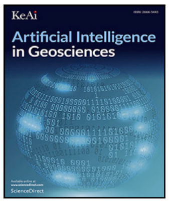
Contents lists available at ScienceDirect

Artificial Intelligence in Geosciences

journal homepage: www.keaipublishing.com/en/journals/artificial-intelligence-in-geosciences



Improving patch-based simulation using Generative Adversial Networks[☆]

Xiaojin Tan[*], Eldad Haber

Department of Earth and Ocean Science, The University of British Columbia, Vancouver, BC, Canada

ARTICLE INFO

Keywords:
Generative Adversial Networks
Patch-based simulation

ABSTRACT

Multiple-Point Simulation (MPS) is a geostatistical simulation technique commonly used to model complex geological patterns and subsurface heterogeneity. There have been a great variety of implementation methods developed within MPS, of which Patch-Based Simulation is a more recently developed class. While we have witnessed great progress in Patch-Based algorithms lately, they are still faced with two challenges: conditioning to point data and the occurrence of verbatim copy. Both of them are partly due to finite size of Training image, from which a limited size of pattern database is constructed. To address these questions, we propose a novel approach that we call Generative-Patched-Simulation (GPSim), which is based on Generative Adversarial Networks (GAN). With this method, we are able to generate sufficient (in theory an infinite) number of new patches based on the current pattern database. As demonstrated by the results on a simple 2D binary image, this approach shows its potential to address the two issues and thus improve Patch-based Simulation methods.

1. Introduction

Multiple-Point Simulation (MPS) (Guardiano and Srivastava, 1993; Hu and Chugunova, 2008) is a geostatistical simulation technique commonly used to model complex geological patterns and subsurface heterogeneity. There have been a great variety of implementation methods (Strebelle, 2002; Mariethoz et al., 2010; Peredo and Ortiz, 2011) developed within MPS, of which Patch-Based Simulation (Arpat and Caers, 2007; Zhang et al., 2006; Honarkhah and Caers, 2010; Tahmasebi et al., 2012; El Ouassini et al., 2008; Chatterjee et al., 2012; Rezaee et al., 2013) is a more recently developed class. In general, Patch-Based Simulation has better pattern reproduction and faster simulation speed than Pixel-Based Simulation (Guardiano and Srivastava, 1993; Strebelle, 2002; Mariethoz et al., 2010) since it synthesizes patches rather than pixels (Wei et al., 2009). Patch-Based Simulation relies on the use of Training images (TI) from which patches are extracted as the building blocks to construct multiple stochastic realizations. While we have witnessed great progress in Patch-Based algorithms recently, they are still faced with several challenges.

The first difficulty is conditioning to point data. Patch-based methods generally have more difficulty with conditioning to point data than Pixel-based algorithms. During the simulation, patches from TI are compared with the neighborhood of the current simulation point (termed Data Event) using the concept of distance (or similarity). The most similar candidate patches are selected and one of them is pasted onto the simulation grid. Data Event often includes hard data which are assumed known and certain, as a result the candidate patches must honor the conditioning hard data exactly. There is a likelihood that we could not find from TI a sufficient number of replicates exactly matching the hard data. What is even worse, no such patch from TI could be found if the hard data within the data event are dense.

The second difficulty is the occurrence of verbatim copy, which refers to portions of TI perfectly reproduced onto the simulation grid. This leads to the reduction of variability between realizations. Since one of the main goals of MPS (or say geostatistics) is the modeling of spatial uncertainty, it is desired if we can improve the variability between realizations to ameliorate the verbatim copy problem.

Both of the challenges above are partly due to finite size of TI (Caers and Zhang, 2004; Mariethoz, 2018; Emery and Lantuéjoul, 2014), from which a limited size of pattern database is constructed. As a result, it is possible that no patches from the database could exactly match a certain hard data pattern in the case of conditional simulation, leading to poor hard data reproduction; in the case of unconditional simulation, there is a possibility that just a few or perhaps even just one patch from the database, with a given conditioning data event, is available for selection, resulting in the occurrence of verbatim copy and a reduced variability between realizations.

