Comparison of two stochastic generative approaches for the simulation of Earth imagery

Caroline Violot

March 2020

Contents

1 Introduction

Nowadays, machine learning is a precious tool in several application fields to explore the variability of natural/artificial processes, to infill missing data, to generate input data for numerical models or analyze their output. The domains of application are expanding everyday and remote sensing is one of them. In this study, the aim is to compare two state-of-the-art stochastic generative techniques to simulate the heterogeneity observed in multi-spectral satellite imagery, widely used to monitor Earth surface processes.

The first approach considered is Generative Adversarial Networks (GAN)[1], a recent family of neural network that is being widely applied in the fields of image analysis and remote sensing. The fundamental GAN architecture consists of two neural networks playing against each other. One generates new images and the other discriminates them against real images. The two neural networks iterate the generation while playing a minmax game where the discriminator gets better at classifying real and fake images and the generator gets better at creating images that fool the discriminator. After a usually intensive training phase, this strategy allows the generator to create realistic images that preserves complex features of the training images. The architecture of GANs will be detailed in Section 2.

The second technique considered is Multiple-Point Statistical (MPS) resampling techniques, originally applied to the geological domain and more recently to environmental remote sensing. MPS resampling aims at generating realistic images by randomly sampling a training dataset and forming new data patterns. Simulated with an iterative workflow, the output images preserve complex spatial features and statistical relations that characterize natural processes, avoiding the formulation of complex probabilistic models.

MPS methods require less data than GAN and no training, but the generation process is less optimized and slower. On the other hand, MPS can easily incorporate auxiliary variables and conditioning data to guide the simulation of complex images. Comparing MPS approach and GAN approach for the generation of multiple type of images gives us insight on the weak/strong points of each method and allows us to suggest future methodological improvements for the considered applications.

In section 2, a more in-depth explanation of the two approaches is given, along with the statistical tools that are used to assess their performance. Section 3 presents the implementation, including the datasets choice and generation, the chosen design for the GANs and the parameters for the MPS method. In section 4, the results are presented and analysed. Finally, in section 5 we discuss the results and exposes our thougths on the use of GANs in earth imagery contexts.

The project is hosted at the Geostatistical Algorithm and Image Analysis (GAIA) laboratory in UNIL.

2 Theoretical Background

2.1 Generative Adversarial Networks (GANs)

Previous machine learning generative models focused on learning the distribution of different dimensions of an image. The generation of new images was then a matter of sampling from those complex, high-dimensional training distribution. Rather than solving this learning problem, GANs sample from a simple distribution (e.g.: random noise) and learn the *trans-*

formation to training distribution. In this part we start by describing the general theory behind GANs and exemplify with a simple multi-perceptron GAN. We then focus on deep convolutional GANs, a particular type of GAN which make use of deep convolutional neural networks.

2.1.1 Textbook GANs

The idea behind GANs is often illustrated using the "art forgery" example. Art forgery is the process of creating and selling works of art which are falsely credited to a famous artist. The generator tries to fool the discriminator by generating real-looking images and the discriminator tries to distinguish between real and fake images. Concretely, A GAN trains two neural nets simultaneously: a generator $G(Z)$ and a discriminator $D(Y)$, where $G(Z)$ draws samples from a random distribution and $D(Y)$ draws samples from an image distribution.

The two networks train jointly in a minimax game with the following objective function :

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

where:
- x is a real image sample, drawn from p_{data},
- z is a 1D-array, drawn from a random distribution $p(z)$,
- $D_{\theta_d}(y) \in [0, 1]$ is the probability that image y is real, given by $D(y)$, using the learned θ_d parameters, and
- $G_{\theta_g}(z)$ is the image generated by $G(z)$, using the learned θ_g parameters.

The discriminator $D(y)$ wants to maximize the objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake). The generator $(G(z))$ wants to minimize objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking that $G(z)$ is real).

Then, training a GAN is the alternance between :

1. Gradient descent on discriminator :

$$\max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

2. Gradient descent on generator :

$$\min_{\theta_g} [\mathbb{E}_{z \sim p(z)} log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

One of the steps in the GAN design is to actually chose which kind of neural networks are to be used for the generator and the discriminator. As this project aims to explore a classical GAN, two simple GAN were designed, one where the two neural networks are simple **multi-layered perceptron** (MLP) and one where the neural networks are **deep convolutional networks**. Both of them are described in the following subsections.

2.1.2 Multi-Layered Perceptron (MLP) GAN

Multi-layered perceptrons are one of the most basic types of neural networks, as they consists of fully connected layers with a leaky Rectified Linear Unit (ReLU) activation function between each layers. A typical MLP Neural Network can be seen in figure 1.

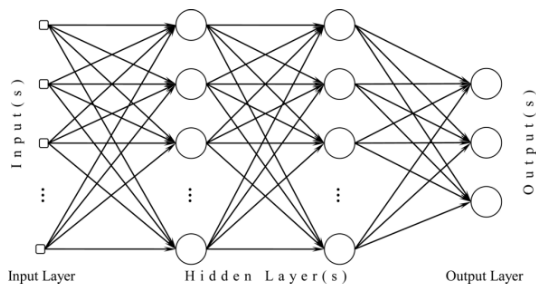

Figure 1: Multi-Layered Perceptron Neural Network (image source:http://www.mdpi.com/)

In a MLPGAN in particular, the generator is fed a random noise vector (usually following a normal distribution) and the input layer is fully connected to the next layer. Then, there is an alternance between leaky ReLU activation and fully connected layers of different sizes until the network can output a (usually) flattened image of the wanted size, meaning the same size as the real images. The Discriminator then takes in the real images, labeled 1 or the generated images labeled 0 and pass them through a similar network until the last layer of dimension 1. This result is finally filtered with a sigmoid activation function, which output 1 if it classified as real by the Discriminator or 0 otherwise. The discriminator gradients are then updated using two losses, one in relation with $D(x)$, with x a real image, and which should be 1 and one in relation with $D(G(z))$, with $G(z)$ a generated image, and which should be 0. The generator is himself updated with the loss concerned with how close to 1 is $D(G(z))$.

This neural network, as was said before, is a very simple one and is not expected to yield good results but it can be useful to understand GANs and GANs training, so designing it and quickly testing it is a good starting point.

