Synthesis of Digital Microstructures using Generative Adversarial Networks

MM3014 UGRC Report

- Vir Karan (MM19B057)

1 INTRODUCTION [1], [2]

An overarching theme of materials research is the design of material microstructures so that the ensuing material exhibits tailored properties. The final property of the material system is intricately connected to the underlying microstructure. Microstructure-sensitive design has been used to tailor a wide variety of properties including strengths, heat and mass diffusivities, energy storage capacity and lifetime, and energy conversion efficiency. In microstructure-sensitive design, quantifying the effect of microstructure features on performance is critical for the efficient design of application-tailored devices. As a component of integrated computational materials engineering (ICME), methods to generate realistic simulation volumes are essential for modeling and simulating materials with complex microstructures. While obtaining microstructures experimentally guarantees realistic simulation volumes, the cost or difficulty of microstructural characterization often limits the size or number of microstructures that can be sampled. Thus, to support computational design and performance surveys, a common goal is to synthesize statistically representative sets of microstructural realizations. A number of successful approaches have been developed and are commonly used amongst the material science community, however, most of them require assumptions about the underlying structure; they use some form of optimization to refine an initial microstructure to satisfy target constraints and thus are often limited to either binary (two-phase) microstructures, uniform microstructures consisting of primitive/regular geometries, or a combination of both and are computationally very intensive, requiring anywhere between a few minutes to a few hours to even simulate one microstructural sample. To this end, the use of deep generative models such as Variational Auto-encoders and GANs have been applied to simulating representative microstructural data while simultaneously being computationally faster than traditional techniques. A well-trained generative model can (theoretically) synthesize an infinite number of unique microstructure realizations from the learned data distribution at relatively trivial speeds and compute costs. Several GAN architectures such as the Deep Convolutional GAN (DCGAN), Conditional GAN (cGAN), Wasserstein GAN (WGAN) and Auxiliary Classifier Wasserstein GAN (AC-WGAN) were implemented on a 2-phase microstructure dataset of varying morphology and a fixed phase fraction. The results were validated using visual inspection and statistical similarity methods.

2 METHODS AND MATERIALS

2.1 Neural Networks: An overview [3]

Artificial neural networks (ANNs) are inspired by biological neural networks in our brains. The fundamental computing unit of ANNs is a neuron, which takes multiple inputs, and outputs a possibly non-linear function (called the activation function) of the weighted sum of its inputs. Several activation functions are commonly used, such as sigmoid, linear, rectified linear unit (ReLU), leaky ReLU, etc. Figure 1 illustrates a fully-connected ANN, also known as multilayer perceptron (MLP), and the ReLU activation function. A deep learning network is an ANN with two or more hidden layers. The manner in which the neurons are connected amongst themselves determines the architecture of the network. The edges or interconnections between neurons have weights, which are learned during neural network training with the goal of making the ANN output as close as possible to the ground truth, which is technically referred to as minimizing the loss function. The training process involves making a forward pass of the input data through the ANN to get predictions, calculating the errors or loss, and subsequently back-propagating them through the network to update the weights of the interconnections via gradient descent in order to try to make the outputs more accurate. A single pass of the entire training data is called an epoch, and it is repeated iteratively till the weights converge. Usually when the data are large, the forward passes are done with small subsets of the training data (called mini-batches), so an epoch would comprise of multiple iterations of mini-batches. The inputs of a neural network are generally normalized to have zero mean

and unit standard deviation, and the same concept is sometimes applied to the input of hidden layers as well (called batch normalization) to improve the stability of ANNs.

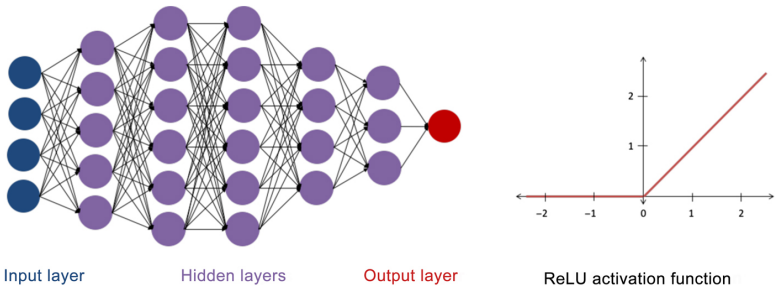


Figure 1 A fully-connected deep ANN with four inputs, one output, and five hidden layers with varying number of neurons (left). The ReLU activation function (right). Image credits: Cambridge University Press.