These observations lead us to a key question concerning the size of TI: How to increase the size of TI if only one TI of finite size is available? More specifically, how to generate an unlimited number of new patches based on the current pattern database? To address

☆ Supported by UBC, Canada.
* Corresponding author.
E-mail address: xtan1566@gmail.com (X. Tan).

https://doi.org/10.1016/j.aiig.2023.05.002
Received 3 November 2022; Received in revised form 27 February 2023; Accepted 21 May 2023
Available online 7 June 2023
2666-5441/© 2023 The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

these questions, we propose a novel approach that we call Generative-Patched-Simulation (GPSim), which is based on Generative Adversarial Networks (GAN) (Goodfellow et al., 2014a). With this method, we are able to generate sufficient (in theory an infinite) number of new patches based on the current pattern database to address the issues above. In recent years, there have been many papers (Chan and Elsheikh, 2017; Mosser et al., 2017; Laloy et al., 2018; Dupont et al., 2018; Song et al., 2021; Chen et al., 2022; Pan et al., 2021) that use GANs for the generation of realistic earth models. However, to our knowledge, all these methods require a relatively large data set as input and have difficulties in generating new data when there is a single image. Our method is, by design, geared towards such scenarios, which, in our experience, are very common in realistic settings. Furthermore, our paper is different from other work as it solves other problems which are notorious in patch-based simulations: (i) lack of patches to match point data patterns exactly and (ii) the occurrence of verbatim copy. Finally, these methods, to our knowledge, are mainly focused on using GAN and its variants to replace MPS to generate geological realizations. Our work is focused on using GAN to improve patch-based algorithms.

The paper is structured as follows. A literature review is conducted in Section 2. In Section 3 we present an overview of GPSim that is developed to improve Patch-based simulations. Experimental results are demonstrated in Section 4. Finally, in Section 5, we summarize the paper.

2. Literature review

2.1. Using GAN to generate geological realizations

GAN is a class of deep neural network architectures consisting of two competing nets trained simultaneously by an adversarial process. Given a training dataset, it learns to create new data such that both of new and original data come from the same distribution. The idea of using GAN and its variants to generate geological models is not new. Chan and Elsheikh (2017), Mosser et al. (2017), Laloy et al. (2018), Dupont et al. (2018), Song et al. (2021), Chen et al. (2022), Pan et al. (2021) try to generate realistic geological models by substituting GAN for MPS. They all require a sufficient number of training images as input to the Discriminator.

Unlike the above papers that generate realistic geological models by substituting GAN for MPS, we take another direction and propose a new approach that combines the ability of GAN to model a distribution of patches and the ability of MPS to generate realizations with one single TI. Our proposed method is mainly focused on using GAN to improve performance of Patch-based algorithms which have two notorious challenges: (a) conditioning to point data; (b) the occurrence of verbatim copy.

2.2. Generating new patterns

Generating new patches used in patch-based simulation is always desirable due to finite size of TI. Recently, Rezaee et al. (2015) enriches the pattern database by generating many independent realizations from which more patches can be extracted. These realizations are obtained by passing the original training image to a patch-based multiple-point simulator. Our paper and Rezaee et al. (2015) address the same problem of improving patch-based algorithms by enriching pattern database. The main difference is as follows:

– the way of generating new patches is different. As stated in Jetchev et al. (2016), patch-based simulations do not “learn” any models of the textures of interest but just reorder the input TI using local similarity. Therefore, it is argued that Rezaee et al. (2015) does not “learn” new patches but generate patches from reordered TI. In our paper, new patches are generated by using GAN, which has the well-known capability of learning a distribution of natural images.