2.1.3 Deep Convolutional GAN (DCGAN)

Deep Convolutional Neural Networks (CNNs) are a special kind of Neural Networks which are typically used in image classification and more generally analyzing visual imagery. Hidden layers of a CNN consist of a series of convolutional layers, which are connected to each other with a sort of convolution operation instead of a matrix multiplication as in some other neural networks like MLP . Another characteristic of a CNN is the presence of a pooling operation, which apply a function on a determined location (for example a small set of adjacent pixel) and return the result of the function instead of the location information [2]. Finally a batch normalisation operation is typically present in CNNs.

In the DCGAN, the discriminator is a classical convolutional neural network which classify images into real or fake, by extracting the features of its input. The generator start from a random noise vector as in the previous model but it performs several deconvolution operations to create the different features that the discriminator is going to examine.

After a few failures which are discussed in Section 5, we used the architecture by Radford and al. presented in their original paper *Unsupervised representation learning with deep convolutional generative adversarial networks* to design our DCGAN. The structure is detailed in section 3.

2.2 Multiple point statistics (MPS)

Multiple point statistics (MPS) is a geostatistic method for representing complex structures using high-order nonparametric statistics [3]. MPS simulations come with a variety of approaches, depending on the type of needing simulations, whether it is (un)conditional simulation, gap-filling, down-scaling, spectrally enhanced remote sensed imagery, and many more. In this project I used the Quantile Sampling (QS) algorithm [4], a general-purpose pixel-based MPS algorithm. Pixel-based methods use a set of pixels representing various properties of a phenomenon in order to simulate one pixel at a time. They allow flexibility in handling the conditioning data and they rarely produce artifacts when dealing on complex structures. The number of used points/pixels and the impact of each vary between the different methods that are built on this approach.

The main structure of pixel-based MPS simulation algorithms [5] was used for QS with an adaptation in the selection of candidate patterns.

2.3 Statistical Comparison of images

Three common statistical methods were used to compare the images generated with the different approaches: histogram comparison, covariance analysis and k-mean classification connectivity. Here we briefly introduce those methods and in Section 4, we present the results obtained with each of them.

2.3.1 Histogram Comparison

The first indicator used to compare our generated images is a histogram. In this method, the different pixel values of the image are counted and showed using a histogram. The range of pixel value is divided into several bins i and the number of pixels corresponding to each bin is counted and stored in n_i. For an image of n pixels and for k the number of bins we have that :

$$n = \sum_{i=1}^{k} n_i \tag{1}$$

This method only compare the overall pixel value distribution between the two images we are comparing, but no spatial patterns are taken into account. For example any images with the same number of black and white pixels would yield the same histogram independently on how those pixels are arranged in the image. The spatial arrangement are controlled with the next two methods.

2.3.2 Covariance Analysis

The second indicator we use is a sample covariance function. By considering 2D images as instance of a random field Z(X) on a domain D, we can calculate a covariance function C(x, y) which gives the covariance of the values of the random field at the two locations x and y, according to the covariance formula [6]:

$$C(x,y) := \operatorname{cov}(Z(x), Z(y)) = \mathbb{E}[\{Z(x) - \mathbb{E}[Z(x)]\} \cdot \{Z(y) - \mathbb{E}[Z(y)]\}] \tag{2}$$

In our covariance analysis method, the point x and y are randomly picked in order to accelerate the computation and we obtain an approximation of the function, which can be parameterized to allow a trade-off between computation time and precision.
In result we obtain a curve in function of x and which we can then interpret and compare with the covariance function curve of a reference image.

2.3.3 K-means classification connectivity

Finally, k-means classification connectivity is a method to classify pixels into different classes and then compare the connectivity of those classes. Pixels are classified according to their value in greyscale, which allow us to work on a categorical image with the desired number of different categories. Then for each category, the python `skimage.measure.regionprops` is used to access the properties of the considered category, including the mean area for each cluster of one category, which can be interpreted as the connectivy as a greater mean area indicates that the pixels of one categories can be found next to each other more often.

3 Methods and Implementation

3.1 Datasets

To assess the quality of our two apporaches, two types of datasets were used. The first one was a dataset consisting of Gaussian Field images, which allowed us to easily compare the result with precise metrics. The second type of datasets contained images with more complex features and required more training for the GAN but were also closer to real-world images, and in so being able to generate them was more critical.

3.1.1 Generation of the Gaussian images datasets

In order to train the GAN on Gaussian fields, two subdatasets were created, both containing 30 000 images. The first dataset contained images of size 28x28 pixels, whereas the second dataset contained images with 64x64 images. The images were created by convolving a Gaussian kernel with random noise and some of the resulting images can be seen in figure 2. The generation of the dataset allowed to avoid having to generate the Gaussian field images during the training but to have it already created and to be able to build a Tensorflow or Pytorch input pipeline to optimize the training of the GANs. It also means that the images would only have to be created once, which lowers drastically the computation cost.

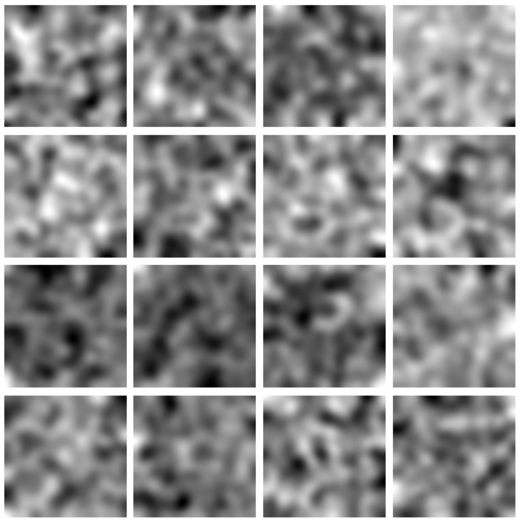

Figure 2: Some images of the Gaussian dataset

3.1.2 More complicated datasets

The more complex images used to train the GAN came from datasets generally used in the GAIA laboratory to design and compare generative methods. They are images with more complex features but also with repetitive patterns, which allows to use the same statistical methods as for the Gaussian Field images (although with different parameters) and still have meaningfull results. The quality of the result would indeed be more tricky to statistically assess with for example, nature pictures from a bank images.

The images were generated using the MPS method with the objective of training Machine Learning Models on them. A sample of the datasets can be seen in figures 3 and 4.