Convolutional neural networks (CNNs) are a class of artificial neural networks particularly well suited to image data and computer vision applications. In its forward mode, a CNN takes in an image and applies a series of signal processing steps (typically filter convolutions, rectification operations, and pooling or downsampling) to generate a compact representation of the visual information contained in the image, termed the feature vector. The feature vector can then be used for a variety of image processing tasks, including visual similarity and classification. The architecture of the CNN, including the type, number, order, and connectivity of the operations, is user-defined, and the training process attempts to optimize the many internal variables for best performance. CNNs can also operate in reverse, taking in a feature vector and transforming it into an image.

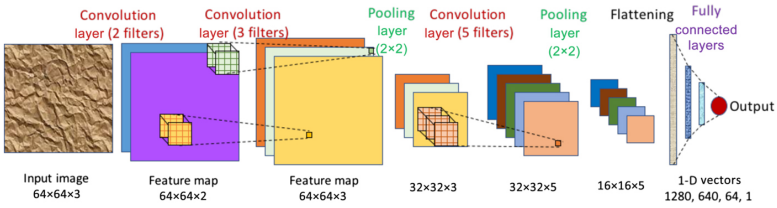


Figure 2 A CNN with three convolution layers, two pooling layers, and three fully connected layers. It takes a 64 × 64 RGB image (i.e., three channels) as input. The first convolution layer has two filters resulting in a feature map with two channels (depicted in purple and blue). The second convolution layer has three filters, thereby producing a feature map with three channels. It is then followed by a 2 × 2 pooling layer, which reduces the dimensionality of the feature map from 64 × 64 to 32 × 32. This is followed by another convolution layer of five filters, and another pooling layer to reduce feature map dimension to 16 × 16 (five channels). Next, the feature map is flattened to get a 1-D vector of 16 × 16 × 5 = 1280 values, which is fed into three fully connected layers of 640, 64, and one neuron(s) respectively, finally producing the output value. Image credits: Cambridge University Press.

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GAN) are powerful models that attempt to learn high-dimensional data distributions such as images. The key strategy is to pose the estimation of generative model as two-player min-max game, with each player being a trainable CNN model, referred as the generator and discriminator respectively. Both the networks are trained simultaneously in an adversarial manner. The generator is trained to generate realistic images by learning a nonlinear mapping from a low-dimensional space of latent parameters to the space of real images, while the discriminator (D) is trained to discriminate or classify the samples generated by the generator (G) as either 'real' or 'fake'. After sufficient training, the generator network is (often) able to reproduce synthetic images that closely resemble the original images. A more vivid analogy of GAN is given by Goodfellow et al.(4): in this adversary scenario, the generator can be thought of a group of counterfeiters who tries to produce fake currency, while the discriminator is

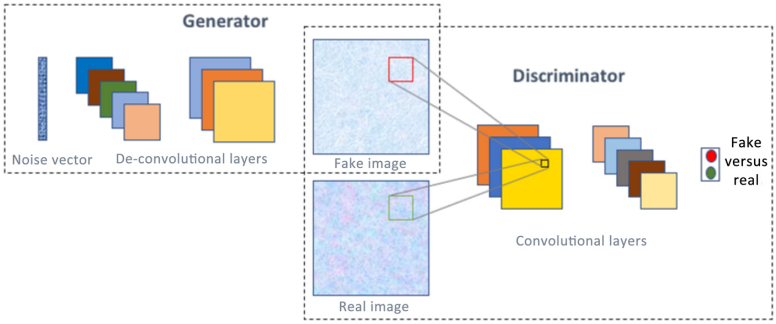


Figure 3 A GAN consists of two neural networks—generator and discriminator, and with proper training, is capable of generating realistic images/data from noise. Image credits: Cambridge University Press.

analogous to a team of police, trying to detect the counterfeit currency from the real money. Competitions in this adversary game would keep pushing both sides to the equilibrium in which the counterfeits are indistinguishable from the real money. The GAN framework proposed can produce visually appealing samples, but it usually suffers from training and mode collapse. The reason for the training difficulty, as explained in (5) is that the divergence is not continuous with respect to the parameters of the generator. A proposed improvement, known as Wasserstein GAN (WGAN), advocates using the Earth Mover Distance (also known as the Wasserstein-1 distance), which is continuous under reasonable assumptions. Moreover, the training can be stabilized via a suitable weight-clipping step in each epoch. Furthermore, by imposing additional structure into the GAN latent space, conditional generative models provide an efficient means to better control features in synthesized samples.

