Reconstructing Pore Networks Using Generative Adversarial Networks

Kelly Guan (kmguan@stanford.edu)*

I. INTRODUCTION

Understanding fluid flow in porous media at the microscale is relevant to many fields, such as oil and gas recovery, geothermal energy, and geological CO_2 storage. Properties such as porosity and permeability are often calculated from laboratory measurements or direct imaging of the microstructure. However, due to acquisition times and experimental costs, it is difficult to evaluate the variability due to rock heterogeneity. Instead, researchers often use statistical methods to reconstruct porous media based on two-point or multi-point statistics [1], [2], [3]. Reconstruction using these methods often require knowledge about the pore and throat size distribution before and can be costly to generate multiple realizations of the same rock sample.

Recent advances in deep learning have shown promising use of generative adversarial networks (GANs) for rapid generation of 3D images with no a priori model [4], [5], [6]. In this project, we investigate the feasibility of using a Deep Convolutional GAN to generate 2D reconstructions of a binarized dataset of a sandstone sample. The accuracy of the reconstructed images are evaluated against real images using morphological properties such as porosity, perimeter, and Euler characteristic. We find that training using a log loss function versus the Wasserstein distance with a gradient penalty yields more accurate images.

A. Generative adversarial networks

GANs are made of two components: 1) a generator G that creates a synthetic training image and 2) a discriminator D that tries to differentiate between the synthetic and real training image. As the GAN is trained, the generator tries to create more realistic training images to "fool" the discriminator, while the discriminator tries to become better at classifying real (label = 1) vs. fake (label = 0) images. z is the latent space vector sampled from a normal distribution, so $G(z)$ maps the latent vector to the image space. x is the data from an image (real or fake), and $D(G(z))$ is the probability that the generated image is real. The two competing objectives result in the following value function,

$$\begin{aligned}\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] \\ + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(x))]\end{aligned} \quad (1)$$

Researchers have shown that using a deep convolutional network in the generator and discriminator models can improve synthetic image generation and training of the network. The overall guidelines for training a DCGAN usually involes: 1) using strided convolutions instead of pooling so that the network can learn its own pooling functions, 2) using batch normalization to improve gradient flow, 3) removing fully connected hidden layers, and 4) using a specific set of activation functions in the generator and discriminator (explained further in the methods) [7].

*Energy Resources Engineering, Stanford University

II. OBJECTIVE AND SCOPE

The main objective of this project is to investigate the accuracy and feasibility of generating 2D/3D sandstone images through training a DCGAN model. While this has been already studied in the literature, it is a relatively new field with many ongoing areas of interest, such as ways to improve training stability, image quality, and incorporating grayscale and multiscale images [5], [6].

This project first aims to successfully create and train a 2D GAN before eventually training a 3D GAN. We can then evaluate how modifying the GAN architecture affects the loss and accuracy of the generated images. Once trained, these images would then be able to be used as inputs into digital rock physics calculations of properties such as permeability and capillary pressure. Understanding how permeability is affected by variations in porosity and connectivity is necessary in many research areas involving fluid flow through porous media.

III. DATASET

The dataset was obtained from micro-x-ray tomography scans of a Bentheimer sandstone. The original data is a greyscale representation of the 3D solid, and has been processed and binarized into the pore and solid phases. The full dataset is 512^3 voxels large with a voxel size of $3.06\,\mu m$. In order for the training image (64 x 64 voxels) to capture an adequate area, the image was downsampled to 256^3 voxels with a voxel size of $6.12\,\mu m$. For data augmentation, subvolumes were extracted every 16 voxels to yield $36,864$ training images. Initial tests were also done on a larger dataset of $72,728$ images and yielded comparable results. To reduce training time, we used the smaller training set for the majority of our tests.

IV. NETWORK ARCHITECTURE

We used a deep convolutional GAN (DCGAN), which uses convolutional-transpose layers in the generator and convolutional layers in the discriminator, as our network model [7]. The model is based on the tutorial from [8], with modifications to allow for single-channel input images and one-sided label smoothing. Figure 1 shows the general architecture of a DCGAN, and the generator and discriminator architectures are shown in Table I.