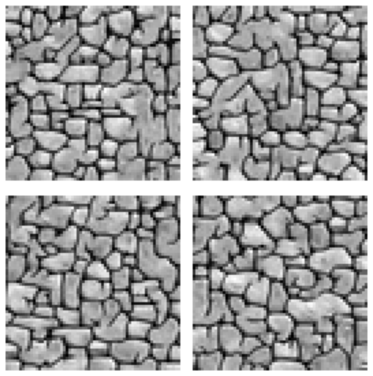
Figure 3: Images from the ”Stone” GAIA dataset

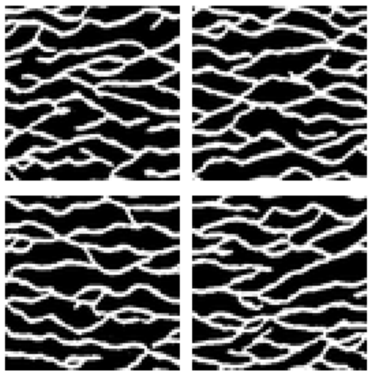
Figure 4: Images from the ”Strebelle” GAIA dataset

3.2 MPS generation

The methodology to launch a MPS generation is defined in https://gaia-unil.github.io/G2S/. As this project did not aim to modify MPS generation but to propose an alternative with GANs, the generation of images using this approach is not described here. Images generated with the MPS can still be seen in figure 5.

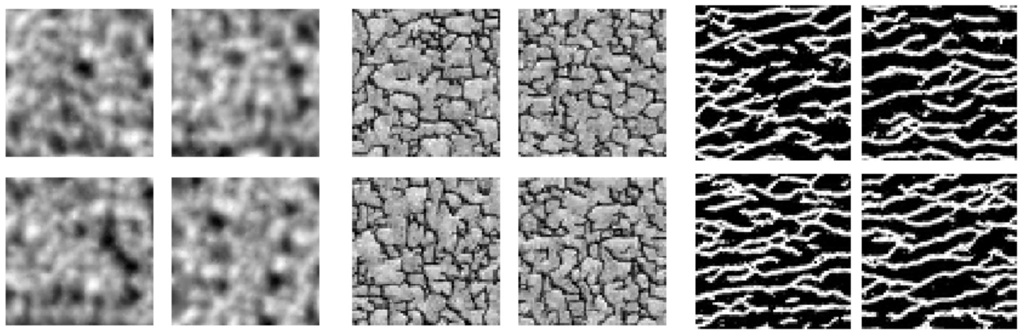
Figure 5: Images generated using the MPS method using one image from the Gaussian dataset (**left**), one image from the Stone dataset (**middle**) and one image from the strebelle dataset (**right**)

3.3 GAN generation

3.3.1 MLP GAN

In this network a random noise vector of dimension 100 is fed to an input layer, itself fully connected to dense layer of size 128. This layer is connected through a LeakyReLU activation function to the output dense layer which has a size depending on the wanted image size (in our case a size of 28x28=784). Therefore the generator $G(z)$ transform random input into a 28x28 images which can then be fed to $D(y)$ in addition to the real images for the discriminator to train. After 200 epochs with batches of 128 images, the generated images can be seen in figure 6. The losses of the two network are presented in figure 7. Both network don't really get better with time and the generated images are not satisfying. As this neural network was only a starting point for the GAN design and understanding, the results won't be further discussed.

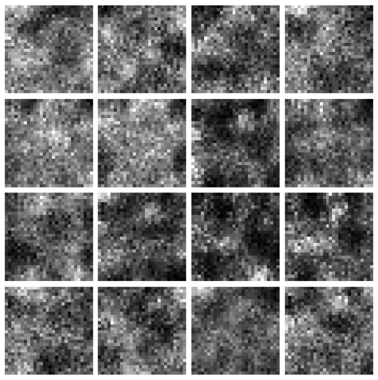

Figure 6: Generated images using the MLPGAN

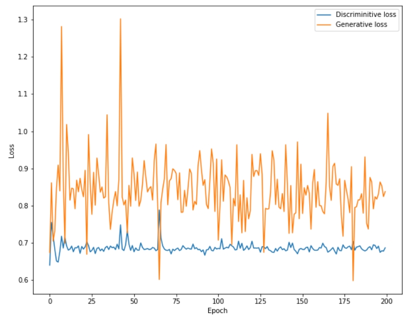

Figure 7: Losses of the generator and the discriminator of the MLPGAN during training

3.3.2 DCGAN

The next step, was to design and train a Deep convolutional GAN (DCGAN), as Deep Convolutional networks for the Generator and the Discriminator tends to yield good results in GANs [7].

I first followed the Deep Convolutional Generative Adversarial Network tutorial provided by tensorflow.org to have a general understanding of the architecture of the DCGAN and of how to deal with the successive structural resizing that are characteristic with Convolutional Neural Net.

From there I tried several hyperparameters, but I stumble across the same issue every time which was mode collapse. Namely, after a few iterations the generator would collapse to a parameter setting where it always produce from the same class (as can be seen in figure 8), in order to beat the discriminator which in turn focuses more on that class to discriminate it and the generator end up learning only one type of images which is not necessarily visually realistic.

The first solution to this problem was to drastically reduce the size of the batch on which the generator was trained, going from 128 to 16 or 32 images per iteration. And the second solution was to set the learning rates to be different between the discriminator and the generator. After several test, the best result appeared to be for the learning rate of the discriminator to be 4 times greater than the learning rate of the generator. After training this new version of the DCGAN the obtained results were much more satisfying as can be seen in Figure 8 (right). For similar reason than for the MLPGAN the results obtained with this DCGAN will not be further discussed either, only the results obtained with the final version of the DCGAN which is exposed right under are explored thereafter.

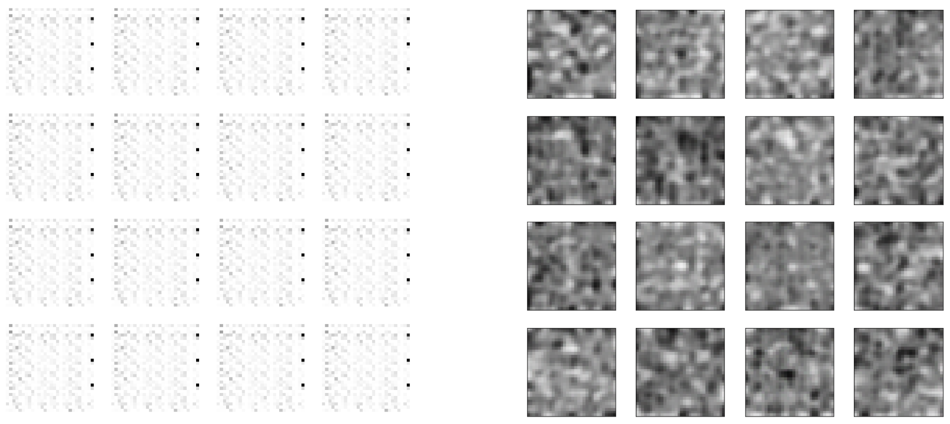

Figure 8: **Left:** Example of mode collapse of DCGAN, trained with 10 epochs (batch size = 128). **Right:** Images generated by the second version of the DCGAN after 10 epochs trained on whole dataset for each epoch (batch size = 32)
.