A Conditional GAN (6), by providing side information (e.g. class labels) to both generator and discriminator, proposes an implementation of this approach and subsequently improves visual quality and diversity of the synthesized images. A later work (7) introduces the auxiliary classifier GAN (AC-GAN) architecture which, in addition to using class labels for synthesizing class conditional image samples, also includes a classifier which predicts class labels for images.

As part of the project, various GAN architectures were tried, and based on the training stability and visual inspection of the results, the WGAN and AC-WGAN architectures were explored deeply. A schematic of the final architectures used is shown in Figure 4. Henceforth, the terms GAN and WGAN, Conditional GAN and AC-WGAN would be used interchangeably.

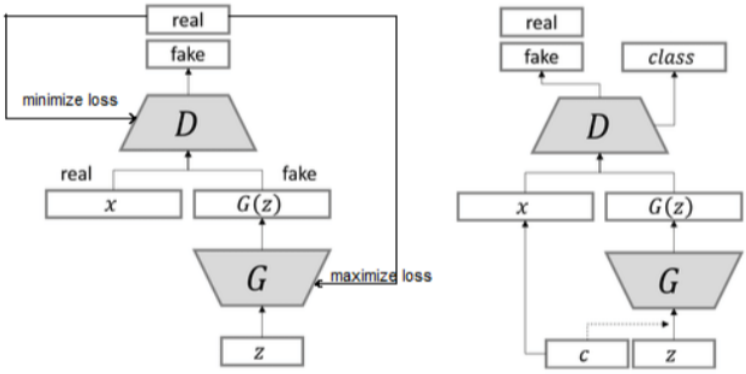


Figure 4 Architecture of a WGAN (left), AC-WGAN (right). z is the latent vector (random noise vector) that is passed into the discriminator (D), which then outputs a batch of fake images, $G(z)$. A batch of fake, $G(z)$, and real images, x, are sampled and passed through the Critic (D), the output of which is a score of the image which indicates the probability that the image is real. In the AC-WGAN, an additional class label is also passed into G with z, and D is made to predict the label. This adds structure into the latent space, and allows for conditional generation of images.

2.3 Training Data

The training data was of 2-phase microstructures simulated using a Gaussian distribution, and consisted of 7 classes of microstructural data, as shown in Figure 5. Each class had 50 samples, giving a total of 350 samples in the dataset. Each class has a different morphology, but all the samples had the same phase fraction of 0.5. Each image had dimensions of 256x256, with the phases being represented by 0s and 1s. Hence, each microstrucutral image was represented as a 256x256x1 array, having either 0 or 1 as it's entries.

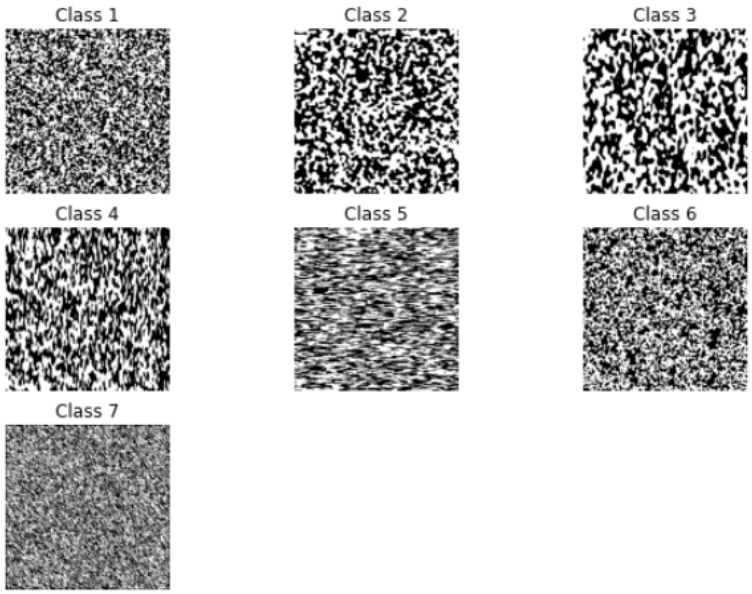


Figure 5 A sample from each class of the training dataset.

2.4 Training and hyper-parameters

The models were built and trained using the Keras API in the Tensorflow framework. The code was run using the GPUs provided via Google Colab. Model architectures were chosen using other GAN literature and implementations [(1), (2)] as reference. Hyper-parameter optimization was done using the help of a Bayesian Hyper-parameter optimization framework, and Structural Similarity Index (SSIM), discussed in greater detail in section 2.5.2, was the metric used to evaluate quality of the generated samples during training, along with manual visual inspection. Each combination of the hyper-parameters took around 120-180 minutes to train, and around 25-30 different combinations were tried for each model. The hyper-paramters used and their descriptions are given in Table 1.

