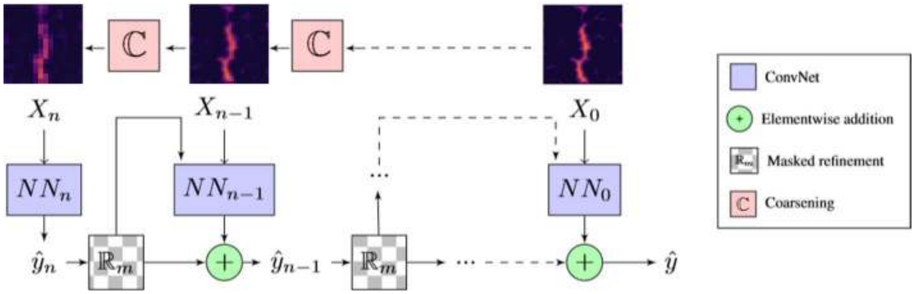


Fig. 3 MS-Net pipeline. Our model consists of a system of fully convolutional neural networks where the feed-forward pass is done from coarse-to-fine (left to right). Each scale learns the relationship of solid structure and velocity response at the particular image resolution. The number of scales (n) is set by the user and all these scales are trained simultaneously. In this figure we are showing the original (finest) scale, the coarsest (n) and the second coarsest ($n-1$). The masked refinement step is explained on Sect. 2.4). X_0 represents the original structure and $\hat{y}$ the final prediction of the model

while the subsequent ones receive two, the domain representation at the appropriate scale, and the prediction from the previous scale (Fig. 3 and Eq. 5). As mentioned above, the input's linear size is reduced by a factor of two between every scale.

Mathematically, the system of networks can be described as such:

$$X_n = \mathbb{C}(X_{n-1}) \tag{3}$$

$$\hat{y}_n = \mathrm{NN}_n(X_n) \tag{4}$$

$$\hat{y}_{n-1} = \mathrm{NN}_{n-1}(X_{n-1}, \mathbb{R}_m(\hat{y}_n)) + \mathbb{R}_m(\hat{y}_n) \tag{5}$$

$$\ldots$$

$$\hat{y} = \hat{y}_0 = \mathrm{NN}_0(X_0, \mathbb{R}_m(\hat{y}_1)) + \mathbb{R}_m(\hat{y}_1), \tag{6}$$

where n indicates the coarsest scale, NN_i the individual neural networks, and $\mathbb{C}()$ and $\mathbb{R}_m()$, the coarsening and refinement operations, respectively. In this system of equations, the input is first coarsened over as many times as there are networks. The neural network that works with the coarsest input (NN_n) has the largest FoV with respect to the original image, and processes the largest scale aspects of the problem. The results of this network are used both as a component of the output of the system, and are made available for the finer scale networks, so that finer-scale, more local processing that can resolve details of flow have access to the information processed at larger scales. The process of coarsening an image progressively doubles the field of vision per scale, yielding the following formula for FoV in MS-net:

$$\mathrm{FoV}_{\text{MS-Net}} = (L\left(k_{\text{size}} - 1\right) + 1) \cdot 2^n, \tag{7}$$

As we stated in the previous section, the finest (first) network has a FoV of 11 voxels. With our proposed system, the second network can see with a window of 22, and so on, with the FoV increasing exponentially with the number of scales. Using these principles, the model is able to analyze large portions of the image that can contain large-scale heterogeneities affecting flow (Fig. 1). The sizes of the windows do not strictly need to be REVs, since

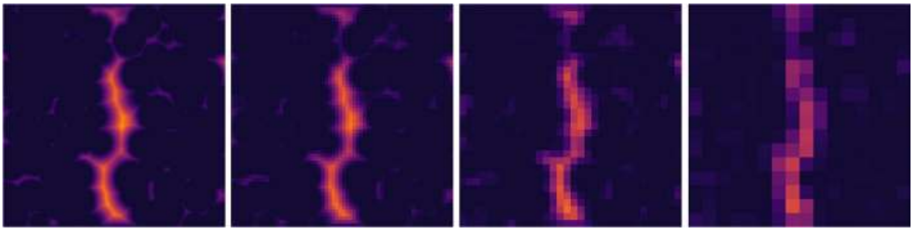


Fig. 4 Original image and three subsequent scales of a fractured sphere pack. The color denotes the distance transform of the original domain. While the computational size of the domain decreases (50% each time) the main features remain present. While most of the pores outside the fracture become less visible, their value is not zero, which still provides valuable information to the coarsest network of the MS-Net

the network still processes the entire image at once. Nevertheless, the bigger the FoV, the easier it is to learn the long range interactions of big heterogeneities affecting flow. Computationally, it would be possible to add enough scales to be able to have FoVs that are close to the entire computational size of the sample (200^3–500^3). Early experiments with a very large number of scales revealed that this limits the performance of the model when applied to small samples.

2.3 Images at Different Scales

Our workflow relies on a multiscale modeling approach. We identify the *scale number* to denote how many times the original image has been coarsened. Hence, *scale* 0 refers to the original image, *scale* 1 is the original image after being coarsened one time, and so on. This process is visualized in Fig. 4. There are two main benefits to this approach, which constructs a series of domain representations with varying level of detail. The first is that coarser networks, which analyze fewer voxels, can be assigned more trainable parameters (larger network widths or more neurons, as depicted in Fig. 1) without incurring memory costs associated with the full sample size. The second is the exponential increase the FoV rendered by this approach (Eq. 7).

For this particular problem of fluid flow, as a proxy for pore-structure, we used the distance transform (also known as the *Euclidean distance*), which labels each void voxel with the distance to the closest solid wall (seen in Fig. 4). We selected this feature because it is very simple and inexpensive to compute, and provides more information than the binary image alone. The fact that no additional inputs are needed makes the MS-Net less dependant on expert knowledge for feature engineering. Note that we treat only the connected void space; the steady-state flow in an isolated void will be zero, and we can safely fill disconnected voids that do not connect to the inlet/outlet system with solid nodes as a cheap preprocessing step.

This distance is related to the velocity field in a non-linear way, which must be learned by the network. Nonetheless, it is possible to visualize how coarser images provide more straightforward information about fluid flow. In Fig. 5, we show input domains against different scales (top row) and corresponding, 2D histograms (bottom row) relating the feature value to the magnitude of the velocity. At scale zero, the distance value is not strongly related to the velocity; for a given distance value, the velocity may still range over more than three orders of magnitude. This deviation from the parabolic velocity profile arises due to the complexity of the porous media. At scale 3, the feature and velocity have been

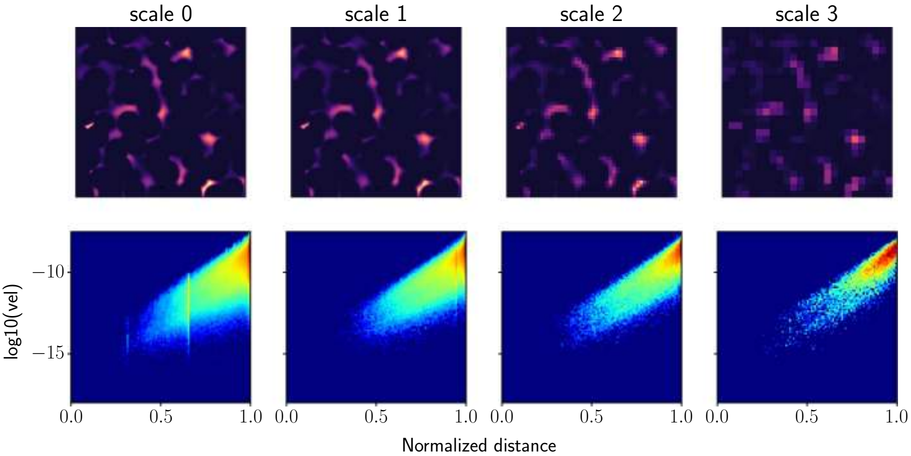


Fig. 5 Top: XY-plane cross-section of the velocity in Z-direction of increasingly coarser scales. Bottom: 2D histograms of the normalized distance transform vs velocity after coarsening steps. As the system is coarsened, the correlation between the distance transform and the velocity becomes more apparent

coarsened several times, and a clearer relationship between the distance and velocity emerges. It is then the job of the *n*th neural network to determine how the 3D pattern of features is non-linearly related to the velocity at this scale, and to pass this information on to networks that operate at a finer scale, as shown in Fig. 3.

2.4 Coarsening and Refinement Operations

We use the term coarsening ($\mathbb{C}$) to describe the operation of reducing the size of an image by grouping a section of neighboring voxels into one voxel (in this case, by averaging). We use the term refinement ($\mathbb{R}$) to denote the operation of increasing the computational size of an image, but not the amount of information (this is also known as image upscaling, but we use the term refinement to avoid potential confusion with upscaling in reservoir engineering or other surrogate modeling).

The coarsening and refining operations should have the following properties applied to data z (i.e., input or output volumes):

$$\langle z_n \rangle = \langle \mathbb{C}(z_{n-1}) \rangle \tag{8}$$

$$z_n = \mathbb{C}(\mathbb{R}_m(z_n)), \tag{9}$$

the angle brackets $\langle\rangle$ represent the volumetric average over space, and the masked refinement $\mathbb{R}_m()$ projects solutions from a coarse space back into the finer resolution space while assigning zero predictions to regions that are occupied by the solid. The first equation indicates that coarsening should preserve the average prediction, and the second says that refinement should be a pseudo-inverse for coarsening, that is, if we take an image, refine it, and then subsequently coarsen it, we should arrive back at the original image. Note that the opposite operation—coarsening followed by refinement—cannot be invertible, as the coarser scale image manifestly contains less information than the fine scale one.