It is noteworthy that we provide an alternative way to enrich the pattern database. There is no conflict between our method and any other way of generating new patches, which means that a combination of these methods can be used to generate more diverse patterns and further improve the performance of patch-based simulation.

3. Methodology

3.1. Construct pattern database

We denote by **ti** a training image consisting of b pixels which are β-dimensional feature vectors,

$$\mathbf{ti} \in \mathbb{R}^V \qquad V = b\beta \tag{3.1}$$

in which $\beta = 1$ for a grayscale training image or $\beta = 3$ for an RGB training image. We then scan the training image **ti** with a template of size ω to extract a list of patches (multi-point vectors), denoted as data events $\mathbb{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_t\}$, where t, the number of patches extracted from **ti**, can be represented as follows,

$$t = (\sqrt{b} + \omega - 1)^2 \tag{3.2}$$

These patches can be used as training dataset to train our GAN model to generate new patches that resemble patches from the dataset.

3.2. Enrich pattern database

The problem can be stated as follows: Given a pattern database $\mathbb{P}$ consisting of t patches which come from the same distribution $\mathbf{P}_{data}$, we would like to generate new patches coming from a probability distribution $\mathbf{P}_G$ which is as close as possible to $\mathbf{P}_{data}$ To achieve this, we will follow an approach called GAN (Goodfellow et al., 2014b) for the generation of such new patches. GAN consists of two competing neural networks trained simultaneously by an adversarial process. These two nets, called the Generator and the Discriminator, are essentially differential functions. We first introduce the Generator, which can be formulated as follows,

$$\tilde{\mathbf{p}} = G(\mathbf{z}; \boldsymbol{\theta}_g) \tag{3.3}$$

where we denote by G the Generator, which takes a random vector $\mathbf{z} \in \mathbb{R}^{n_z}$ as input and outputs a high-dimensional vector, specifically a synthetic patch $\tilde{\mathbf{p}} \in \mathbb{R}^{n_p}$ in our case. Note that $n_z \ll n_p$ and the random vector **z** is assumed to come from a prior distribution $\mathbf{P_z}$ (e.g. a uniform distribution or a Gaussian distribution). Given a fixed prior distribution $\mathbf{P_z}$, the weights of the generator G, denoted by θ_g are the sole determinant for the shape of the distribution $\mathbf{P_G}$. The goal of G is to update θ_g so that $\mathbf{P_G}$ is as close as possible to the target distribution $\mathbf{P}_{data}$, which can be formulated as follows,

$$\underset{\theta_g}{\text{minimize}} \quad \mathbb{DIV}(\mathbf{P_G}, \mathbf{P}_{data}) \tag{3.4}$$

where $\mathbb{DIV}$ indicates a function called divergence that measures the similarity between two probability distributions $\mathbf{P_G}$ and $\mathbf{P}_{data}$.

To compute the divergence between these two distribution, the other component of GAN called the Discriminator will be introduced below,

$$y = D(\mathbf{p}; \boldsymbol{\theta_d}) \tag{3.5}$$

where we denote the discriminator by D, which takes a patch **p** as input and outputs a scalar y. Larger output value y means a higher probability that an input patch **p** comes from the true distribution of training patches. More specifically, the parameters $\boldsymbol{\theta_d}$ will be updated to assign high values to the patches coming from the pattern database and low values to the patches generated from the generator G. This boils down to a binary classification problem and can be formulated as follows,

$$\underset{\theta_d}{\text{maximize}} \quad \mathbb{E}_{\mathbf{p}\sim\mathbf{P}_{data}}[\log D(\mathbf{p};\theta_d)] + \mathbb{E}_{\mathbf{p}\sim\mathbf{P_G}}[\log(1 - D(\mathbf{p};\theta_d))] \tag{3.6}$$