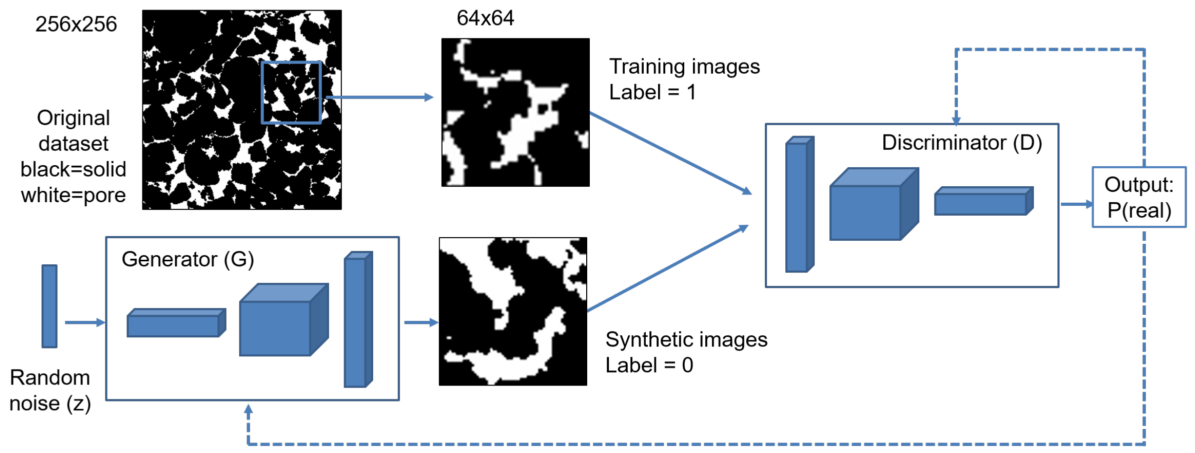
256x256
Original dataset black=solid white=pore
64x64
Training images Label = 1
Discriminator (D)
Output: P(real)
Generator (G)
Random noise (z)
Synthetic images Label = 0

Fig. 1. High level architecture of DCGAN

TABLE I
GENERATOR AND DISCRIMINATOR ARCHITECTURE

Layer	**Type**	**Filters**	**Kernel**	**Stride**	**Padding**	**Batch Normalization**	**Activation**
Generator							
1	ConvTransp2D	512	4 x 4	1	0	Yes	ReLU
2	ConvTransp2D	256	4 x 4	2	1	Yes	ReLU
3	ConvTransp2D	128	4 x 4	2	1	Yes	ReLU
4	ConvTransp2D	64	4 x 4	2	1	No	Tanh
Discriminator							
1	Conv2D	64	4 x 4	2	1	No	LeakyReLU
2	Conv2D	128	4 x 4	2	1	Yes	LeakyReLU
3	Conv2D	256	4 x 4	2	1	Yes	LeakyReLU
4	Conv2D	512	4 x 4	1	0	No	Sigmoid

The architecture follows the architecture described in [8] with the number of channels modified to be 1 (binarized image) instead of 3 (RGB)

TABLE II
DATA PARAMETERS

Image size	64^3 voxels
Batch size	64
Size of z latent vector	100
Generator filters	64
Discriminator filters	64
Learning rate, α	2×10^{-5}
Momentum, β_1	0.5
Label smoothing, ϵ	0.2

Parameters used for training DCGAN-1. Label smoothing refers to replacing the class label of 1 by $1 - \epsilon$

The images are loaded and normalized between $[-1, 1]$ prior to training. Batch normalization is done on sets of mini-batches of real and fake images. The model weights are randomly initialized from a normal distribution with $\mu = 0$ and $\sigma = 0.2$. For DCGAN, two separate Adam optimizers are used to optimize D and G. A one-sided label smoothing was also applied to the true label (1) as it has shown to improve model stability [9]. Parameters such as the leraning rate and label smoothing were varied to investigate the effect on training stability and accuracy. Figure 4 shows the results of some trial runs to highlight the effect of changing different parameters. Table II shows the parameters that were used for the optimal network. The network was trained using a Nvidia GeForce GTX 1070 GPU for about 1 hour.