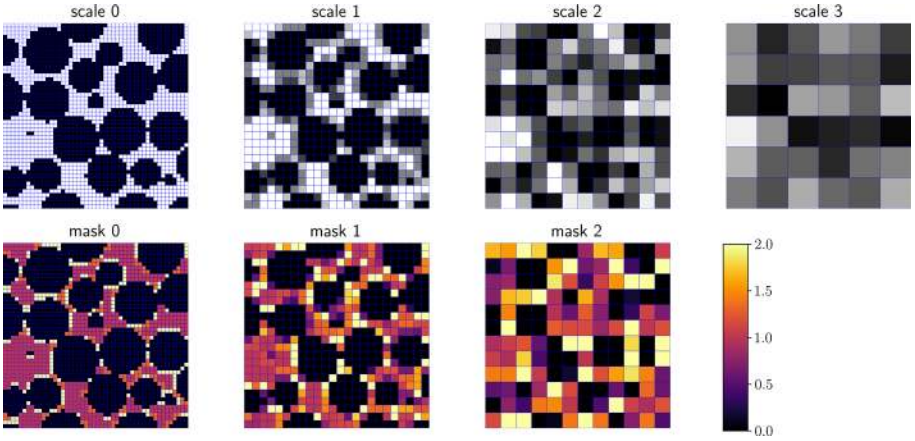


Fig. 6 Schematic of the coarsening and masking process. Top: Starting from the original domain, a series of coarsening steps are performed, every 2^3 neighboring voxels are averaged to produce one voxel at the following scale. Structural information is lost along the way. Bottom: Masks for each scale, which re-weight a naive refinement operator. These masks have larger weights in regions where the prediction must be redistributed, near the boundaries with solid nodes, and is zero in regions that correspond entirely to solid nodes. For example, if the image of scale 3 is refined using nearest-neighbors and multiplied by the mask 2, the image at scale 2 would be fully recovered

The coarsening operation is simple. As mentioned in Sect. 2.3, we first coarsen our input domain n-times. Coarsening is applied via a simple nearest neighbor average; every 2^3 region of the image is coarsened to a single voxel by averaging. This operation is known as *average pooling* in image processing.

The refinement operation is more subtle. There exists a naive nearest-neighbors refinement algorithm, wherein the voxel value is replicated across each 2^3 region in the refined image. However, this presents difficulties for prediction in porous media—namely, that if this operation is used for refinement, flow properties from coarser networks will be predicted on solid voxels where they are by definition zero (refer to Fig. 13), and the fine-scale networks will be forced to learn how to cancel these predictions exactly. Early experiments with this naive refinement operation confirmed that this behavior is problematic for learning.

Instead, we base our refinement operation on a refinement mask derived from the input solid/fluid nodes. This is performed such that, when refined back to the finest scale, the prediction will be exactly zero on the solid nodes and constant on the fluid nodes, while conserving the average. We refer to this masked refinement as $\mathbb{R}_m$. This requires computing refinement masks that re-weight regions in the refined field based on the percentage of solid nodes in each sub-region. Refinement masks for a particular example are visualized in Fig. 6. An example calculation and pseudo-code for this operation is given in the A.3. Then, the masked refinement operation can simply be computed as naive refinement, followed by multiplication by the mask. Figure 14 demonstrates the difference between the operations by comparing naive refinement with masked refinement.

The masked refinement operation is cheap and parameter-free; nothing needs to be learned by the neural network, unlike, for example, the transposed convolution (Goodfellow et al. 2016). We thus find it an apt way to account for the physical constraints posed

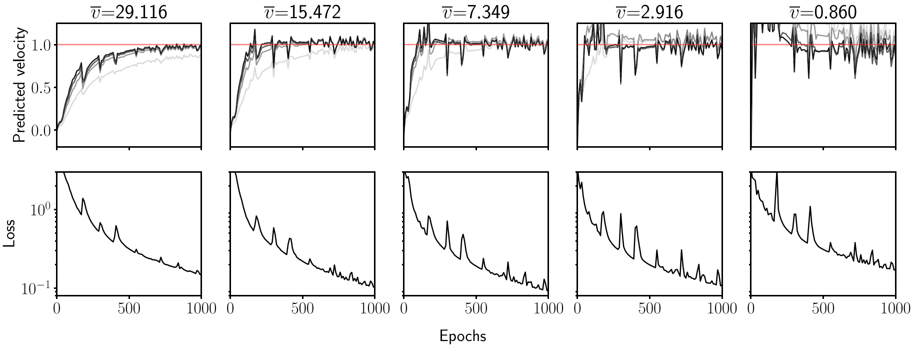


Fig. 7 Velocity prediction per scale and sample loss $\mathcal{L}_s$ for the training samples, organized by column. (top row) Normalized mean velocity (that is, $\bar{\hat{v}}/\bar{v}$ at each scale. Coarser scales are shown with lighter lines. Note that the average velocity is predicted mostly by the coarsest scale, and refined by further scales. (bottom row) Corresponding values of the loss function for each sample during training. Although the permeability ($k \propto \bar{v}$) of the samples spans orders of magnitude, the network assigns roughly the same importance to them during training using a well-normalized loss function

by fields defined only within the pore. The masked refinement operation is also the unique refinement operation such that when applied to the input binary domain, coarsening and refinement are true inverses; the masked refinement operation recovers the original input domain from the coarsened input domain.

2.5 Loss Function

To train the MS-Net, the weights of the convolutional kernels and biases of the network (Eq. 14) are optimized to minimize the following equation based on the mean squared error for every sample s at every scale i between the prediction at that scale $\hat{y}_{i,s}$ and the true value coarsened at that scale $y_{i,s}$. The full loss $\mathcal{L}$ can be broken in to contributions for each image $\mathcal{L}_s$, and scale, $\mathcal{L}_s^i$:

$$\mathcal{L}_s^i = \frac{\langle (y_{i,s} - \hat{y}_{i,s})^2 \rangle}{\sigma_{y_s}^2} \tag{10}$$

$$\mathcal{L} = \sum_s \mathcal{L}_s = \sum_s \sum_i \mathcal{L}_s^i \tag{11}$$

$$= \sum_{s=0}^{S} \sum_{i=0}^{n} \frac{\langle (y_{i,s} - \hat{y}_{i,s})^2 \rangle}{\sigma_{y_s}^2}, \tag{12}$$

where n is the total number of scales, and S the number of samples, $\langle ... \rangle$ denote the spatial averaging operation. This equation accounts for the variance in the true velocity, $\sigma_{y_s}^2$, in order to weight samples that contain very different overall velocity scales (related to permeability) more evenly. Since the coarsest scale is implicitly present in the solution at every scale (Eq. 6), the coarsest network is encouraged to output most of the magnitude of the velocity Fig. 7. This loss function is also useful to be able to train with samples of varying structural heterogeneity (and fluid response), since the mean square error is normalized with the sample variance to obtain a dimensionless quantity that is consistent for every sample. The velocity variance does not scale all the samples to the exact same distribution, but it is a good first order approximation to allow training models with very different velocity distributions.

3 Data Description: Flow Simulation

To train and test our proposed model, we carried-out single-phase simulations using our in-house multi-relaxation time D3Q19 (three dimensions and 19 discrete velocities) lattice-Boltzmann code (D'Humieres et al. 2002). Our computational domains are periodic in the z-direction, where an external force is applied to drive the fluid forward simulating a pressure drop. The rest of the domain faces are treated as impermeable. The simulation is said to achieve convergence when the coefficient of variation of the velocity field is smaller than $1\text{x}10^{-6}$ between 1000 consecutive iterations. We run each simulation on 96 cores at the Texas Advanced Computing Center. The output of the LB solver is the velocity field in the direction of flow (here, the z-direction). To calculate the permeability of our sample, we use the following equation:

$$k_{\text{sample}} = \frac{\overline{v}\mu}{\frac{dP}{dl}}\left(\frac{dx}{dl}\right)^2, \tag{13}$$

where $\overline{v}$ is the mean of the velocity field in the direction of flow, μ is the viscosity of the fluid, $\frac{dP}{dl}$ is the pressure differential applied to the sample, and $\frac{dx}{dl}$ is the resolution of the sample (in meters per voxel). Although we used the LBM to carry-out our simulations, the following workflow is method agnostic. It only relies on having a voxelized 3D domain with its corresponding voxelized response.

4 Results

Below we will present two computational experiments (porous media and fractures) that were carried-out to show how the MS-Net is able to learn from 3D domains with heterogeneities at different scales. In the first subsection, we will show to what extent the MS-Net is able to learn from very simple sphere-pack geometries to be able to accurately predict flow in a wide range of realistic samples from the Digital Rock Portal (Prodanovic et al.). It is worth noting that simulating the training set took less than one hour per sample, and training the model took seven hours, while some samples in the test set took over a day to achieve convergence through the numerical LBM solver. In the second experiment, we show that training using two fracture samples of different aperture sizes and roughness parameters is enough to estimate permeability for a wider family of aperture sizes, and roughnes.

4.1 Training the MS-Net with Sphere Packs

To explore the ability of the MS-Net to learn the main features of flow through porous media, we utilize a series of five 256^3 numerically dilated sphere packs (starting from the original sphere pack imaged by Finney and Prodanovic (2016)) to train the network (Fig. 15). The porosity of the training samples range from 10 to 29%, and their permeability from 1 to 37 Darcy. For reference, the individual simulation times to create the training samples range from 10 to under 50 minutes to achieve convergence.

Our model consists of four scales, using 2^{2n+1} filters per scale (2 in the finest model and 128 in the coarsest). During training, each sample is passed through the model (as shown in Fig. 3), and then the model optimizes its parameters (the numbers in the 3D filters and the biases) to obtain a functional relationship between the 3D image and the velocity field by minimizing the loss function (Eq. 12). The MS-Net can be trained end-to-end since all included operations are differentiable (including the coarsening and masked refinement) with respect to the model's weights.

In short, we are looking to obtain a relation of the form of velocity v_z as a function of the distance transform feature x, that is, $v_z = f(x)$.