Parameter	Description	Range
Lr	Learning Rate of the Optimizer	0.01 - 0.00001
Conv blocks	Number of Convolution blocks (or layers) in the critic model (D)	4 - 5
Deconv blocks	Number of Deconvolution blocks (or layers) in the generator model (G)	4 - 5
Nodes(D)	Number of nodes in the last layer of D	128 - 512
Nodes(G)	Number of nodes in the last layer of G	128 - 512
Epochs	Number of epochs models are trained for (training duration)	2000 - 6000
Batch size	Number of training samples in each batch	32 - 128
Training data size	Number of samples used for training	50/350/1050
Training Ratio	Number of times D is updated for every update of G	1 - 5

Table 1 List of hyper-parameters and their descriptions

2.5 Statistical similarity measures

2.5.1 Mean Squared Error (MSE)

$$MSE = \frac{1}{mn}\Sigma_{i=1}^{n}\Sigma_{j=1}^{m}(X[i,j]-Y[i,j])^2$$

Where, X and Y are two images represented as arrays of dimensions $m \times n$. A value of MSE close to 0 implies high similarity between X and Y.

2.5.2 Structure Similarity Index (SSIM)

For two images X and Y of the same dimensions,

$$SSIM(X,Y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Where, $\mu_x, \mu_y, \sigma_x, \sigma_y$ are the means and variances of the entries of X and Y respectively, σ_{xy} is the covariance of X and Y and c_1, c_2 are two arbitrary constants close to 0, whose values are determined from X and Y. A SSIM close to 1 implies X and Y are very similar and a SSIM close to 0 implies X and Y are very dissimilar.

2.5.3 Peak signal to Noise ratio (PSNR) and Signal to Noise ratio (SNR)

$$PSNR = 10\log_{10}\left(\frac{MAX_X^2}{MSE(X,Y)}\right)$$

$$SNR = 10\log_{10}\left(\frac{RMS_X^2}{MSE(X,Y)}\right)$$

Where, MAX_X is the maximum value of the pixels in the input images, which is equal to 1 for the microstrucutral data, RMS_X is the Root Mean Squared value of the input images and $MSE(X,Y)$ is the same as in Section 2.5.1. Here, "Signal" refers to the actual images that are input, and "Noise" refers to the difference in the images (given by the MSE). The higher the value of PSNR and SNR, the more similar the images are.

2.6 2-Point Correlations [8]

2-point spatial correlations (also known as 2-point statistics) contain information about the fractions of local states (or phases) as well as the first order information on how the different local states are distributed in the microstructure. 2-point statistics can be thought of as the probability of having a vector placed randomly in the microstructure and having one end of the vector be on one specified local state and the other end on another specified local state. This vector could have any length or orientation that the discrete microstructure allows. The equation for 2-point statistics:

$$f[r|l,l'] = \frac{1}{S}\sum_{s} m[s,l]m[s+r,l']$$

In this equation, $f[l]$ is the probability of finding the local state l in any randomly selected spatial bin in the microstructure, $m[s,l]$ is the microstructure function (the digital representation of the microstructure), S is the total number of spatial bins in the microstructure, s refers to a specific spatial bin, $f[rl,l']$ is the conditional probability of finding the local states l and l' at a distance and orientation away from each other defined by the vector r. When the 2 local states are the same, $l = l'$, it is referred to as a autocorrelation. If the 2 local states are not the same it is referred to as a crosscorrelation.

3 RESULTS AND DISCUSSION

3.1 WGAN (with data augmentation)

3.1.1 Model Architecture and Hyperparameters

D consists of 5 Convolution blocks (Convolution layer + Dropout layer (0.25) + Batch normalization layer + LeakyReLu activation). These are in turn connected to two fully connected layers with a sigmoid activation and an output layer which gives the realness of the input image. G consists of an input layer to take in the latent vector which is connected to 4 Deconvolution blocks (Convolution layer + Batch normalization layer + ReLu activation + Upsampling layer) which is connected to a Convolution layer

with Tanh activation. Both models use an Adam optimizer with learning rate of 0.00024 and a momentum of 0.5. The generator is an approximate mirror image of the discriminator. Furthermore, the discriminator is updated twice before an update of the generator.
Training was done using only the samples from Class 1 of the dataset, along with data augmentation techniques (flipping and rotation of the images), to give a training set with 1050 images. Hyper-parameters used are given in Table 2.