V. TRAINING: LOSS FUNCTION

To train D and G, we used two different loss functions. We first use the binary cross entropy loss function (model DCGAN-1),

$$\begin{aligned} \ell(x, y) &= L = \{l_1, ..., l_N\}^T, \\ l_n &= -[y_n \log x_n + (1 - y_n) \log(1 - x_n)] \end{aligned} \tag{2}$$

Training is performed in two steps: 1) train the discriminator to maximize

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(x))] \tag{3}$$

while keeping the generator fixed, and 2) train the generator to minimize

$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(x))] \tag{4}$$

while keeping the discriminator fixed. In practice, due to the effect of vanishing gradients, it is easier to maximize $\mathbb{E}_{z \sim p_z(z)} \log(D(G(z))$ instead.

Convergence is theoretically reached when the generator can generate a distribution $p_g(x)$ that equal to $p_{\text{data}}(x)$, which corresponds to a discriminator output of 0.5. Further details about the training steps can be found in [8].

We also investigated the effect of using the Wasserstein distance as the loss function instead (model DCGAN-2).

The primary advantages of using the Wasserstein loss are that it can prevent mode collapse in the generator and allow for better convergence. The Wasserstein distance measures the distance between two probability functions and the discriminator now becomes a "critic" that evaluates the Wasserstein distance between the real and synthetic images. The distance is calculated by enforcing a Lipschitz constraint on the critic's model, either through weight clipping or a gradient penalty [10]. In our model, we use a gradient penalty to enforce the Lipschitz constraint which results in the following value function,

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] + \lambda \mathbb{E}_x[(||\nabla_x D(x)||_2 - 1)^2] \tag{5}$$

Where λ is the gradient penalty coefficient and is set to 10 for our model.

VI. MORPHOLOGICAL EVALUATION METRICS

The objective of using a DCGAN with our dataset is to recreate a pore network image with similar morphological properties as the original porous media. To evaluate the accuracy of our model, we use a set of morphological descriptors known as Minkowski functionals. In 2D, there are 3 Minkowski functionals that describe the surface: area, perimeter, and Euler characteristic. The area is simply the percent of pixels that are labeled as pore, n_{pore} divided by the total number of pixels n_{total},

$$\text{Area} = \frac{n_{\text{pore}}}{n_{total}} \tag{6}$$

The perimeter can be calculated using the Crofton formula, which takes a set of lines with varying orientations and counts the number of intersections with the region of interest. In our case, we consider the intersections with pixels that are labeled as pores. This is also normalized against the total number of pixels,

$$P = \frac{n_{intersect}}{n_{total}} \tag{7}$$

Finally, the Euler characteristic in 2D describes the connectivity of the surface and is defined as the difference between the number of connected regions (solid) and the number of holes (pores).

$$\chi = \frac{n_{connected} - n_{holes}}{n_{total}} \tag{8}$$

A region with a negative Euler characteristic will have more holes than connected regions, which indicates low connectivity across the area. A region with a positive Euler characteristic will have more connected regions and therefore a high connectivity across the area, which can allow for fluid flow.

For evaluation, after the model is fully trained, we create 64 realizations of 100^2 pixel images using the generator and randomly select the same number of images from our training set. The generator outputs images with values between $[-1, 1]$. The images are normalized, filtered using a median filter with a neighborhood of 2 pixels and binarized using Otsu's method. An example of the final synthetic image used for evaluation is shown in Figure 2. The Minkowski functionals were calculated using the MorphoLibJ and BoneJ plug-ins in ImageJ [11], [12].

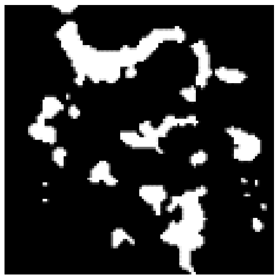

Fig. 2. Synthetic 100^2 pixel image after processing (white = pore, black = solid)

VII. EXPERIMENTAL RESULTS

Figure 3 shows the loss function of the generator and discriminator vs. number of iterations for our DCGAN-1 model (using the log loss function). We see that loss function initially increases, which corresponds to random noise images. After about 4000 iterations, the generator loss function drops and the image structure begins to resolve itself. This behavior is not unexpected and has been observed in training 3D GANs on similar datasets [5]. Further training shows that the image begins to resolve itself at $15,000$ iterations and further refinement occurs at $35,000$ iterations. We do not observe any mode collapse as the output images are all different from one another.