Final DCGAN designed for 64x64 images

At this point, the decision was made to switch from tensorflow to Pytorch as it offers stronger support for GPUs and the Neural Net would be trained on a cluster of GPUs. This switch allowed to design a DCGAN for images of 64x64, following the architecture defined by Radford and al. in 2016 [7], for which the generator architecture can be seen in figure 9.

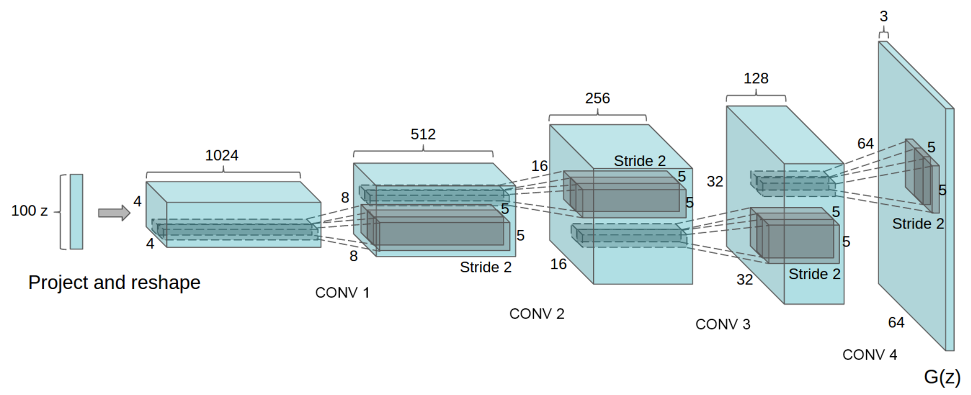

Figure 9: DCGAN Generator architecture to generate images of size 64x64, as defined by Radford and al. [7].

This final version of the GAN was trained on the different datasets with the parameters shown in table 1. The resulting images and statistical properties are presented and discussed in section 4.

	Gaussian	Stone	Strebelle
dataset size	30 000	25 750	198 000
n epochs	3	5	2
batch size	32	32	16

Table 1: Training parameters used for the DCGAN training on the different datasets

4 Results

In this section, the two different approaches are compared in term of quality of results and computation cost. This section is divided between the Gaussian Fields Validation, which allows to compare performance on simple images and the Application to more complicated images. In both applications, the results for the three statistical comparison methods described in section 2 are presented and commented. A colormap is automatically applied on the results for a clearer visualisation. The time performances will be discussed in section 5.

4.1 Gaussian Fields Validation

The generation of Gaussian field images is very easy as the features are not complex at all and does not require the help of advanced simulations techniques as GANs or MPS. Therefore, this part was mainly to create a baseline to compare the two approaches using rigorous distance from real images with the help of the very simple features of the Gaussian fields. We compare the images from our training dataset consisting of Gaussian Field images of size 64x64 and we import images generated by the GAN and the MPS methods.

Histogram Comparison

The reference is plotted as a line and the generated images are represented as boxplot on the same figure. The obtained results can be seen in figure 10.
Both reference values and generated image values were taken on 10 different images from the real dataset or the generated dataset. The reference images are the same for both methods and thus the reference curve is identical between the two figures.

Both methods exhibit histogram values close to the reference, but the std for the GAN method are relatively smaller than the std for the MPS and GAN images exhibit no outliers.

Covariance Function

The second statistic we want to use to compare our images is the covariance function. To do so, 3 images were generated with the MPS method, 3 images were generated by the DCGAN, and 3 images were randomly picked in the dataset. The covariance function was then computed on each image as explained in the 2. The result are shown in figure 11. The pixels next to each others (with a distance (lag) less than 5-7 pixels are positively connected which is coherent with the images, indeed pixels close to each others are likely to have a similar brightness. On the other hand, around 10 pixels, the covariance function tends to become negative, as is expected from looking at the Gaussian images, bright area are next to dark area and vice-versa, so values 10 pixels from each other are likely to be on the opposite spectrum of brigthness. Finally the values of pixels far from each other do not vary with one another and this is supported by a covariance close to 0 for pixels which are more than ~ 10 pixels from each others. Both generative approaches exhibit image covariance curves close to the ones from the dataset images but the GAN approache yields covariance function closer to the real images covariance functions.

K-mean classification connectivity

Finally we classify our images in different categorical classes of pixel values and test their connectivity. The connectivity in Gaussian Field images is expected to be the highest for very bright pixels or very dark pixels as they make clusters of pixels, whereas the intermediate values are more likely to form smaller thin clusters around the bright or dark areas. This can be seen on figure **??** where pixel values has been classified into 7 categories.

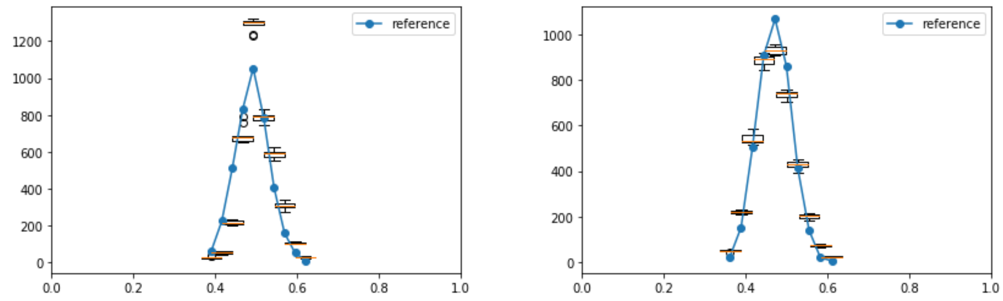

Figure 10: Histogram comparison between real images as the reference, images generated with the MPS method (**left**) and images generated with the GAN method (**right**)

