CS230

Reconstructing Pore Images Using Generative Adversarial Networks

Kelly Guan
Department of Energy Resources Engineering
Stanford University
kmguan@stanford.edu

Abstract

Estimating rock porosity and permeability is a vital component of reservoir engineering and is often done through laboratory measurements or direct imaging of the microstructure. Images are reconstructed from 2D or 3D datasets and used for modeling flow within the rock structure. Recent advances in deep learning have made using GANs a potential method for rapidly generating multiple realizations of a rock sample. In this project, we investigate how changing the loss function for a 2D DCGAN affects image quality and morphological parameters such as porosity, perimeter, and Euler characteristic. We also present progress towards training a 3D model to generate inputs for calculating permeability.

1 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 of these properties due to rock heterogeneity. Instead, researchers often use statistical methods to reconstruct porous media based on two-point or multi-point statistics [**?** **?**]. Reconstruction using these methods often requires 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 [1, 2, 3]. Last quarter, we created a 2D Deep Convolutional GAN to generate 2D reconstructions of a binarized dataset of a sandstone sample, as shown in Fig. 1 [4]. The accuracy of the reconstructed images were evaluated against real images using morphological properties such as porosity, perimeter, and Euler characteristic. This project aims to extend the work to create and train a 3D GAN to generate 3D reconstructions of a pore network and then evaluate the effect of varying hyperparameters or the network architecture.

While training the 3D was much more difficult than anticipated, we were able to investigate the effects of changing the loss function, specifically by adding a gradient penalty term, in the 2D case. Current 3D GAN techniques in the field have focused on DCGANs or conditional GANs that use higher resolution, 2D images as the conditional input [5, 2, 3]. Previous research has shown the potential of improving either training stability or image quality by modifying the loss function of the GAN [6].

CS230: Deep Learning, Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

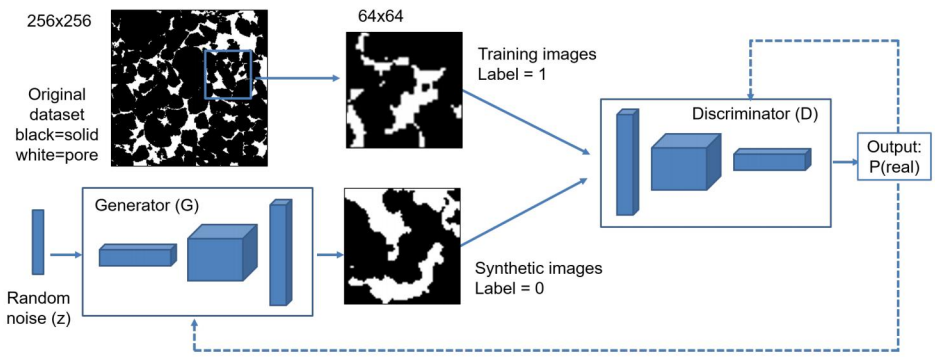

Figure 1: High level architecture of 2D DCGAN built previously. The original dataset is subsampled into multiple 64x64 sections for training.

2 Dataset and Features

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 $24,384$ training images. For the 3D case, a similar process was performed to extra $12,195$ training images with a size of 64 x $64x64$ voxels.

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 $\sigma = 0.2$. 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 [7]. Parameters such as the learning rate and label smoothing were varied to investigate the effect on training stability and accuracy. Table 4 shows the network architecture and Table 2 shows the parameters that were used for the 2D case. The 3D architecture is similar, with the convolutional and convolutional transpose layers in 3D instead. The number of discriminator filters was reduces from 64 to 16 for the 3D model in order to speed up training.

3 Methods

Table 1: Generator and discriminator architecture

Layer	Type	Filters	Kernel	Stride	Padding	Batch Norm	Activation
Generator							
1	ConvTransp3D	512	4 x 4 x 4	1	0	Yes	ReLU
2	ConvTransp3D	256	4 x 4 x 4	2	1	Yes	ReLU
3	ConvTransp3D	128	4 x 4 x 4	2	1	Yes	ReLU
4	ConvTransp3D	64	4 x 4 x 4	2	1	No	Tanh
Discriminator							
1	Conv3D	64	4 x 4 x 4	2	1	No	LeakyReLU
2	Conv3D	128	4 x 4 x 4	2	1	Yes	LeakyReLU
3	Conv3D	256	4 x 4 x 4	2	1	Yes	LeakyReLU
4	Conv3D	512	4 x 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)

To train D and G, we used two different loss functions. We first use the binary cross entropy loss function (model DCGAN-1). Training is performed in two steps: 1) train the discriminator to

Table 2: 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}
Label smoothing, ϵ	0.2

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

maximize

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

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

$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(x))] \tag{2}$$

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.

A gradient penalty term was also applied to the vanilla DCGAN, as it has been shown to improve image quality and prevent the discriminator from overfitting.