Practically, we do not know how to compute explicitly the two distributions but we know how to generate samples from them, therefore (3.6) can be reformulated as,

$$\underset{\theta_d}{\text{maximize}} \quad \frac{1}{m}\sum_{i=1}^{m}\log D(\mathbf{p}^i;\theta_d) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(\tilde{\mathbf{p}}^i;\theta_d)) \tag{3.7}$$

whose maximum objective value is related to Jensen–Shannon divergence (Goodfellow et al., 2014b). By combining (3.4) with (3.7), the problem of enriching pattern database boils down to a minmax optimization problem,

$$\min_{\theta_g}\max_{\theta_d} \quad \frac{1}{m}\sum_{i=1}^{m}\log D(\mathbf{p}^i;\theta_d) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(\mathbf{z}^{\mathbf{i}};\theta_g);\theta_d)) \tag{3.8}$$

which, in practise, can be iteratively solved in two successive steps: (1) fix θ_g and update θ_d to maximize (3.7); (2) fix θ_d and update θ_g to minimize (3.4). As a result, the optimum θ_g can be used in (3.3) to generate new patches, denoted as $\tilde{\mathbb{P}} = \{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_s\}$, where s is the number of new patches generated.

3.3. Simulation with patterns

By combining these two groups of patches $\mathbb{P}$ and $\tilde{\mathbb{P}}$, a new pattern database $\mathbb{Q}$ is created and can be used to generate realizations. $\mathbb{Q}$ can be considered to come from a larger and pattern-richer "training image". The simulation, in general, follows a typical patch-based approach, except that our patches not only come from the original TI, but also from the Generator. A typical patch-based approach can be summarized as follows,

– The input to this simulation process consists of an empty grid (SG) whose nodes are denoted as $\mathbf{u}$, an enriched pattern database $\mathbb{Q}$, and, if available, a set of hard data which are assigned to their nearest grid nodes in the SG.
– The path is defined along the rest of nodes in the SG. There are two common simulation paths, one is random and the other is unilateral. Either one of them can be employed.
– For each node $\mathbf{u}$ of the simulation grid along the path, a pattern centered on $\mathbf{u}$, denoted as $\mathbf{B}(\mathbf{u})$, is extracted using a fixed template that is the same size as patches in $\mathbb{Q}$. This pattern $\mathbf{B}(\mathbf{u})$, termed data event, consists of previously simulated values and/or any hard data in the neighborhood of the node $\mathbf{u}$.
– We calculate a distance (mismatch) between the data event $B(\mathbf{u})$ and the patch $\mathbf{q}_i$ in $\mathbb{Q}$,

$$\min_i \quad d(B(\mathbf{u}),\mathbf{q}_i) \quad . \tag{3.9}$$

once the most similar patches in $\mathbb{Q}$ are found, we randomly select one of them and paste it into the simulation grid to replace the current data event.
– A realization $\mathbf{y}$ is constructed by proceeding to next node until all nodes of the grid have been simulated.

There are several things worth noting in the above summary:

– the pattern recognition literature (Ashby and Ennis, 2007) provides many different distance functions, which should be selected properly according to the property of our variables. A simple example of a distance function that often works well for continuous variables is the Euclidean distance, expressed as,

$$d(B(\mathbf{u}),\mathbf{q}_i) = \frac{1}{\rho}\sum_{j=1}^{\rho}\omega_j \left\|\mathbf{B}_j - \mathbf{q}_{ij}\right\|^2 \tag{3.10}$$

where $\mathbf{B}$ is abbreviated from $B(\mathbf{u})$; ρ is the total number of nodes in the data event; ω_j is the weight associated with the j_{th} node. A large weight will be assigned if the node is hard data.