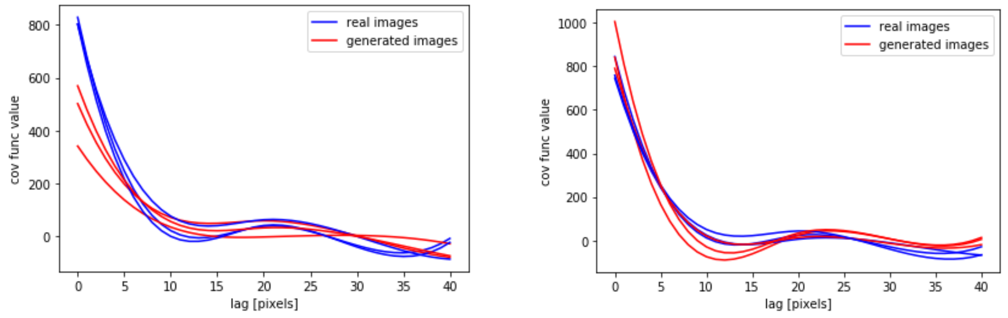

Figure 11: Comparison of the covariance function applied to 3 MPS generated images (**left**), 3 GAN generated images (**right**) and 3 Gaussian Field dataset images

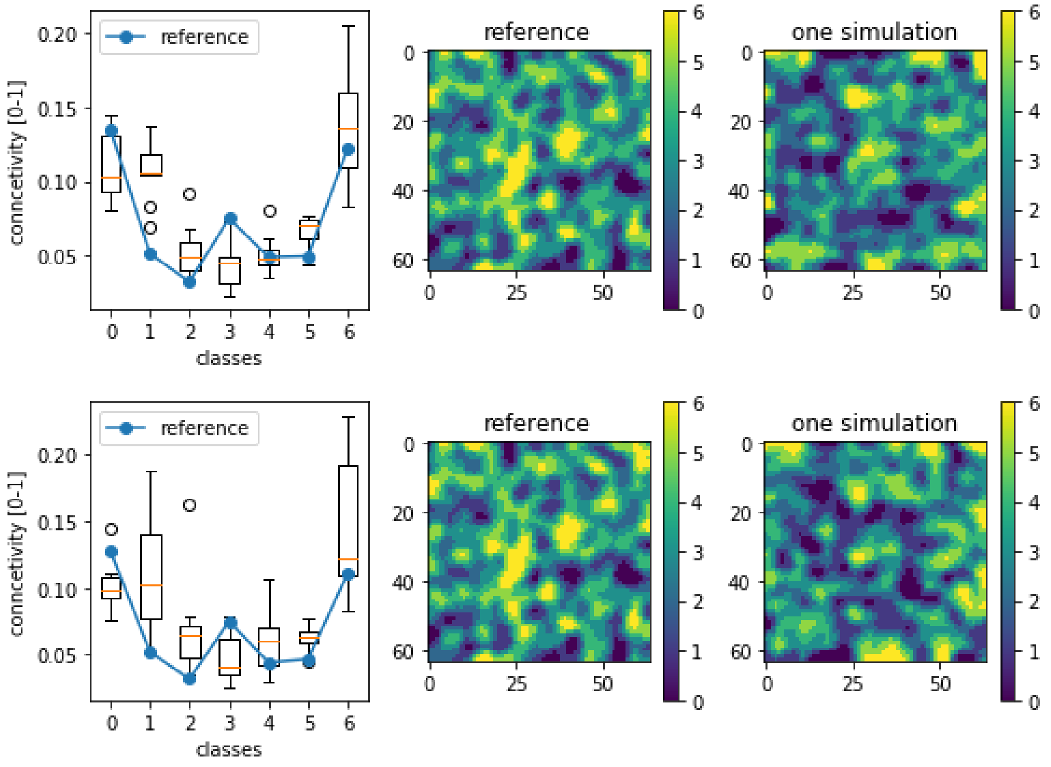

Figure 12: Comparison of K-mean classification connectivities between MPS generated images (**top**), GAN generated images (**bottom**) and dataset images for 7 classes of pixel values

4.2 Application to more complicated images

Finally, the same statistical methods were used to compare images generated by the two methods but this time the images were generated using more complex image datasets. Some of the final generated images can be found in the Appendix.

Histogram Comparison

To make up for the differences inherent to each images, the reference in this histogram comparison is in fact the mean between 3 images from the Stone dataset. In deed different images of the dataset presented very different histogram profiles and the comparison between them and the histogram profiles of generated images did not make a lot of sense. For the Strebelle dataset on the other hand, the histogram profiles were very similar between different images of the dataset (as was expected from a black and white picture) and so, only one image was used as the reference.

The histogram can be seen in figure 13 for the Stone images and in figure 14 for the Strebelle images.

Both method fit pretty well the reference curves for both datasets, but some particularities can be noted. First the GAN method seem to better catch the particular distribution of the pixel for the Stone images, where the MPS method histogram values tend to follow more of a Gaussian distribution again. On the other hand, as the MPS method randomly pick pixels from the test images, the fact that the Strebelle images pixels belong in $\{0, 1\}$ is very well reproduced by this method. On the contrary, the GAN method transforms a random images with close to continuous values into the learned distribution of the dataset images. This continuity can be seen in the fact that the bright values are more in $[0.8, 1]$, than only in $\{1\}$ as would be desired. The images generated with GAN are satisfying to look at in that they ressemble the images from the dataset, but for this particular type of images, a threshold filter applied to the image after its generation would yield results more close to the dataset images.

Covariance Function

The images from the stone dataset have a small pattern of around 5 pixels, similar to the pattern exhibit by the Gaussian fields images. And in a similar manner, this pattern can be seen in the covariance function curve as pixels next to each other tends to share the same value. And pixels afar from eachother are not correlated whatsoever. This is less the case for the Strebelle images, as the white lines usually extend on the whole images. The alternance between while lines and black areas can somewhat be seen on the reference covariance curve. This however could also be an artifact from the sampling of the pixels as it is not exhibit on all the reference curves and the generated images covariance curves don't display this behavior either.

K-mean classification connectivity

Finally, 5-mean classification yields good results for MPS generation but less good results for the GAN generated images. The overall shape of the curve is reproduced by the box-plots but the dark pixels are over-represented where the bright pixels are sub-represented in the GAN generated images. This could also be just an artefact from the computation in that the DCGAN updates its generator at each iteration and the last iteration could have produced a bit darker images than some of the previous versions of the generator. One way to deal with this would be to run the statistical tests on the generated images during the training of the GAN in order to end this training at the best moment possible instead of at the end of the epoch, as is done here.