We next show the effect of varying different model parameters during the training process. Figure 4a shows the effect of changing the generator activation functions to leaky ReLU rather than ReLU. The resulting image quality is lower, which is expected given previous research results [13]. Figure 4b shows the result when using the Wasserstein distance (DCGAN-2) as the loss function instead of the log loss. Figure 5 shows the corresponding loss function of the discriminator while training our DCGAN-2 model.

TABLE III
EVALUATION RESULTS FOR DCGAN MODELS

Model	**Area**	**Perimeter**, $\times 10^{-2}$	**Euler char.**, $\chi \times 10^{-4}$
Train set	0.220	6.94	-3.89
DCGAN-1	0.217	6.82	-4.28
DCGAN-2	0.268	42.26	-88

Table III shows the morphological evaluation metrics used on our training set and two DCGAN models. We see that DCGAN-1 produces images with similar Minkowski functional values as the training set, while DCGAN-2 performs significantly worse. While we expected the Wasserstein loss

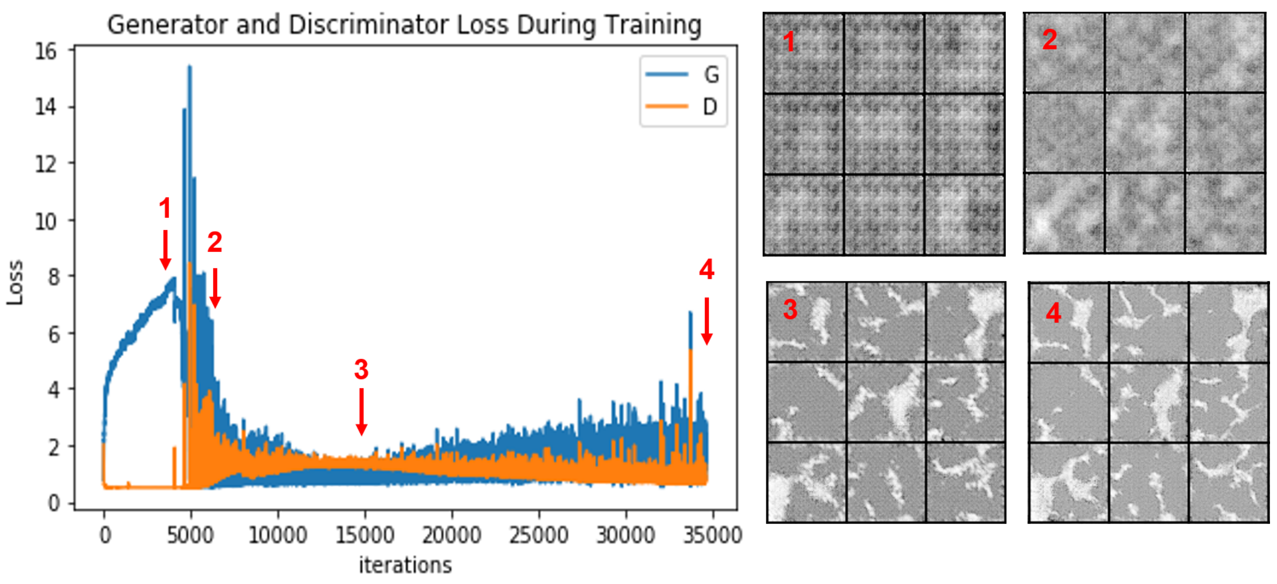

Fig. 3. Discriminator and generator loss function over 60 epochs for 36 , 864 training images. Numbers correspond to sequence images shown to the right

would improve training stability, the loss function and metrics indicate otherwise. One explanation is that the training time was not long enough, since the final images appear rather noisy. Another possibility is that the generator and discriminator architecture may need further modification, as previous papers have shown that batch normalization with the Wasserstein loss should be avoided [10].