The network was trained for 2500 epochs, with a batch size of all five of the samples (i.e., full-batch training). This process took approximately seven hours. To highlight the order of magnitude of the loss function during training, the first 1000 epochs of the training are shown in Fig. 7. This illustrates that magnitude of the loss for each sample is similar, despite the very different velocity scales associated with each sample. We also tried augmenting the training dataset utilizing 90 degree rotations about the flow axis, but no benefits were observed. The loss function value per sample and the mean velocity per scale of the network are plotted in Fig. 7. As seen in the top row of the figure, the coarsest scale is responsible for most of the velocity magnitude, and finer-scale networks make comparatively small adjustments. The bottom row of plots shows the loss for each sample used for training. We note that the normalization of our loss (Eq. 12) puts the loss for each sample on a similar scale, despite the considerable variance in porosity and permeability between samples.

Once trained, the network produces very accurate predictions on the training samples. Figure 8a shows predicted and true velocity fields, and relative error, for a cross section of one of the training example. Figure 9a shows a histogram of the true vs. predicted velocity across all pore-space voxels, along with a plot of the relative change in fluid flux across x-y slices of the sample. However, performance on training data does not necessarily translate to useful performance on the vast variety of real-world microstructures, and as such we endeavor to assess the trained MS-net on many test microstructures compiled from various sources. Figure 8, rows b through e, and Fig. 9, rows b through e, show velocity field performance on various samples from test sets, which we examine in the following sections.

4.1.1 Fontainebleau Sandstone Test Set

Training to the sphere-pack dataset reveals that the model can learn the main factors affecting/contributing to flow through permeable media. To assess the performance of the trained MS-Net, we used Fontainebleau sandstones (Berg 2016) at different computational sizes (256^3 and 480^3). The cross sections of these structures are visualized in the right panel of Fig. 15. Overall results are presented in Table 1, and flow field plots for a single sample are presented in Fig. 8b and 9b. The relative percent error of the permeability can be calculated as $e_r = |1 - \frac{k_{pred}}{k}|$. The typical accuracy of the permeability is approximately within 10%. One remarkable fact is that the model retains approximately the same accuracy when applied to 480^3 samples as 256^3 samples.

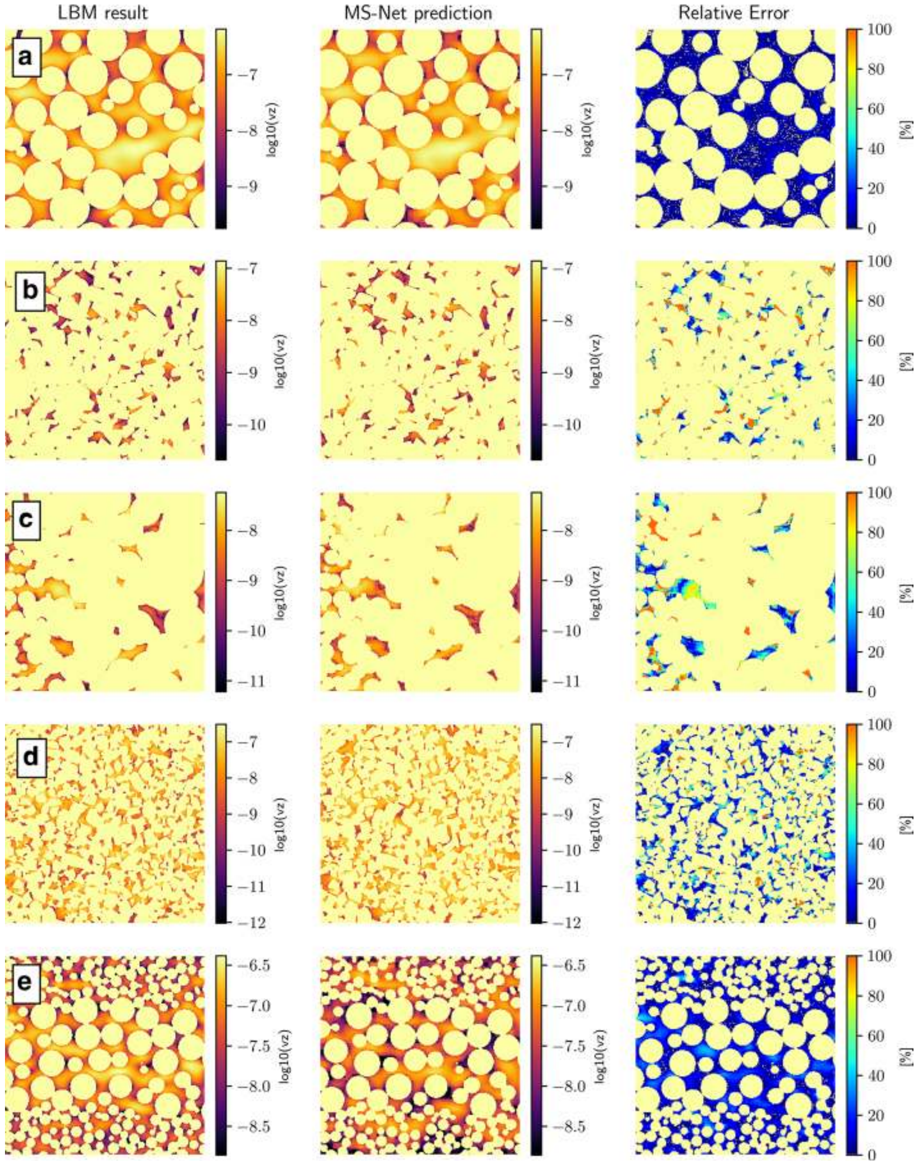


Fig. 8 Cross-sectional view of simulation result, the predictions of the model trained with sphere packs and the relative error. The samples displayed are the following: **a** Training sphere pack with porosity 29%, **b** Fontainebleau Sandstone with a porosity of 9%, **c** Porosity gradient sample from Group I, **d** Castlegate sandstone from Group II, and **e** Propped fracture from Group III

It is worth noting that the simulation of the sample with a porosity of 9.8 % took 13 hours to converge; this single sample takes more computational effort than the entire construction of training data and the training the model together.

Fig. 9 2D histogram of the true velocities vs the predictions of the model trained with sphere packs in ▶ log10, and the percentage change in the flux per slice of the domain. The percentage change of flux shows how much the fluid flux varies from slice-to-slice across the sample. In our results, we can see percentage changes up to 10%. Each row corresponds to the sample samples as shown in Fig. 8

4.1.2 Additional Test Predictions on more Heterogeneous Domains

To assess the ability of the model trained with sphere packs to predict flow in more heterogeneous materials, we tested samples of different computational size and complexity. We split the data into three groups according to their type:

- Group I: Artificially created samples: In this group, we include a sphere pack with an additional grain dilation (that lowered the porosity) from the lowest porosity training sample[1], a vuggy core created by removing 10% of the matrix grains from the original sphere pack (Finney and Prodanovic 2016) to open up pore-space and create disconnected vugs (Khan et al. 2019, 2020), then the grain were numerically dilated two times to simulate cement growth, a sample made out of spheres of different sizes where the porosity at the inlet starts at 35% and it decreases to 2% at the outlet, and this last sample reflected in the direction of flow.
- Group II: Realistic samples: Bentheimer sandstone (Muljadi 2015), an artificial multiscale sample (microsand)(Mohammadmoradi 2017), Castlegate sandstone (Mohammadmoradi 2017). The sizes where selected such that they were an REV.
- Group III: Fractured domains: Segmented micro-CT image of a fractured carbonate (Prodanovic and Bryant 2009), layered bidispered packing recreating a propped fracture (Prodanovic et al. 2010), and a sphere pack where the spheres intersecting a plane in the middle of the sample where shifted to create a preferential conduit, and this same structure rotated 90 degrees in the direction of flow (so that the fracture is in the middle plane of the flow axis and deters flow) (Prodanovic et al. 2010).

The lowest porosity sample (porosity of 7.5%) took 26 hours running on 100 cores to achieve convergence. Besides an accurate permeability estimate, another measure of precision if the loss function value at the finest scale (from Eq. 12). These two are related, but not simple transformations of each other. The loss function provides a volumetric average of the flow field error. We normalized this value using the sample's porosity to obtain a comparable quantity, which results in a quantity that is roughly the same for all samples. Visualizations of some of these samples are shown in Fig. 16, and the prediction results are shown in Table 2.

Table 2 reveals remarkable performance on the variety of geometries considered. Samples from all groups are predicted very well, with permeability errors for the most part within about 25% of the true value, through samples ranging by three orders of magnitude in permeability. Flow field errors for one sample from each group are shown in Figs. 8c–e, 9c–e.

[1] We tried to carry-out a simulation of a structure with an additional grain dilation (4.7 % porosity); however, the job timed-out after 48 hrs without achieving convergence.