Parameter	Value
Lr	0.00024
Conv blocks	5
Deconv blocks	4
Nodes(D)	371
Nodes(G)	482
Epochs	4000
Batch size	128
Training data size	1050
Training Ratio	2

Table 2 List of hyper-parameters used for WGAN (with data augmentation)

3.1.2 Results

The microstructure generated by the WGAN is shown in Figure 6.

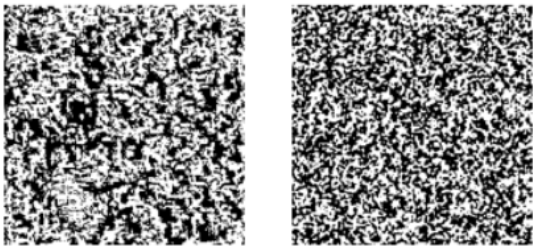


Figure 6 Left: Generated Image, Right: Ground truth

The 2-point correlations (auto correlations) are also found for these images and are shown in Figure 7. A statistical comparision of both, the images as well as the autocorrelations, was done and the results are tabulated in Tables 3 and 4 respectively. A local SSIM map for the images in Figures 6 and 7 is shown in Figure 8a and 8b respectively.

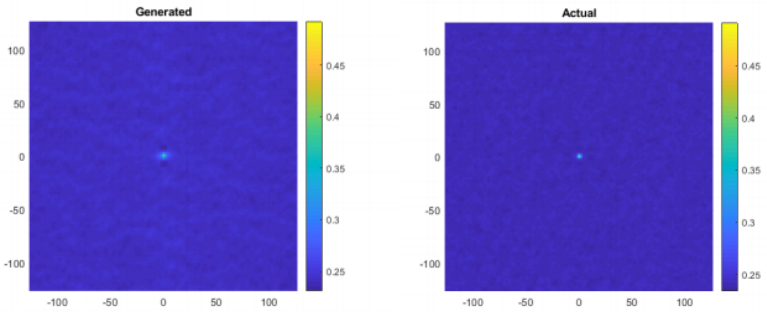


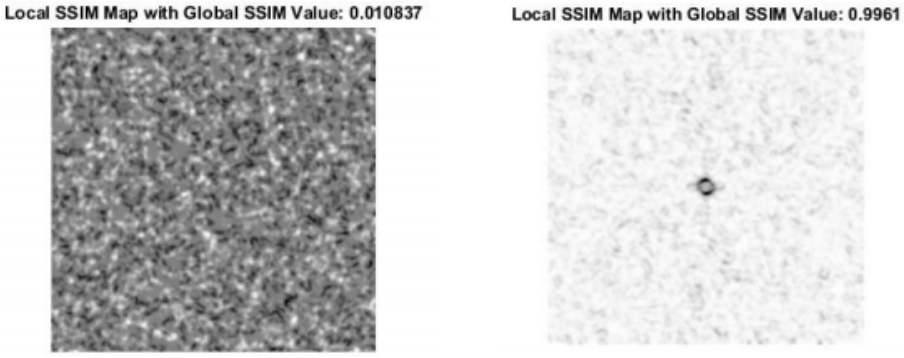
Figure 7 Left: Auto correlation map of the generated Image, Right: Auto correlation map of ground truth; Autocorrelations ($f[rl,l']$) are found as given in Section 2.6, and a map of $f[rl,l']$ is plotted as a function of x and y coordinates in the figures given above.

Metric	Value
SSIM	0.01083
PSNR	2.9718
SNR	-0.1150
MSE	0.5045

Table 3 Statistical comparison of the two images in Figure 6

Metric	Value
SSIM	0.9961
PSNR	50.2003
SNR	37.8536
MSE	9.84e-6

Table 4 Statistical comparison of the two auto correlations in Figure 7



(a) SSIM map for images in Figure 6

(b) SSIM map for images in Figure 7

Figure 8 SSIM map is calculated over small sections of the images, and plotted for the full images. Black regions in the Figures correspond to regions of dissimilarity, and white regions correspond to regions of high similarity between the images. From the figures, it can be seen that the images given in Figure 6 have low similarity, and the regions of dissimilarity are distributed throughout the images. Meanwhile, the auto correlations of the same images (shown in Figure 7) have a high similarity, and regions of dissimilarity are concentrated only at the centre.