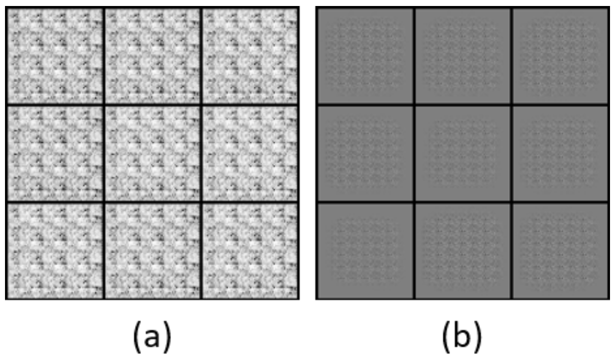

Fig. 4. Effect of changing (a) generator activation function from ReLU to LeakyReLU (b) loss function from log loss to Wasserstein distance

VIII. CONCLUSION AND FUTURE WORK

We were able to successfully implement and train a 2D DCGAN model to generate reasonable images with similar morphological properties to the original rock sample. However, it is still unclear if the generator is accurately capturing the underlying probability distribution of the real data. Further investigation could involve using the Wasserstein loss, as it is a measurement of the distance between two probability distributions. While our model using the Wasserstein loss did not perform as well, there have been extensive studies on ways to improve GANs and DCGANs and only some of the suggestions have been implemented here [9], [14], [13].

The ultimate goal of this work is to create a 3D GAN to create 3D pore networks for use in digital rock physics. The major challenge when scaling from 2D to 3D is expected to be in the computational train required to train the 3D network. Therefore, it is still important to understand the underlying architecture in 2D and knowledge gained from this project will be invaluable when constructing the 3D model.

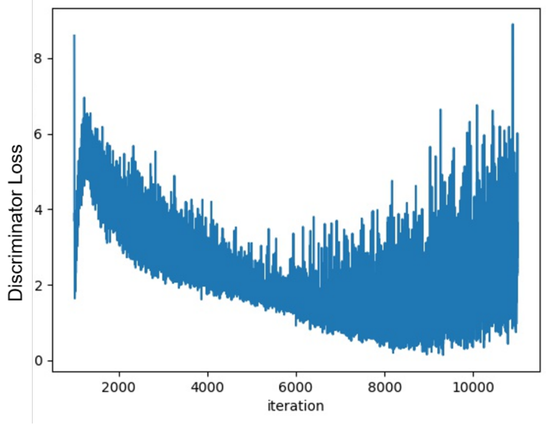

Fig. 5. Discriminator loss while training DCGAN-2

IX. ACKNOWLEDGEMENTS

Thanks to Tim Anderson and Prof. Anthony Kovscek for their guidance on this project. Part of this work was performed at the Stanford Nano Shared Facilities (SNSF), supported by the National Science Foundation under award

ECCS-1542152.

X. Project Code

Project code can be downloaded by clicking here (Stanford Google Drive).

References

[1] P.-E. Øren and S. Bakke, "Reconstruction of Berea sandstone and pore-scale modelling of wettability effects," *Journal of Petroleum Science and Engineering*, vol. 39, pp. 177–199, sep 2003.

[2] Y. Wang, C. H. Arns, S. S. Rahman, and J. Y. Arns, "Porous Structure Reconstruction Using Convolutional Neural Networks," *Mathematical Geosciences*, pp. 1–19, 2018.

[3] H. Okabe and M. J. Blunt, "Pore space reconstruction of vuggy carbonates using microtomography and multiple-point statistics," *Water Resources Research*, vol. 43, dec 2007.

[4] I. J. Goodfellow, J. Pouget-abadie, M. Mirza, B. Xu, and D. Warde-farley, "Generative-Adversarial-Nets," *Nips*, pp. 1–9, 2014.

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

[6] L. Mosser, O. Dubrule, and M. J. Blunt, "Stochastic Reconstruction of an Oolitic Limestone by Generative Adversarial Networks," *Transport in Porous Media*, vol. 125, no. 1, pp. 81–103, 2018.

[7] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks," p. 16, 2016.

[8] N. Inkawhich, "DCGAN Tutorial."

[9] T. Salimans, I. Goodfellow, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," pp. 1–10.

[10] I. Gulrajani, "Improved Training of Wasserstein GANs,"

[11] D. Legland, I. Arganda-Carreras, and P. Andrey, "MorphoLibJ: integrated library and plugins for mathematical morphology with ImageJ,"

[12] M. Doube, M. M. Kłosowski, I. Arganda-Carreras, F. P. Cordelières, R. P. Dougherty, J. S. Jackson, B. Schmid, J. R. Hutchinson, and S. J. Shefelbine, "BoneJ: Free and extensible bone image analysis in ImageJ," *Bone*, vol. 47, pp. 1076–1079, dec 2010.

[13] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, "How to Train a GAN? Tips and tricks to make GANs work."

[14] I. Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," 2016.