For the Strebelle dataset, 3 classes only were used to classified. A black one, a white one and a small one in the middle with almost zero pixels. The images generated with the MPS method are a bit cut and so the connectivity of the bright class falls way under the connectivity of the reference image. The images generated with the GAN present a better connectivity profile. The middle class connectivity is null as the pixels are mostly isolated.

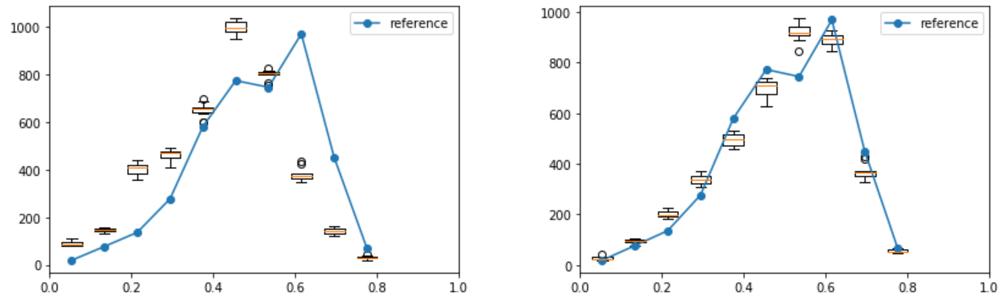

Figure 13: Histogram comparison between real images as the reference, images generated with the MPS method (**left**) and images generated with the GAN method (**right**)

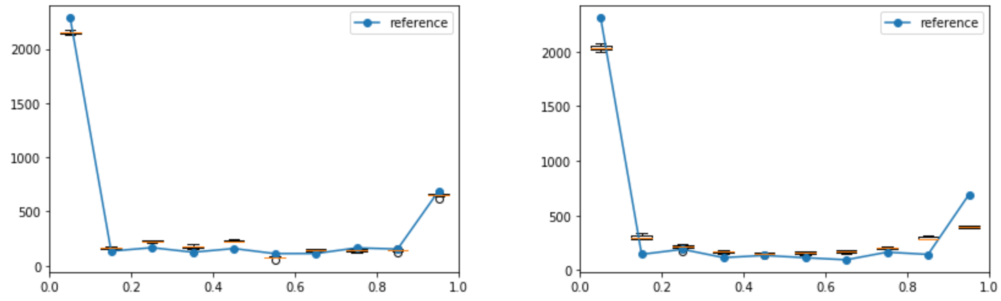

Figure 14: Histogram comparison between real images as the reference, images generated with the MPS method (**left**) and images generated with the GAN method (**right**)

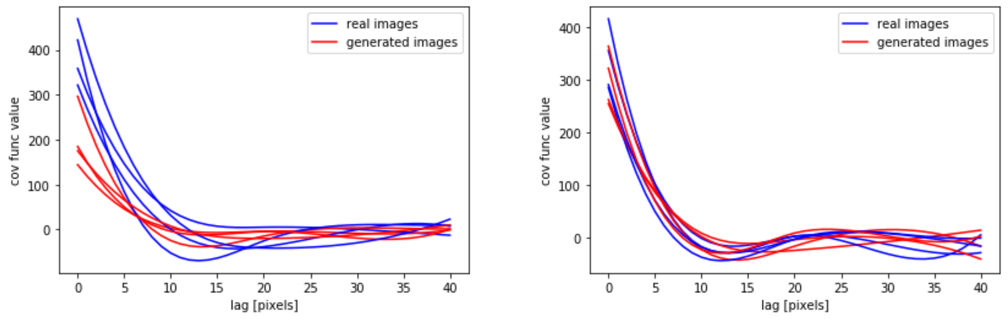

Figure 15: Comparison of the covariance function applied to 3 MPS generated images (**left**), 3 GAN generated images (**right**) and 3 Gaussian Field dataset images

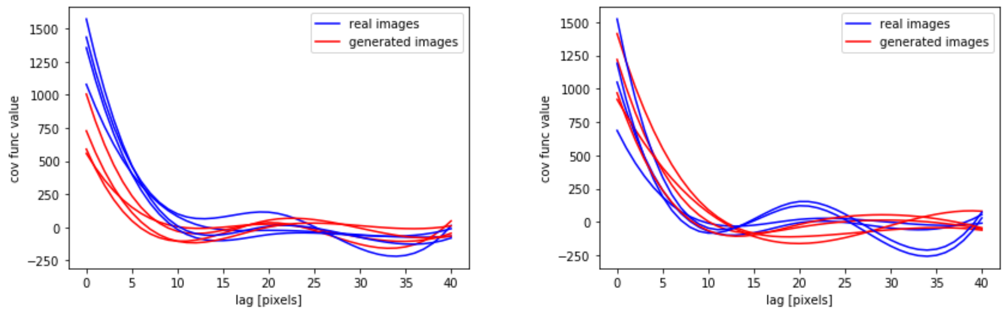

Figure 16: Comparison of the covariance function applied to 3 MPS generated images (**left**), 3 GAN generated images (**right**) and 3 Gaussian Field dataset images

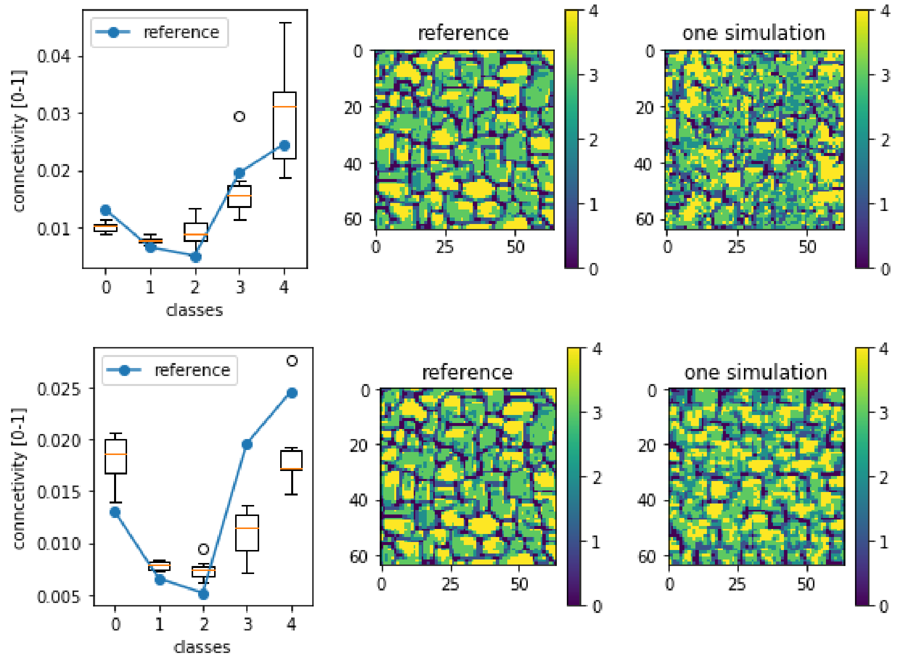

Figure 17: Comparison of K-mean classification connectivities between MPS generated images (**top**), GAN generated images (**bottom**) and dataset images for 7 classes of pixel values