3.2 WGAN (without data augmentation)

3.2.1 Model Architecture and Hyperparamters

Model architecture is the same as 3.1.1, but with a different set of hyperparameters, given in Table 5. Training was done using the samples from Class 1 of the dataset, without any augmentation. Hence, training set consisted of only 50 samples.

Parameter	Value
Lr	0.000074
Conv blocks	4
Deconv blocks	4
Nodes(D)	371
Nodes(G)	482
Epochs	4000
Batch size	128
Training data size	50
Training Ratio	4

Table 5 List of hyper-parameters used for WGAN (without data augmentation)

3.2.2 Results

The microstructure generated by the WGAN is shown in Figure 9. The results of the statistical comparison of the images as well as the auto correlations is tabulated in Table 6.

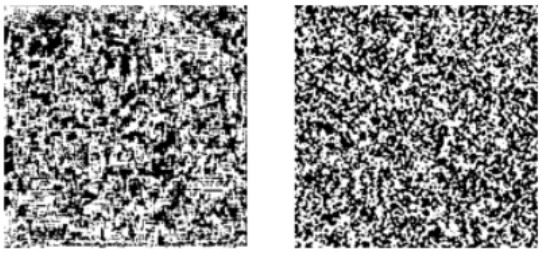


Figure 9 Left: Generated Image, Right: Ground truth

Metric	Value (images)	Value (auto correlations)
SSIM	0.0137	0.9776
PSNR	N/A	25.8653
SNR	N/A	13.5187
MSE	0.4970	0.0026

Table 6 Statistical comparison of the two images in Figure 9 and their auto correlations

3.3 Analysis of images generated by WGAN

The images generated (Figures 6 and 9) seem visually similar to the ground truth data. However, the SSIM values for these images is quite low (around 0.01), which seems to suggest otherwise. In most image proccessing applications, SSIM is a reliable metric for measuring image similarity, however, as mentioned in the paper(9), SSIM fails when working with images simulated using a Gaussian distribution. This is quite problematic as the data used for training the model, as well as the latent vector (z) that is passed as input into G, are both drawn from a Gaussian (Normal) distribution. Hence, SSIM is not a very reliable metric when working with raw microstructural image data.
The 2-point correlations (autocorrelations) of the microsturcture, unlike the raw image, only contains information about the distribution of the phases (or states), among other morphological features, and not the underlying distribution used to simulate the phases themselves. Hence, to resolve this issue of the SSIM, the autocorrelations of the data were found, and the statistical metrics were used to quantify them instead, which led to a better judgement of the similarity of the generated data to the actual data.
As can be seen through Figure 7 and 8 and the data in Table 4, the autocorrelations of the generated and actual images are quite similar, meaning that the underlying distribution of the microstrucutral features in the generated data is statistically similar to the distribution of the training data, which was the goal of using GANs. Furthermore, the 1-point correlations (phase fractions) of the generated microstrucutres lie in the range of $(0.48, 0.53)$, which is consistent with the training data, which had phase fractions in the range of (0.49, 0.51).
Another issue faced was that unlike the training data, the images generated by the GAN were not strictly consisting of 0s and 1s, i.e, the image was grayscale rather than pure black and white. To perform any microstructural analysis of the data, thresholding was done on the images, as shown in Figure 10. This was achieved using automatic thresholding functions from the skimage library. The slight difference in the distribution of the phase fractions of the generated data as compared to the actual data can be attributed to the use of the thresholding process.

On comparing the images generated by the models in Section 3.1 and 3.2 (Figures 6 and 9 respectively), it can be clearly seen that the use of data augmentation techniques (flipping and rotation by 90^0), to increase the size of the training set, leads to better generated microstructures, both visually and statistically speaking. This can be attributed to the fact that 50 training samples might not be sufficient for training complex, data-hungry networks such as GANs. On the downside, use of data augmentation techniques is likely to inhibit the GAN from learning microstructurally anisotropic features, which makes the resultant model incapable of generating microstrucutres that are completely statistically equivalent to the training data.

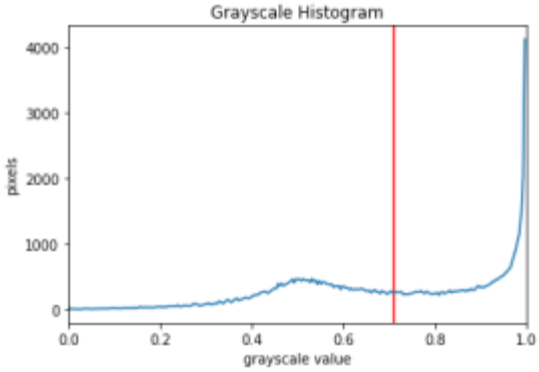


Figure 10 Histogram of a generated sample; Red line corresponds to the threshold value selected (minima of histogram curve on left), all pixels lying to the right of this line are set to 1, and all pixels on the left are set to 0, to generate true two-phase microstructural data.