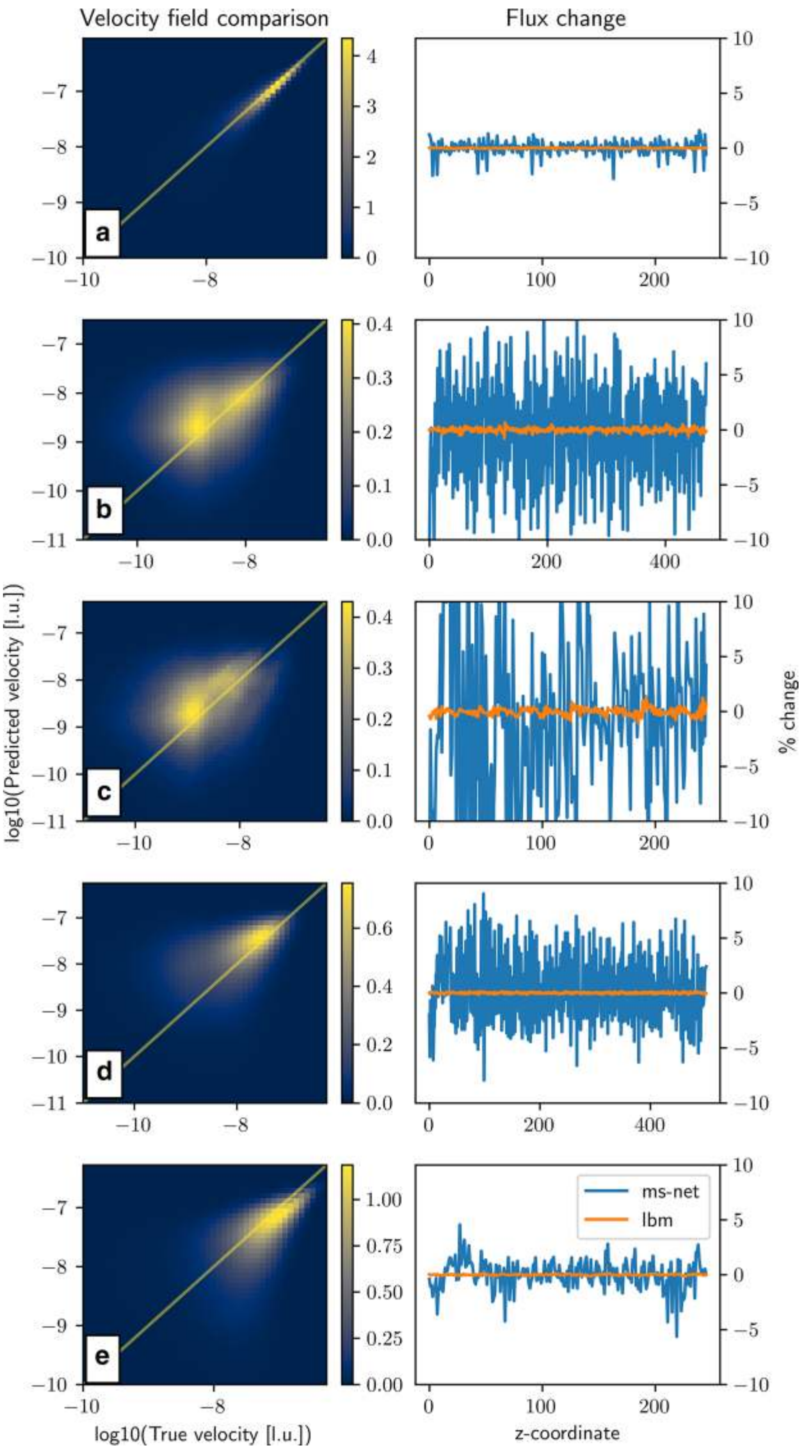

Velocity field comparison
Flux change
a
b
c
d
e
log10(Predicted velocity [l.u.])
log10(True velocity [l.u.])
% change
z-coordinate
ms-net
lbm
−7
−8
−9
−10
−11
−10
−8
4
3
2
1
0
0.4
0.3
0.2
0.1
0.0
0.6
0.4
0.2
0.0
1.00
0.75
0.50
0.25
0.00
10
5
0
−5
−10
0
100
200
400

Table 1 Results of the Fontainebleau sandstone test set

Porosity (%)	Size	k (m^2)	k_{pred}/k
10.4	256^3	8.21E−13	0.94
9.8	480^3	6.50E−13	1.08
12.6	256^3	1.47E−12	1.10
12.4	480^3	1.28E−12	0.98
15.4	256^3	2.72E−12	1.03
15.2	480^3	3.18E−12	0.99
18.0	256^3	5.28E−12	0.98
17.4	480^3	4.95E−12	1.02
24.7	256^3	1.30E−11	1.03
24.3	480^3	1.38E−11	0.98

We show the true permeability k, calculated using the LBM, and the ratio between the true permeability and the prediction of our model k_{pred}/k

Table 2 Results of the predictions on the test set

Group	μCT	Sample	Size	ϕ	k (m^2)	$\mathcal{L}_s^0 * \phi$	k_{pred}/k
I		Porosity gradient	256^3	9.7%	1.78e−13	0.151	0.98
		Reflected porosity gradient	256^3	9.7%	1.78e−13	0.154	0.98
		Synthetic vuggy core	256^3	32.1%	2.71e−11	0.148	0.94
		Tight sphere pack	480^3	7.5%	3.80e−13	0.077	1.01
II	✓	Bentheimer sandstone	256^3	20.6%	5.10e−12	0.092	1.08
		Multiscale microsand	480^3	29.8%	8.48e−12	0.196	1.23
	✓	Castlegate sandstone	512^3	20.4%	4.25e−12	0.235	1.38
III	✓	Carbonate fracture	256^3	9.2%	2.12e−12	0.098	0.87
		Propped fracture	256^3	37.5%	9.55e−11	0.140	0.73
		Orthogonal fractured sphere pack	256^3	41.2%	1.04e−10	0.170	0.95
		Parallel fractured sphere pack	256^3	41.2%	5.14e−10	0.347	**0.20**

We additionally show here the ratio between the loss (Eq. 12) and the porosity, which is another measure of accuracy in the active cells of the prediction. In bold, the worst performing sample using the model trained on sphere packs

Two notable failure cases emerged. In the first, the Castlegate sandstone, we find that the flow field prediction is still somewhat reasonable, as visualized in Figs. 8d and 9d. The largest failure case (highlighted in bold in Table 2), is the fractured sphere pack with a fracture parallel to fluid flow. In this case, the model is not able to provide an accurate flow field due to the difference in flow behavior that a big preferential channel (like this synthetic fracture) imposes compared with the training data (sphere packs), and as a result the predicted permeability is off by a factor of 5. Likewise, the sample also has the highest loss value. However, since no example of any similar nature is found in the training set, we investigate in the following section the ability of the model to predict on parallel fractures when presented with parallel fractures during training.

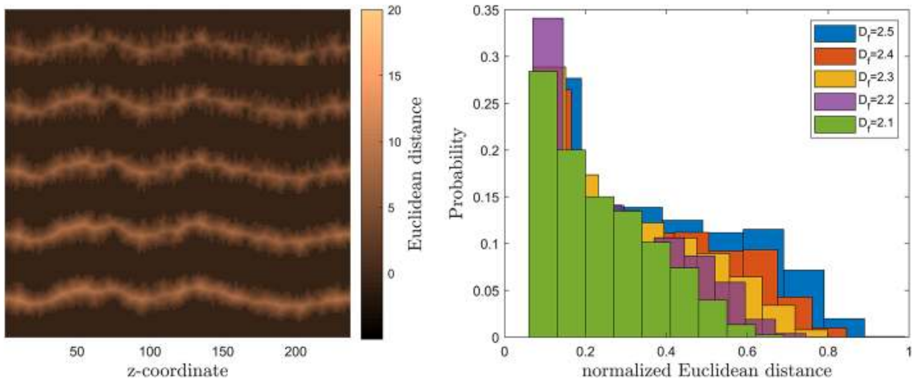


Fig. 10 Five synthetic fracture cross-sections and their distance transform histograms. To the left, The distance transform map of five fractures with different roughness is shown. All of the domains are shown in one merged figure for brevity. All the fracture properties stay constant while their fractal exponent (D_f) decreases from 2.5 to 2.1 (top to bottom). To the right, the histogram of the distance transform of entire domains is shown. Rougher fractures have wider distributions

Table 3 Results of training and testing in different fractures

Train	$\overline{Ap}$	D_f	k (m^2)	k_{pred}/k
	44	2.1	9.41e−10	1.03
		2.2	7.54e−10	1.08
		2.3	5.69e−10	1.01
		2.4	4.17e−10	1.03
✓		2.5	2.97e−10	1.01
✓	22	2.1	2.21e−10	1.007
		2.2	1.89e−10	1.003
		2.3	1.52e−10	1.01
		2.4	1.18e−10	1.04
		2.5	8.98e−11	0.98
		Propped fracture	9.57e−11	1.17
		Parallel fractured sphere pack	5.13e−10	**1.03**

The first column indicates if the sample was part of the training set, followed by the mean aperture $\overline{Ap}$, fractal exponent D_f and permeability k. In bold, the worst performing sample of Table 2

4.2 Training the MS-Net with Fractures

Since the MS-Net is able to see the entire domain at each iteration, we carried-out an additional numerical experiment with domains hosting single rough fractures from a study of the relationship between roughness and multiphase flow properties (Santos and Jo 2019; Santos et al. 2018a). Alqahtani et al. (2021) identified flow through fractures as an outstanding problem in deep-learning of porous materials.

Table 4 Prediction size achieved in three different GPUs

GPU	Memory	Size achieved
Nvidia P100	12 Gb	640^3
Nvidia M6000	24 Gb	704^3
Nvidia A100	40 Gb	832^3

The domains were created synthetically using the model proposed by Ogilvie et al. (2006), where the roughness of the fracture surfaces is controlled by a fractal dimension D_f. The cross sections of the domains and their histograms are shown in Fig. 10. We utilize two sets of five fractures fractures with increasing roughness D_f. Within each group, each fracture has a constant mean fracture aperture $\overline{Ap}$ and therefore the porosity of each sample in a group is identical. The total computational size of these is 256× 256×60. We trained our model using two of these synthetic fractures, one from each group, and tested them on the other 8 fractures. The results are shown in Table 3.

We selected two samples with different mean aperture and roughness exponent so that the network might learn how these factors affected flow. From the results of Table 3, we can conclude that the network is able to extrapolate accurately to construct solutions of the flow field for a wide variety of fractal exponents. The training fractures have permeability of 224 and 301 Darcy, whereas accurate test results are found ranging between 91 and 953 Darcy. This gives strong evidence that the model is able to distill the underlying geometrical factors affecting flow, even when the coarse-scale view of the pore is quite similar.