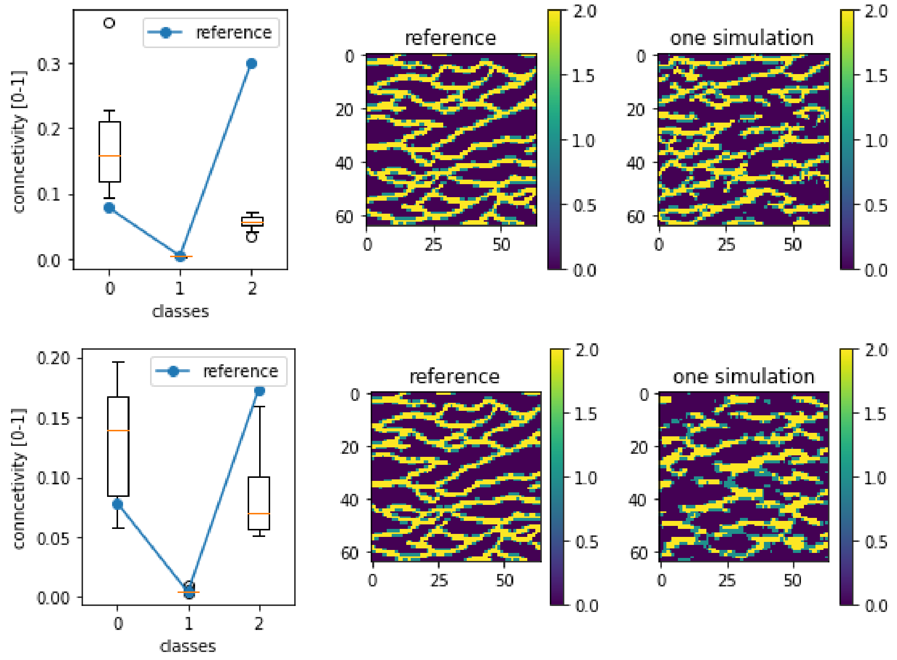

Figure 18: Comparison of K-mean classification connectivities between MPS generated images (**top**), GAN generated images (**bottom**) and dataset images for 7 classes of pixel values

5 Discussion

GANs are a relatively new methods. And DCGAN are even more recent, before the architecture presented by Radford and al. in 2016, previous attempts at DCGAN could not be considered successful [7]. But even so, between 2016 and today a lot of new architecture has been presented, for example in 2017, Arjovsky and al. presented Wasserstein-GAN [8], which proved to improve the stability of learning and solve the problem of modal collapse. Another example, as presented by Runde Li and al. in 2018, would be the use of conditional GAN dehazing [9] to enhance the generated images.

Nevertheless, this study aimed at determining if a classical GAN could be a good substitution for some dataset augmentation applications. As was shown in Section 4, for images of a size of 64x64 pixels, the results are mostly satisfying. Unfortunately, a simple adaptation of the DCGAN to 128x128 pixels images systematically lead to mode collapse. The basic architecture of DCGAN is therefore not sufficient to generate big images as it is and would require a careful design of the architectures of the Generator and the Discriminator, fine tuning all the parameters and this architectures and parameters would probably be particular to each type of images. This falls beyond the scope of this project.

Table 5 presents the training times for all types of image as well as the time required to generated 100 new images. The DCGAN trainings were performed on one node of Octopus [10], a high density hybrid GPU supercomputer, where each node hosts 2 CPUs and 4 Nvidia GeForce GTX Titan X GPUs.

	training	generation of a 100 fake images
DCGAN Gaussian (1 epoch)	43.283 s	0.92734s
DCGAN Stone (1 epoch)	37.224 s	0.82244s
DCGAN Strebelle (1 epoch)	196.88 s	0.86256s

Table 2: Time needed for 1 epoch of DCGAN training and the generation of 100 fake images on one node of Octopus.

The generation of 100 fake images after training took 25.468s on my personal laptop with an Intel® Core™ i5-8265U CPU @ 1.60GHz $\times$ 8. To compare, for the same size, the generation of 100 images with the MPS method took 71.721s on my personal laptop, which is substantially longer, even for small images of 64x64 pixels.

6 Conclusion

Overall, the results of the DCGAN training are satisfying. The statistical comparisons show that the generated images share similar properties with the dataset images, for all the different type of images. The issue of mode collapsing is present as in most DCGANs but could be overcome by modifying the design of the Generator and the Discriminator with more recent GAN alternatives.
Some simpler solutions could be applied in the meantime. First as was mentioned before, through the batch iterations of the training datasets, the DCGAN updates its weigth through gradient descent. And the last iteration is not necessary the better, so one way to deal with this particularity would be to stop training when the quality of the images attain a certain threshold. The quality of the images could be a combination of the losses of the GAN and

the statistical metrics defined in this report. Another issue is the continuity inherent to the generated images coming form the continuity in the noise vector randomly sampled by the generator to create new images. For the Strebelle images this was exhibit by the difficulty to create black and white images. for those types of image a simple threshold filter could enhance the resulting images.
In conclusion, the images are visually satisfying (some of the generated images can be found in the Appendix) and with the necessary amount of architecture design and parameters tuning to overcome mode collapsing, DCGAN should prove to be an interesting substitute to current generative methods like MPS.

References

[1] Goodfellow I.J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley, Ozair S. D., Courville A., and Bengio Y. Generative adversarial nets. 2014.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[3] Guardiano F.B. and Srivastava R.M. *Multivariate Geostatistics: Beyond Bivariate Moments. In: Soares A. (eds) Geostatistics Tróia '92. Quantitative Geology and Geostatistics, vol 5. Springer, Dordrecht.* 1993.

[4] Mariethoz G. and Gravey M. Quantile sampling: a robust and simplified pixel-based multiple-point simulation approach. 2019.

[5] Mariethoz G. and Caers J. *Multiple-point Geostatistics: Stochastic Modeling with Training Images, p. 156.* 2014.