3.4 AC-WGAN (Conditional GAN)

3.4.1 The Need for Conditional Generation

Training a model using different type (or classes) of microstructural data (shown in Figure 5), it is possible to provide a structure to the latent space (z), using which it is possible to control the type of features that the GAN ends up generating. As per most popular GAN literature, use of class labels also leads to better training stability and an improvement in the quality of the generated data. To do this, an Auxiliary Classifier WGAN was trained on all of the training data (50 samples per class, 350 samples in total).

3.4.2 Model Architecture and Hyperparamters

D consists of 5 Convolution blocks (Convolution layer + Dropout layer (0.25) + Batch normalization layer + LeakyReLu activation). These are in turn connected to two fully connected layers with a sigmoid activation and two output layers, one gives the realness of the input image, and the other predicts the class to which the image belongs. G consists of an input layer to take in the latent vector, along with an embedding layer that takes in the class label as input, which are then concatenated and connected to 4 Deconvolution blocks (Convolution layer + Batch normalization layer + ReLu activation + Upsampling layer) which is connected to a Convolution layer with Tanh activation. Both models use an Adam optimizer with learning rate of 0.00012 and a momentum of 0.5. The generator is an approximate mirror image of the discriminator. Furthermore, the discriminator is updated five times before an update of the generator. The hyperparameters used are given in Table 7.

Parameter	Value
Lr	0.00012
Conv blocks	5
Deconv blocks	4
Nodes(D)	232
Nodes(G)	377
Epochs	4000
Batch size	64
Training data size	350
Training Ratio	5

Table 7 List of hyper-parameters used for AC-WGAN

3.4.3 Results

The microstructures generated by the model are shown in Figure 11. A smaller sized version of these, along with the corresponding microstructures from the training data are shown in Figure 12. The results

of the statistical comparison of both, the generated and actual images as well as their auto correlations, are given in Tables 8 and 9 respectively.

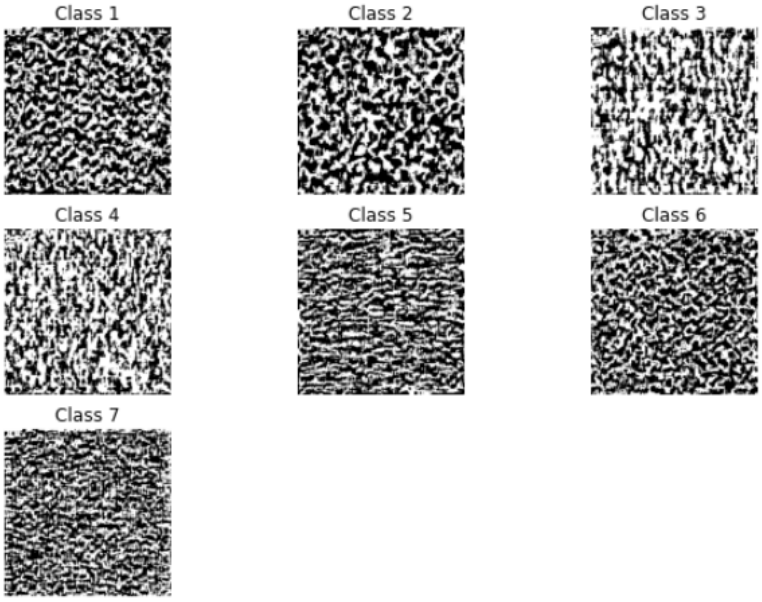


Figure 11 Generated images corresponding to each class of the training data

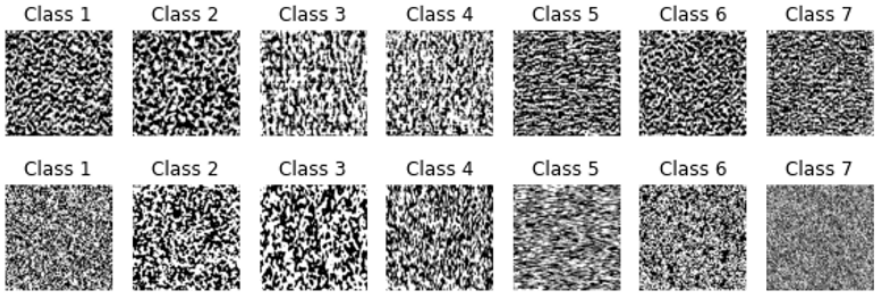


Figure 12 Top: Generated Images, Bottom: Ground Truth

Metric	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
SSIM	0.0031	0.0021	0.0259	0.0268	0.011	0.0024	0.0122
MSE	0.5053	0.5197	0.4867	0.4865	0.5188	0.5051	0.4995