We contrast the machine learning approach with classical method such as the cubic law (Tokan-Lawal et al. 2015). For these synthetically created fracture, the cubic law would return a permeability value that depends only on the aperture size, whereas LBM data reveals that the roughness could influence the permeability by a factor of 3. There have been a number of papers attempting to modify the cubic law for a more accurate permeability prediction. However, there is evidence that those predictions could be off by six orders of magnitude (Tokan-Lawal et al. 2015). There are also other approaches in line with our hypothesis that a fracture aperture field should be analyzed with local moving windows (Oron and Berkowitz 1998). This model could be used to estimate more accurately the petrophysical properties of fractures for hydraulic fracturing workflows (Xue et al. 2019; Park et al. 2021).

5 Discussion

We have shown the MS-Net performing inference in volumes up 512^3, chosen to obtain the LBM solutions in a reasonable time-frame. The MS-Net can be scaled to larger systems on a single GPU. Table 4 reports the maximum size system which a forward pass of the finest-scale network was successful for various recent GPU architectures, without modifying our workflow. Additional strategies such as distributing the computation across multiple GPUs, or pruning the trained model (Tanaka 2020; Li et al. 2017) might be able to push this scheme to even larger computational domains. For all architectures tested, the prediction time was on the order of one second, whereas LBM simulations on a tight material may take several days to converge, even when running on hundreds of cores.

The number of scales used could be varied. For our experiments, we chose to train a model with four scales, since we did not see an increase in accuracy with more scales. This number is a parameter that could be explored in further research. The FoV of the coarsest network is of 88 voxels wide, and the model itself operates on the entire domain simultaneously, rather than on subdomains. For comparison, the FoV of the PoreFlow-Net (Santos et al. 2020b) is of 20 voxels, and operated on subdomains of size 80^3 due to memory limitations. We utilize $22n + 1$ filters per scale (2 in the finest network and 128 in the coarsest).

We also believe that our work could be also utilized for compressing simulation data, since, as shown in Fig. 7, a single model is able to learn several simulations with a high degree of fidelity to make them more easily portable (also called *representation learning* in deep learning). A training example is on the order of 500 Mb of data in single precision float-point, whereas the trained model is approximately 25 Kb. Thus, when training to a single 512^3 example, the neural network encodes the solution using approximately 2×10^{-4} bytes per voxel; it is tremendously more efficient than any floating point representation. One would also need to keep the input domain to recover the solution, but this is itself a binary array that is more easily compressed than the fluid flow itself. For example, we applied standard compression methods to the binary input array for the Castlegate sandstone, which was then fit into 2.4 MB of memory.

6 Conclusions and Future Work

It is well-established that semi-analytical formulas or correlations derived from experiments can fail to predict permeability by orders of magnitude. Porosity values alone can be misleading due to the fact that this does not account for how certain structures affect flow in a given direction, or due to the presence of heterogeneities. However, going beyond simple approximations is often expensive. We have presented MS-Net, a multiscale convolutional network approach, in order to better utilize imaging technologies for identifying pore structure and associate them with flow fields. When training on sphere packs and fractures, MS-Net learns complex relationships efficiently, and a trained model can make new predictions in seconds, whereas new LBM computations can take hours to days to evaluate.

We believe that it would be possible to train the MS-Net using more and more diverse data to create a predictive model that could be able to generalize to more domains simultaneously (for example, unsaturated soils, membranes, mudrocks, and kerogen). This could be done using active learning principles, carrying out simulations where the model has a low degree of confidence in its prediction, such as in Santos et al. (2020a), to examine a vast variety of possible training samples and only compute physics-based flow simulations on a targeted set.

The MS-Net architecture is an efficient way of training with large 3D arrays compared to standard neural network architectures from computer vision. Although this model is shown to return predictions that are accurate, there are still high local errors. A plausible solution is to use the model prediction as the initial input for a full-physics simulation, as shown in Wang et al. (2021b). This workflow can be specially efficient if the simulator can take advantage GPU computing.(McClure et al. 2014, 2021)

On the other hand, there are desirable physical properties that might be realized by a future variant, such as mass or momentum continuity. One avenue of future work could be to focus on designing conservation modules for the MS-Net using such hard constraints for ConvNets (Mohan et al. 2020). An important hurdle of applying these techniques in porous media problems is that the bounded domains make the implementation more challenging.

Another important area of future work would be to address data from different scientific domains. This includes similar endeavors such as steady-state multiphase flow, waved propagation through a solid matrix, and component transport in porous media. The model could also be applied to other 3D problems, such as seismic inversion (Cho et al. 2018; Cho and Jun 2021), astronomical flows (van Oort et al. 2019), or the flow of blood through highly branched vessel structures.

Lastly, we believe that an important endeavor is to create more realistic domains, with multiscale features such as fractal statistics. One avenue to pursue such methods is the Generative Adversarial Network (GAN), another ML technique which allows a generator model to learn to create new data by fooling a discriminator model (the *adversary*) that is trained to distinguish between real data and the Generator's outputs. The multiscale technique has been applied to many real-world datasets such as human faces, but has not, to our knowledge, been used to construct synthetic porous media.

Appendix A Single neural network description

Each network of our system (one per scale) is composed by fully convolutional networks (which means that the dimensions of the 3D inputs are not modified along the way). Each of them is composed by stacks with the following layers:

- 3D convolution with a 3^3 kernel: This layer contains kernels (or *filters*) of size 3^3 that are slid across the input to create feature maps via the convolution operation:

$$x_{\text{out}} = \sum_{i=1}^{F} x_{\text{in}} * k_i + b_i, \tag{14}$$

where F denotes the number of kernels (or output channels) of that layer, $*$ is the convolution operation and b a bias term.

- Instance Normalization (Ulyanov et al. 2016): This layer normalizes its inputs to have a mean of zero and a standard deviation of one. This facilitates training a model with samples that have strong velocity contrasts (different orders of magnitude). This is done to every sample using their individual statistics and no trainable parameters.

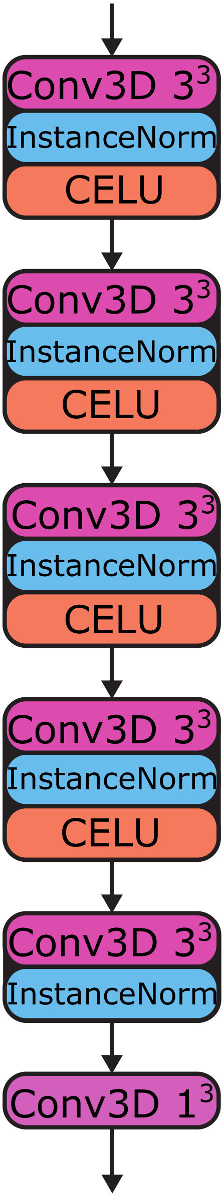
$$x_{\text{out}} = \frac{x_{\text{in}} - \bar{x}}{\sqrt{\sigma^2 + \epsilon}}, \tag{15}$$

where $\bar{x}$ is the sample mean and σ its standard deviation, ϵ is a small constant to avoid divisions over zero. This layers allows better flow of information (by constraining the mean and the standard deviation of the outputs) and reduces the risk of training diverging.

- Continuously Differentiable Exponential Linear Unit (*CeLU*) (Barron 2017): This layers helps to build nonlinear relationships (like the one between pore-structure and velocity field, shown in Fig. 5). All the data that passes through this layer is transformed using the following equation:

$$x_{out} = \max(0, x_{\text{in}}) + \min\left(0, \alpha \cdot (e^{\frac{x_{\text{in}}}{\alpha}} - 1)\right), \tag{16}$$

Fig. 11 Schematic of a single-scale network



where α is set to 2. We utilize this function because it speeds-up and improves training by virtue of not having vanishing gradients and by having mean values near zero. It also able to output negative values (unlike the *ReLU*) The outputs of this network are constrained from minus two to infinity.

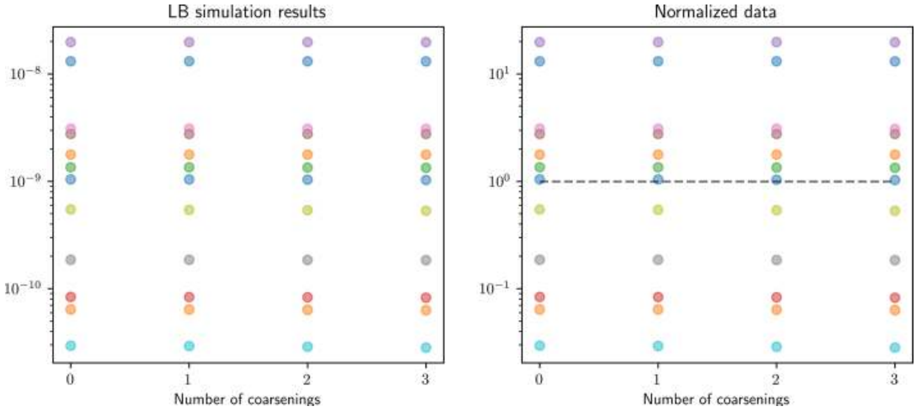


Fig. 12 (left) Mean velocity per scale of the LB simulation results (in lattice units). Each dot represents a sample. (right) Mean velocity after normalizing the data

The last layer of each network is a 3D convolution with a 1^3 kernel: The 1^3 kernel acts as a linear regressor which reduces the dimensionality of the output to one single 3D image (in our case, the velocity field). This is done to weight all the feature maps from the previous layer (Fig. 11) and output a single 3D matrix (in this case, the velocity at that particular scale). The three components of the velocity tensor (vx,vy, and vz) can be predicted if this layer is set to have three filters.

The fourth block of our network does not include an activation function because we would like to give the network expressive power to be able to output negative velocities. Every convolutional layer includes a bias term. This system is visualized explicitly in Fig. 11.