[6] *Statistics for Spatial Data*, chapter 1, pages 1–26. John Wiley Sons, Ltd, 2015.

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[9] Runde Li, Jinshan Pan, Zechao Li, and Jinhui Tang. Single image dehazing via conditional generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] Swiss geocomputing centre - octopus (https://wp.unil.ch/geocomputing/octopus/).

7 Appendix

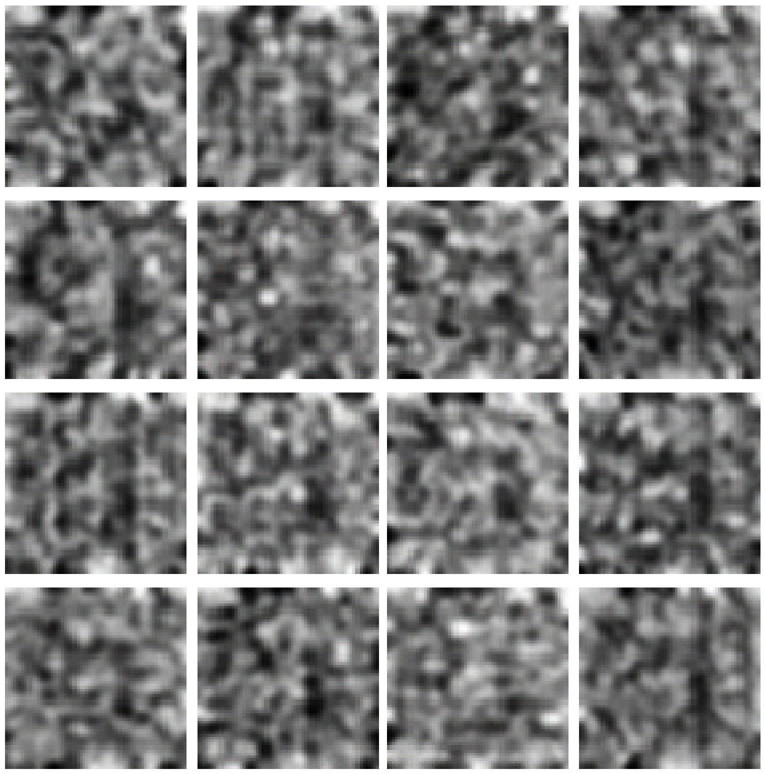

Figure 19: Fake Gaussian images generated by the DCGAN after training

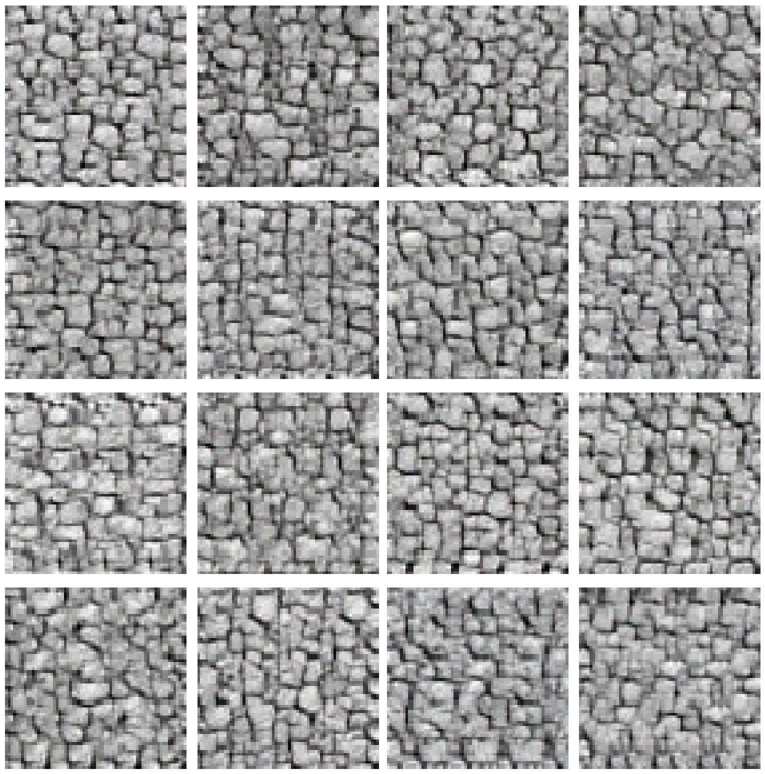

Figure 20: Fake Stone images generated by the DCGAN after training

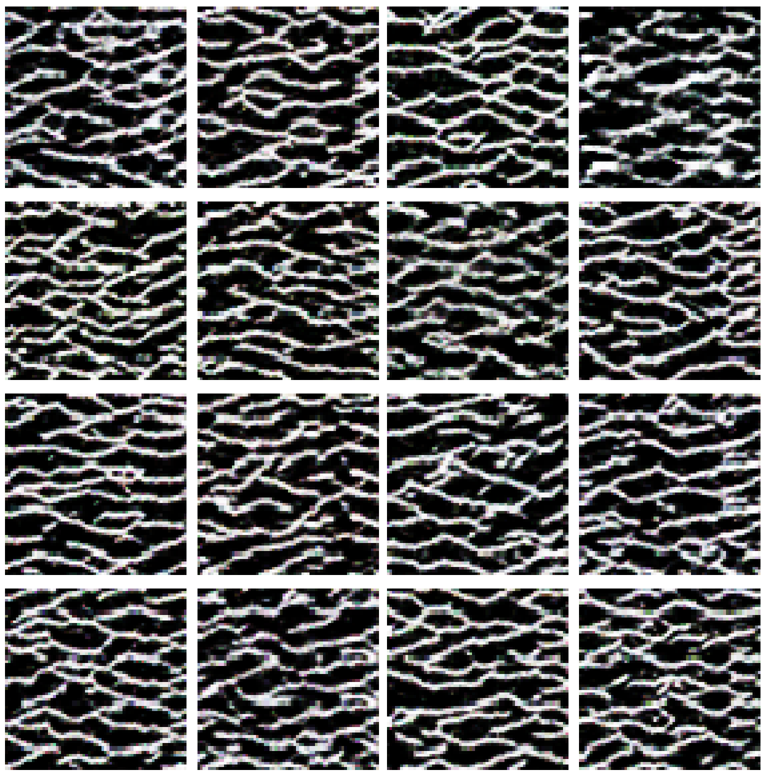

Figure 21: Fake Strebelle images generated by the DCGAN after training