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))] + \lambda \mathbb{E}_x[(||\nabla_x D(x)||_2 - 1)^2] \tag{3}$$

We also investigate 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 [6]. In our model, we use a gradient penalty to enforce the Lipschitz constraint which results in the following value function,

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z \sim p_z(z)}[D(G(z))] + \lambda \mathbb{E}_x[(||\nabla_x D(x)||_2 - 1)^2] \tag{4}$$

4 Experiments/Results/Discussion

Model	**Pore area**	**Perimeter,** $\times 10^{-2}$	**Euler characteristic,** $\chi \times 10^{-4}$
Train set	18.4 ± 7.8	6.94	-3.89
DCGAN	21.6 ± 7.7	6.82	-4.28
DCGAN-GP	20.2 ± 8.1	6.87	-4.01

Gradient penalty norm	**Pore area,** %
1	20.2 ± 8.1
0.5	21.7 ± 7.5
0.25	20.3 ± 7.8

The first step was to modify the code for the 2D GAN to a 3D version. The main issues that occurred were 1) saving the data in a different format (hdf5 vs. png) and loading it appropriately and 2) memory issues when running the training loop.

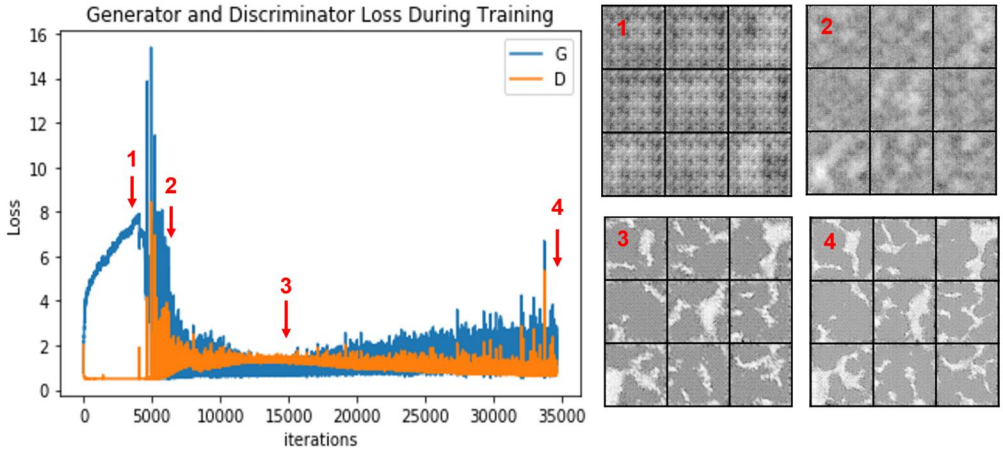

Figure 2: Discriminator and generator loss function over 60 epochs for $36,864$ training images. Numbers correspond to sequence images shown to the right

Possible solutions to the first problem is looking more closely at the documentation of working with hdf5 files, as there have been reported issues when using it with a dataloader with more than 1 worker. To progress with training, another smaller subset of .png formatted images are being created as this has been shown to work previously.

5 Conclusion/Future Work

For the second issue, one approach can be to reduce the batch size from 64 images to 32 or 16 when training. Or, downsampling the training image size from 64^3 voxels to 32^3. The major disadvantage of downsampling, however, is that important features of the image may be lost. Reducing the batch size will also affect training of the model, but previous work in 2D has shown that the model will still train properly, as shown in Figure **??**

Finally, it is important to establish a working baseline model. Previous results showed that using a log loss yielded visually and morphologically similar images from the generator. When training with a Wasserstein loss function, early attempts showed that the discrminiator loss (and generator loss, not shown), would blow up over training time, shown in Figure **??** and yield noisey, unrealistic images. Possible reasons are due to the binary nature of the input image, so adding Gaussian noise could improve training. The next steps to be done on this project are to 1) train the 3D model using a smaller batch size and 2) improve the baseline model using the Wasserstein distance.

6 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.

References

[1] Lukas Mosser, Olivier Dubrule, and Martin J. Blunt. Reconstruction of three-dimensional porous media using generative adversarial neural networks. *Physical Review E*, 96(4), 2017.

[2] Junxi Feng, Qizhi Teng, Xiaohai He, and Xiaohong Wu. Acta Materialia Accelerating multi-point statistics reconstruction method for porous media via deep learning. *Acta Materialia*, 159:296–308, 2018.

[3] Denis Volkhonskiy, Ekaterina Muravleva, Oleg Sudakov, Denis Orlov, Boris Belozerov, Evgeny Burnaev, and Dmitry Koroteev. Reconstruction of 3D Porous Media From 2D Slices. pages 1–15, 2019.

[4] Kelly Guan. Reconstructing pore networks using generative adversarial networks. http://cs229.stanford.edu/proj2018/report/222.pdf. 2018.

[5] Lukas Mosser, Olivier Dubrule, and Martin J. Blunt. Stochastic Reconstruction of an Oolitic Limestone by Generative Adversarial Networks. *Transport in Porous Media*, 125(1):81–103, 2018.

[6] Ishaan Gulrajani. Improved Training of Wasserstein GANs.

[7] Tim Salimans, Ian Goodfellow, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. pages 1–10.

[8] Nathan Inkawhich. DCGAN Tutorial.