Normalization of the Data and Initialization of the Network Parameters

We first start the training workflow by coarsening the initial inputs n times (depending on the number of scales desired Sect. 2.3). Then, we center the velocity of all the training set to be near one by dividing the LB simulation results with a constant. (This procedure is shown in Fig. 12). This has the advantage of not having to compute and store the summary statistics of the training set (as opposed to default normalization approaches). It also preserves the solid values as zero.

We have also observed that if we scale the weights of the last layer of the coarsest network to output results that are close to one (the mean velocity of our normalized data), the training exhibited a speed-up of several hours, since the initial prediction is a closer approximation to the solution compared to the default initialization scheme (He et al. 2015).

Coarsening (Pooling) Operation

The coarsening (or *pooling*) operation is defined as:

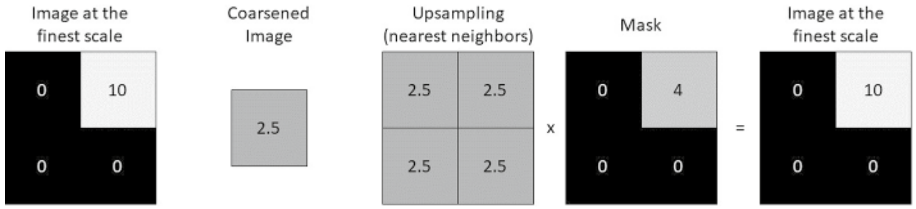


Image at the finest scale
0
10
0
0
Coarsened Image
2.5
Upsampling (nearest neighbors)
2.5
2.5
2.5
2.5
x
Mask
0
4
0
0
=
Image at the finest scale
0
10
0
0

Fig. 13 Schematic of the coarsening and refining process

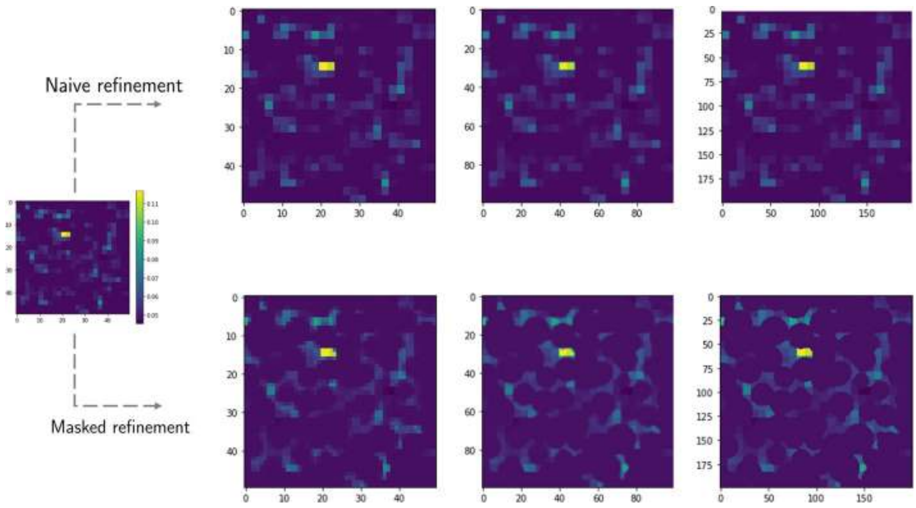


Fig. 14 Comparison between upscaling and the proposed masked upscaling. Even when all the details are not fully recovered with the method, it preserves the mean velocity predictions from the coarse scale and it does not allow fluid in the solid space. Note how the dimensions of the image change (not to scale)

$$\begin{aligned} k &= J_d \cdot \frac{1}{2^d} \\ \mathbb{C}(x_{\text{in}}) &= x_{\text{in}} \underset{2}{*} k \end{aligned} \tag{17}$$

where J is an array of all ones, d the number dimensions of the problem, and $\underset{2}{*}$ the convolution operation with a stride of 2.

Mask Calculation

We use the following python code to generate the masks of each sample at the beginning of the training cycle:

```python
masks      = [None]*(num_scales-1)
pooled     = [None]*(num_scales)
pooled[0]  = binary_image
for scale in range(1,num_scales):
    pooled[scale] = AvgPool3d(kernel_size = 2)(pooled[scale-1]) # coarsen
    denom = pooled[scale].clone() # calculate the denominator for the mask
    denom[denom==0] = 1e8         # regularize to avoid division over zero
    for ax in range(3):           # refine using nearest neighbors
        denom=denom.repeat_interleave( repeats=2, axis=ax )
    masks[scale-1] = pooled[scale-1]/denom
```

Listing 1 Python pseudo-code example for obtaining the refinement masks.

As an example, parting from a 2D region of an image of 2x2 pixels where 3 out of 4 pixels are solid, when this image is coarsened all these pixels are averaged into one value. When this coarsened image is refined using the masked approach, the only void pixel would get the value of the nearest neighbors matrix times 4 (Fig. 13).

End-to-End Model Training

1. Initialize MS-Net: In this paper we used a model with n=4 scales (4 networks from Appendix A). Each of these networks has 2^{2s+1} convolutional kernels in each of their 5 layers. The network that works with the finest scale has 2 kernels while the one that receives the coarsest input has 128.

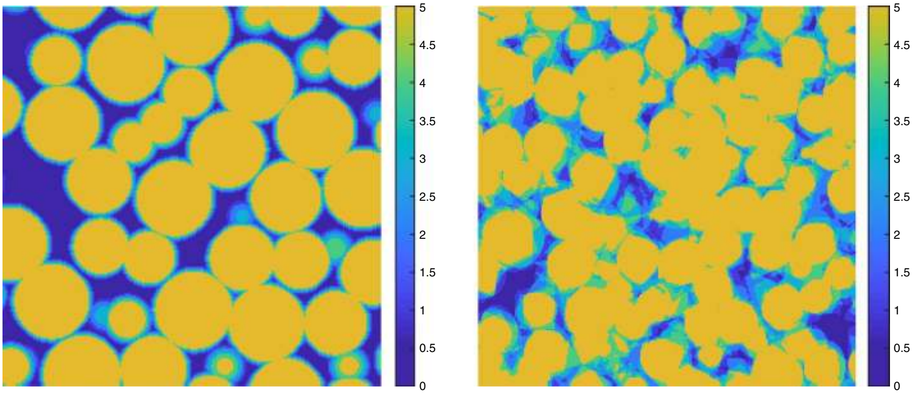


Fig. 15 Superimposed cross sections of the sphere pack training set (left) and the sandstone test set (right). The image shows the five binary samples per set which were superimposed for visualization purposes. Since the geometries are binary (zeros in the void space and ones in the solids) is possible to sum all five to obtain a new array ($im = sand_1 + sand_2 + sand_3 + sand_4 + sand_5$). The highs of the color bar stand for solids that are present in every domain while the lows are sections that are only present in the lower porosity samples

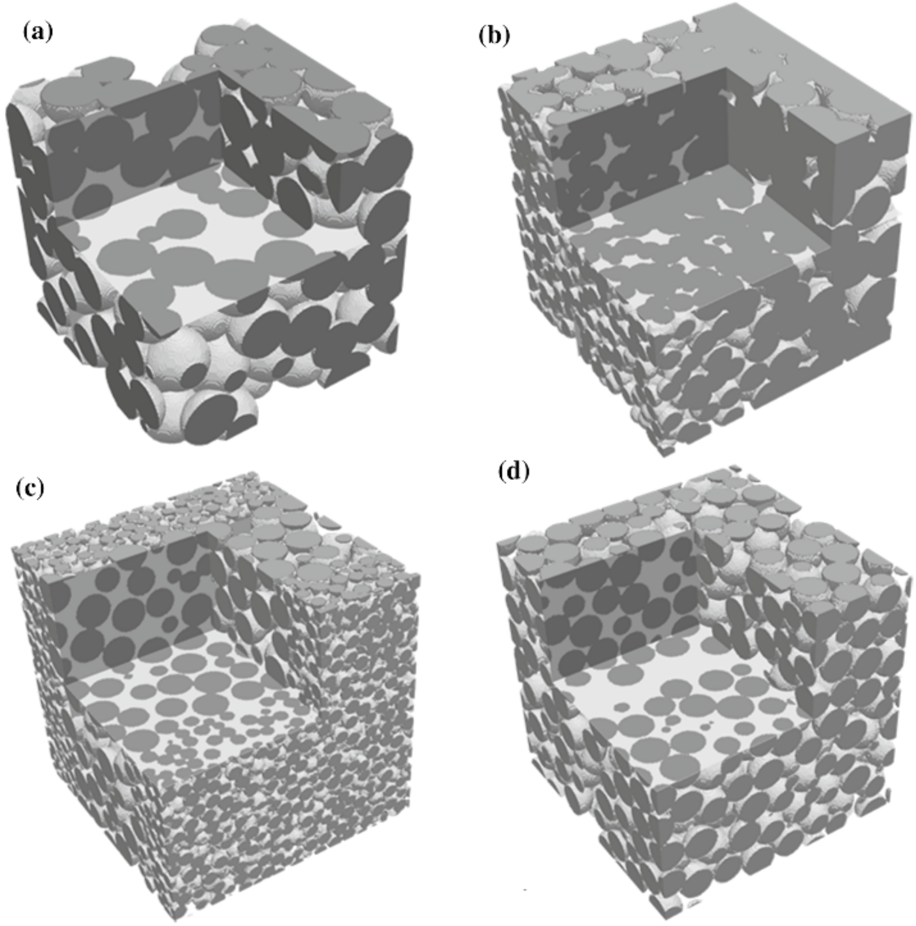


Fig. 16 Examples from the additional test set **a** Vuggy core, **b** Porosity gradient, **c** Propped fracture, **d** Fractured sphere pack. The computational size of these samples is 256^3

2. Calculate the velocity variance ($\sigma^2_{y_s}$) of each training sample (s). This is used in the loss function (Eq. 12).
3. Coarsen the x (Euclidean distance) and the y (velocity) of each sample $n-1$ times and calculate the masks (Appendix A.3).
4. Pass the coarsest image (x_n) through the network NN_n to obtain $\hat{y}_n$.
5. Upscale $\hat{y}_n$ with the *masked upscaling* approach (Sect. 2.4) to obtain $y_{n_{up}}$.
6. Concatenate $y_{n_{up}}$ with x_{n-1} and pass them through NN_{n-1}.
7. Add $y_{n_{up}}$ to the output of NN_{n-1} to obtain $\hat{y}_{n-1}$
8. Perform the last three steps subsequently with the rest of the networks.
9. Calculate the loss function.
10. Update the weights with a learning rate of 1×10^{-3}. This is considered to be one epoch.
11. Train the model until the loss stops improving.