Table 8 Statistical comparison of the two sets of images in Figure 11

3.5 Analysis of the images generated by AC-WGAN

Images from classes 2, 3, 4 and 5 (to certain extent) seem to be visually quite similar to the actual microstructural samples, while there is room for improvement in classes 1 and 6, and class 7 seems to be very off visually from the microstructural data. All generated samples had a phase fraction in the interval of (0.47, 0.54), while the actual data had phase fractions in the range of (0.49, 0.51). The SSIM and MSE of the images with the ground truth images was poor overall, however, using the arguement provided in Section 3.3, performing a comparison of the auto correlations gave much higher values for these metrics, showing that the inherent distribution of microstructural features is very similar between the generated samples and the ground truth data. From the statistical comparison, autocorrelations of the generated

Metric	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
SSIM	0.9637	0.9707	0.9562	0.9432	0.9151	0.9521	0.9655
PSNR	25.3562	26.6129	21.8641	20.8583	21.1777	24.0633	25.4180
SNR	13.0096	14.1922	10.2747	8.5695	9.6289	11.8012	13.2225
MSE	0.0029	0.0022	0.0065	0.0082	0.0076	0.0039	0.0029

Table 9 Statistical comparison of the auto correlations of the two sets of images in Figure 11

samples across all the classes seem to be quantitatively similar to the microstructural data, with PSNR and SNR being the highest of the lot for class 2, followed by classes 7, 1 and 6, following the opposite trend as the visual similarity, as classes 3 and 4 rank lower. The SSIM was > 0.9 for all the classes, implying that the model is capable of capturing the inherent distribution of the microstrucutral features quite well. However, the model seems to be struggling to generate microstructures with very small grain size, as is apparent by microstructures generated for class 7. Further, it also fails to capture features such as the diagonal axis along which grains have been distributed in class 7. Furthermore, the statistical similarity of the images generated by the Conditional GAN is less than that of the images generated by the (unconditional) WGAN, that was trained only on one class of microstructures.

3.6 Problems Faced and Possible Improvements

The lack of a single reliable metric to measure the quality of the generated microstuctural images has led to the need of manual visual inspection of the generated samples during training and hyperparamter optimization, especially for the conditional GAN. This is however, a problem faced by the GAN community in general. If a suitable metric is found, the process of hyperparameter optimization can be automated, which could lead to better quality of the generated microstructures. To improve the stability of the training process, Spectral Normalization can be used in place of Batch Normalization in D. Further, to improve quality of the generated samples, more training data can be acquired and used, and better hyperparameter optimization can be done. All these could not be included in this report due to the lack of time and computing resources at the time of writing this report.

REFERENCES

[1] R. Singh, V. Shah, B. Pokuri, S. Sarkar, B. Ganapathysubramanian, and C. Hegde, "Physics-aware deep generative models for creating synthetic microstructures," *arXiv preprint arXiv:1811.09669*, 2018.

[2] T. Hsu, W. K. Epting, H. Kim, H. W. Abernathy, G. A. Hackett, A. D. Rollett, P. A. Salvador, and E. A. Holm, "Microstructure generation via generative adversarial network for heterogeneous, topologically complex 3d materials," *JOM*, vol. 73, no. 1, pp. 90–102, 2021.

[3] A. Agrawal and A. Choudhary, "Deep materials informatics: Applications of deep learning in materials science," *MRS Communications*, vol. 9, no. 3, pp. 779–792, 2019.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[5] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[6] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[7] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans. arxiv preprint arxiv: 161009585," 2016.

[8] D. B. Brough, D. Wheeler, and S. R. Kalidindi, "Materials knowledge systems in python—a data science framework for accelerated development of hierarchical materials," *Integrating materials and manufacturing innovation*, vol. 6, no. 1, pp. 36–53, 2017.

[9] F. M. Altufaili, H. R. Mohammed, and Z. M. Hussain, "A correlative information-theoretic measure for image similarity," *Global Journal of Pure and Applied Mathematics*, vol. 13, no. 2, pp. 159–169, 2017.