Appendix B Training and Testing Data

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11242-021-01617-y.

Acknowledgements We gratefully recognize the Texas Advanced Computing Center, Los Alamos National Laboratory's Institutional Computing, and Los Alamos National Laboratory's Darwin cluster for their high performance computing resources. M. Pyrcz, J. Santos, and H. Jo acknowledge support from DIRECT Industry Affiliates Program (IAP), and M. Prodanović and J. Santos acknowledge support from Digital Rock Petrophysics IAP both of The University of Texas at Austin. J. Santos, Q. Kang, H. Viswanathan, and N. Lubbers were funded in part by the U.S. Department of Energy through Los Alamos National Laboratory's Laboratory Directed Research and Development program (LANL-LDRD). J. Santos would like to thank Alexander Hillsley for his assistance with the diagrams, Hasan Khan for providing the vuggy geometry, and Rafael Salazar-Tío for many useful discussions.

Funding J. Santos, Q. Kang, H. Viswanathan, and N. Lubbers were funded in part by the U.S. Department of Energy through Los Alamos National Laboratory's Laboratory Directed Research and Development program (LANL-LDRD).

Availability of data and material Will be made available through the digital rock portal.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Code and Data availability We carried-out the neural network work using PyTorch (Paszke et al. 2019) and the plots were generated using matplotlib (Hunter 2007). The code will be made available via the author's repository https://github.com/je-santos.

References

Alakeely, A., Horne, R.N.: Simulating the behavior of reservoirs with convolutional and recurrent neural networks. SPE Reservoir Eval. Eng. **23**(3), 992–1005 (2020). https://doi.org/10.2118/201193-PA

Alqahtani, N.J., Chung, T., Wang, Y.D., Armstrong, R.T., Swietojanski, P., Mostaghimi, P.: Flow-based characterization of digital rock images using deep learning. SPE J. (2021). https://doi.org/10.2118/205376-pa

Bachmat, Y., Bear, J.: On the concept and size of a representative elementary volume (REV). In: *Advances in transport phenomena in porous media*, pp. 3–20. Springer, Berlin (1987)

Barron, J.T.: Continuously differentiable exponential linear units. *arXiv* (3), 1–2 (2017)

Bear, J.: Dynamics of fluids in porous media. American Elsevier, Amsterdam (1972)

Berg, C.F.: Fontainebleau 3D models. http://www.digitalrocksportal.org/projects/57 (2016)

Bihani, A., Daigle, H., Santos, J.E., Landry, C., Prodanovic, M., Milliken, K.: MudrockNet: Semantic segmentation of mudrock SEM images through deep learning. 2 (2021). https://arxiv.org/abs/2102.03393

Bond, C.E., Kremer, Y., Johnson, G., Hicks, N., Lister, R., Jones, D.G., Haszeldine, R.S., Saunders, I., Gilfillan, S.M.V., Shipton, Z.K., Pearce, J.: International journal of greenhouse gas control the physical characteristics of a co 2 seeping fault : the implications of fracture permeability for carbon capture and storage integrity. Int. J. Greenhouse Gas Control **61**, 49–60 (2017). https://doi.org/10.1016/j.ijggc.2017.01.015

Carman, P.C.: Permeability of saturated sands, soils and clays. J. Agric. Sci. **29**, 262–273 (1939)

Carman, P.G.: Fluid flow through granular beds. Chem. Eng. Res. Des. **75**(1 Suppl), S32–S48 (1997). https://doi.org/10.1016/s0263-8762(97)80003-2

Cho, Y., Jun, H.: Estimation and uncertainty analysis of the CO2 storage volume in the sleipner field via 4D reversible-jump markov-chain Monte Carlo. J. Pet. Sci. Eng. **200**, 108333 (2020). https://doi.org/10.1016/j.petrol.2020.108333

Cho, Y., Gibson, R.L., Zhu, D.: Quasi 3D transdimensional Markov-chain Monte Carlo for seismic impedance inversion and uncertainty analysis. Interpretation **6**(3), T613–T624 (2018). https://doi.org/10.1190/INT-2017-0136.1

Chung, T., Da Wang, Y., Armstrong, R.T., Mostaghimi, P.: CNN-PFVS: integrating neural network and finite volume models to accelerate flow simulation on pore space images. Transp. Porous Media **135**(1), 25–37 (2020). https://doi.org/10.1007/s11242-020-01466-1

Cnudde, V., Boone, M.N.: High-resolution X-ray computed tomography in geosciences: a review of the current technology and applications. Earth-Sci. Rev. **123**, 1–17 (2013). https://doi.org/10.1016/j.earscirev.2013.04.003

Costanza-Robinson, M.S., Estabrook, B.D., Fouhey, D.F.: Representative elementary volume estimation for porosity, moisture saturation, and air-water interfacial areas in unsaturated porous media: Data quality implications. Water Resour. Res. **47**(7), 1–12 (2011). https://doi.org/10.1029/2010WR009655

Cunningham, K., Sukop, M.: Multiple Technologies Applied to Characterization of the Porosity and Permeability of the Biscayne Aquifer. Florida, Technical Report February (2011)

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee (2009)

D'Humieres, D., Ginzburg, I., Manfred, K., Pierre, L., Li-Shi, L.: Multiple-Relaxation-Time Lattice Boltzmann Models in 3D Multiple-relaxation-time lattice Boltzmann. NASA, AUGUST (2002)

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: RepVGG: Making VGG-style ConvNets Great Again. (2021). http://arxiv.org/abs/2101.03697

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. pp. 1–21, (2020). http://arxiv.org/abs/2010.11929

Santos, J.E., Jo, H.: A selection of synthetic fractures with varying roughness and mineralogy. http://www.digitalrocksportal.org/projects/198 (2019)

Finney, J., Prodanovicc, M.: Finney packing of spheres. (2016) http://www.digitalrocksportal.org/projects/47

Fukushima, K.: Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybern. **36**(4), 193–202 (1980). https://doi.org/10.1007/BF00344251

Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)

Guiltinan, E., Santos, J.E., Kang, Q.: Residual Saturation During Multiphase Displacement in Heterogeneous Fractures with Novel Deep Learning Prediction. In: *Unconventional Resources Technology Conference (URTeC)*, Austin (2020a). Society of Petroleum Engineers (SPE). https://doi.org/10.15530/urtec-2020-3048. URL https://www.onepetro.org/download/conference-paper/URTEC-2020-3048-MS?id=conference-paper%2FURTEC-2020-3048-MS

Guiltinan, E.J., Santos, J.E., Cardenas, M.B., Espinoza, D.N., Kang, Q.: Two-phase fluid flow properties of rough fractures with heterogeneous wettability: analysis with lattice Boltzmann simulations. Water Resour. Res. (2020). https://doi.org/10.1029/2020WR027943

He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. Proc. IEEE Int. Conf. Comput. Vis. (2015). https://doi.org/10.1109/ICCV.2015.123

He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, https://doi.org/10.1007/978-3-319-46493-0_38

Ho, J., Kalchbrenner, N., Weissenborn, D., Salimans, T.: Axial attention in multidimensional transformers. *arXiv*, pp. 1–11 (2019). ISSN 23318422

Holley, B., Faghri, A.: Permeability and effective pore radius measurements for heat pipe and fuel cell applications. Appl. Therm. Eng. **26**(4), 448–462 (2006). https://doi.org/10.1016/j.applthermaleng.2005.05.023

Huang, H., He, R., Sun, Z., Tan, T.: Wavelet-SRNet: A Wavelet-Based CNN for Multi-scale Face Super Resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1698–1706 (2017). ISSN 15505499. https://doi.org/10.1109/ICCV.2017.187

Hunter, J.D.: Matplotlib: A 2D graphics environment. Comput. Sci. Eng. **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., Carreira, J.: Perceiver: general perception with iterative attention (2021). http://arxiv.org/abs/2103.03206

Jenny, P., Lee, S.H., Tchelepi, H.A.: Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. J. Comput. Phys. **187**(1), 47–67 (2003)

Jha, D., Smedsrud, P.H., Riegler, M.A., Johansen, D., Lange, T.D.: ResUNet ++: an advanced architecture for medical image segmentation

Jo, H., Santos, J., Pyrcz, M.: Conditioning well data to rule-based lobe model by machine learning with a generative adversarial network. Energy Exp. Exp. (2020). https://doi.org/10.1177/0144598720937524

Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive Growing of GANs for Improved Quality, Stability, and Variation. pp. 1–26, 10 (2017). URL http://arxiv.org/abs/1710.10196

Khan, H.J., Prodanović, M., DiCarlo, D.A.: The effect of vuggy porosity on straining in porous media. SPE J. **24**(3), 1164–1178 (2019). https://doi.org/10.2118/194201-PA

Khan, H.J., DiCarlo, D., Prodanović, M.: The effect of vug distribution on particle straining in permeable media. J. Hydrol. (2020). https://doi.org/10.1016/j.jhydrol.2019.124306

Kozeny, J.: Ueber kapillare Leitung des Wassers im Boden. Akad. Wiss. Wien **136**, 271–306 (1927)

LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539

Li, H., Samet, H., Kadav, A., Durdanovic, I., Graf, H.P.: Pruning filters for efficient convnets. In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–13 (2017)

McClure, J.E., Prins, J.F., Miller, C.T.: A novel heterogeneous algorithm to simulate multiphase flow in porous media on multicore CPU-GPU systems. Comput. Phys. Commun. **185**(7), 1865–1874 (2014). https://doi.org/10.1016/j.cpc.2014.03.012

McClure, J.E., Li, Z., Berrill, M., Ramstad, T.: The LBPM software package for simulating multiphase flow on digital images of porous rocks. Comput. Geosci. **25**(3), 871–895 (2021). https://doi.org/10.1007/s10596-020-10028-9

Mohammadmoradi, P.: A multiscale sandy microstructure. (2017) http://www.digitalrocksportal.org/projects/92

Mohan, A.T., Lubbers, N., Livescu, D., Chertkov, M.: Embedding Hard Physical Constraints in Neural Network Coarse-Graining of 3D Turbulence. pp. 1–13, (2020). http://arxiv.org/abs/2002.00021

Molina, E., Arancibia, G., Sepúlveda, J., Roquer, T., Mery, D., Morata, D.: Digital rock approach to model the permeability in an artificially heated and fractured granodiorite from the liquiñe geothermal system (39° S). Rock Mech. Rock Eng. **53**(3), 1179–1204 (2020). https://doi.org/10.1007/s00603-019-01967-6

Mosser, L., Dubrule, O., Blunt, M.J.: Reconstruction of three-dimensional porous media using generative adversarial neural networks (2017). https://doi.org/10.1103/PhysRevE.96.043309.

Mosser, L., Dubrule, O., Blunt, M.J.: Stochastic reconstruction of an oolitic limestone by generative adversarial networks. pp. 1–22 (2017). http://arxiv.org/abs/1712.02854

Muljadi, B.P.: Bentheimer sandstone. http://www.digitalrocksportal.org/projects/11 (2015)

Ogilvie, S.R., Isakov, E., Glover, P.W.: Fluid flow through rough fractures in rocks. II: A new matching model for rough rock fractures. Earth Planet. Sci. Lett. **241**(3–4), 454–465 (2006). https://doi.org/10.1016/j.epsl.2005.11.041

Oron, A.P., Berkowitz, B.: Flow in rock fractures: The local cubic law assumption reexamined. Water Resour. Res. **34**(11), 2811–2825 (1998). https://doi.org/10.1029/98WR02285

Pan, C., Hilpert, M., Miller, C.T.: Lattice-Boltzmann simulation of two-phase flow in porous media. Water Resour. Res. **40**(1), 1–14 (2004). https://doi.org/10.1029/2003WR002120

Pan, W., Torres-Verdín, C., Pyrcz, M.J.: Stochastic Pix2pix: a new machine learning method for geophysical and well conditioning of rule-based channel reservoir models. Nat. Resour. Res. (2020). https://doi.org/10.1007/s11053-020-09778-1

Park, J., Iino, A., Datta-Gupta, A., Bi, J., Sankaran, S.: Novel hybrid fast marching method-based simulation workflow for rapid history matching and completion design optimization of hydraulically fractured shale wells. J. Pet. Sci. Eng. **196**, 107718 (2020). https://doi.org/10.1016/j.petrol.2020.107718

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. (NeurIPS) (2019). http://arxiv.org/abs/1912.01703

Prodanovic, M., Bryant, S.: Physics-driven interface modeling for drainage and imbibition in fractures. SPE J. **14**(3), 11–14 (2009). https://doi.org/10.2118/110448-PA

Prodanovic, M., Esteva, M., Hanlon, M., Nanda, G., Agarwal, P.: Digital Rocks Portal: a repository for porous media images

Prodanovic, M., Bryant, S.L., Karpyn, Z.T.: Investigating matrix/fracture transfer via a level set method for drainage and imbibition. pp. 125–136 (2010)

Santos, J., Prodanovic, M., Landry, C., Jo, H.: Determining the impact of mineralogy composition for multiphase flow through hydraulically induced fractures. In: *SPE/AAPG/SEG Unconventional Resources Technology Conference 2018, URTC 2018* (2018). https://doi.org/10.15530/urtec-2018-2902986

Santos, J.E., Prodanović, M., Pyrcz, M.: Characterizing effective flow units in a multiscale porous medium. Am. Geophys. Union Fall Meet. Abstr. (2018b). https://doi.org/10.1002/essoar.10502121.1

Santos, J.E., Mehana, M., Wu, H., Prodanović, M., Kang, Q., Lubbers, N., Viswanathan, H., Pyrcz, M.J.: Modeling nanoconfinement effects using active learning. J. Phys. Chem. C (2020). https://doi.org/10.1021/acs.jpcc.0c07427

Santos, J.E., Xu, D., Jo, H., Landry, C.J., Prodanović, M., Pyrcz, M.J.: PoreFlow-Net: A 3D convolutional neural network to predict fluid flow through porous media. Adv. Water Resour. **138**, 103539 (2020). https://doi.org/10.1016/j.advwatres.2020.103539

Saxena, N., Hofmann, R., Alpak, F.O., Berg, S., Dietderich, J., Agarwal, U., Tandon, K., Hunter, S., Freeman, J., Wilson, O.B.: References and benchmarks for pore-scale flow simulated using micro-CT images of porous media and digital rocks. Adv. Water Resour. **109**, 211–235 (2017). https://doi.org/10.1016/j.advwatres.2017.09.007

Schepp, L.L., Ahrens, B., Balcewicz, M., Duda, M., Nehler, M., Osorno, M., Uribe, D., Steeb, H., Nigon, B., Stöckhert, F., Swanson, D.A., Siegert, M., Gurris, M., Saenger, E.H.: Digital rock physics and laboratory considerations on a high-porosity volcanic rock. Sci. Rep. **10**(1), 1–16 (2020). https://doi.org/10.1038/s41598-020-62741-1

Shaham, T.R., Dekel, T., Michaeli, T.: SinGAN: learning a generative model from a single natural image (2019). URL http://arxiv.org/abs/1905.01164

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In; *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1874–1883 (2016). https://doi.org/10.1109/CVPR.2016.207

Sun, H., Vega, S., Tao, G.: Journal of petroleum science and engineering analysis of heterogeneity and permeability anisotropy in carbonate rock samples using digital rock physics. J. Pet. Sci. Eng. **156**, 419–429 (2017). https://doi.org/10.1016/j.petrol.2017.06.002

Tanaka, H.: Pruning neural networks without any data by iteratively conserving synaptic flow. (NeurIPS) (2020)

Tartakovsky, A., Meakin, P.: Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* 72(2), 1–9 (2005). https://doi.org/10.1103/PhysRevE.72.026301

Tokan-Lawal, A., Prodanovic, M., Eichhubl, P.: Investigating flow properties of partially cemented fractures in Travis Peak Formation using image-based pore-scale modeling. J. Geophys. Res. B Solid Earth **120**(8), 5453–5466 (2015). https://doi.org/10.1002/2015JB012045

Torquato, S.: Predicting transport characteristics of hyperuniform porous media via rigorous microstructure-property relations. Adv. Water Resour. **140**, 103565 (2020). https://doi.org/10.1016/j.advwatres.2020.103565

Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: the missing ingredient for fast stylization. (2016) URL http://arxiv.org/abs/1607.08022

van Oort, C.M., Duo, X.U., Offner, S.S., Gutermuth, R.A.: Casi: a convolutional neural network approach for shell identification. 880(2), 83 (2019). https://doi.org/10.3847/1538-4357/ab275e

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Process. Syst. **6**, 5999–6009 (2017)

Wang, Y.D., Blunt, M.J., Armstrong, R.T., Mostaghimi, P.: Deep learning in pore scale imaging and modeling. Earth-Sci. Rev. **215**, 103555 (2021). https://doi.org/10.1016/j.earscirev.2021.103555

Wang, Y.D., Chung, T., Armstrong, R.T., Mostaghimi, P.: ML-LBM: predicting and accelerating steady state flow simulation in porous media with convolutional neural networks. Transp. Porous Media (2021). https://doi.org/10.1007/s11242-021-01590-6

White, J.A., Borja, R.I., Fredrich, J.T.: Calculating the effective permeability of sandstone with multi-scale lattice Boltzmann/finite element simulations. Acta Geotech. **1**(4), 195–209 (2006)

Wildenschild, D., Sheppard, A.P.: X-ray imaging and analysis techniques for quantifying pore-scale structure and processes in subsurface porous medium systems. Adv. Water Resour. **51**, 217–246 (2013). https://doi.org/10.1016/j.advwatres.2012.07.018

Xue, X., Yang, C., Park, J., Sharma, V.K., Datta-Gupta, A., King, M.J.: Reservoir and fracture-flow characterization using novel diagnostic plots. SPE J. **24**(3), 1248–1269 (2019). https://doi.org/10.2118/194017-PA

Yamanaka, J., Kuwashima, S., Kurita, T.: Fast and accurate image super resolution by deep CNN with skip connection and network in network. vol. 7 (2017). URL http://arxiv.org/abs/1707.05425

Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.-H.: Deep learning for single image super-resolution: a brief review. **8**, (2018)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Javier E. Santos[1] · **Ying Yin**[2] · **Honggeun Jo**[1] · **Wen Pan**[1] · **Qinjun Kang**[3] · **Hari S. Viswanathan**[3] · **Maša Prodanović**[1] · **Michael J. Pyrcz**[1] · **Nicholas Lubbers**[3]

[1] The University of Texas, Austin, USA

[2] Xi'an Jiaotong University, Xi'an, China

[3] Los Alamos National Laboratory, Los Alamos, NM, USA