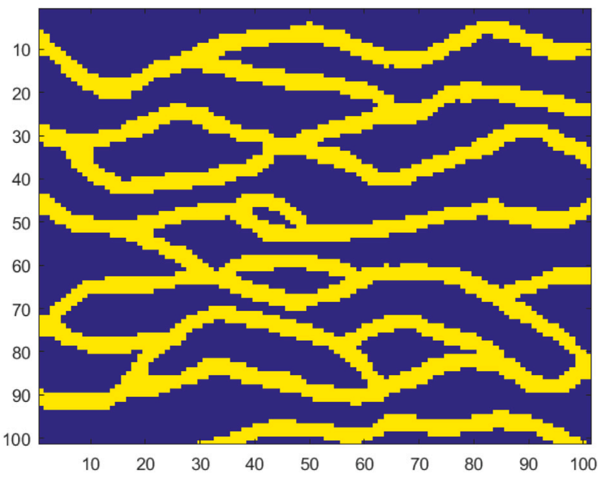


Fig. 1. Training Image.

For categorical variables, a simple measure of similarity can be represented as,

$$d(\mathbf{B},\mathbf{q}_i) = \frac{1}{\rho}\sum_{j=1}^{\rho}\omega_j \left\|\mathbf{B}_j - \mathbf{q}_{ij}\right\|^0 \tag{3.11}$$

which indicates the percentage of ill-matched nodes.
– Instead of comparing $\mathbf{B}(\mathbf{u})$ with every patch $\mathbf{q}_i$ in $\mathbb{Q}$, we can first classify patches of $\mathbb{Q}$ into a certain number of categories based on a similarity criterion. Each class is then represented by the average of all its patterns, namely a prototype. As a result, we only need to compare $\mathbf{B}(\mathbf{u})$ with every prototype.
– there are various implementation methods developed within patch-based approaches. The emphasis in the paper is not on describing all implementation methods. It is on how to improve the performance of those existing methods using GAN. For a detail review of patch-based algorithms see, e.g. Hu and Chugunova (2008), Mariethoz and Caers (2014), Mariethoz and Lefebvre (2014), and references therein.

4. Experiments

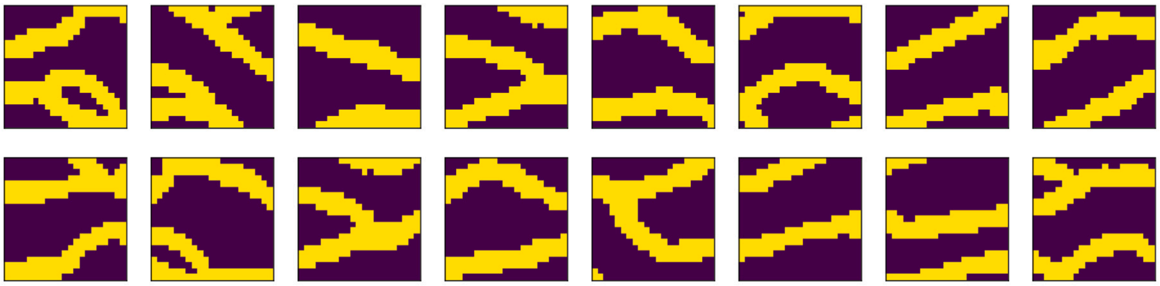
4.1. The dataset

We start with a two-dimensional binary training image of size 101 × 101 (Strebelle, 2002) shown in Fig. 1. The yellow color stands for the high permeability channels, and the blue color stands for the low permeability background. We then scan the training image with a template of size 21 to extract 6561 patches of size 21 × 21. These patches will be used as training dataset to train our GAN model to generate new patches. Some of these patches are shown in Fig. 2(a).

4.2. Generate new patches

Our GAN model consists of two neural networks, called the Generator and the Discriminator, respectively. The architecture of our GAN model generally follows Radford et al. (2015).

– The Discriminator consists of 3 blocks followed by a fully connected layer (FCL). Each block includes a convolution layer (CL) and a leaky ReLU layer. Batch normalization (BN) is used on each CL except the first block. The Discriminator has a Binary Cross-Entropy Loss function on the last (fully-connected) layer. The architecture of the Discriminator is shown in Table 1.


(a) patches extracted from the training image

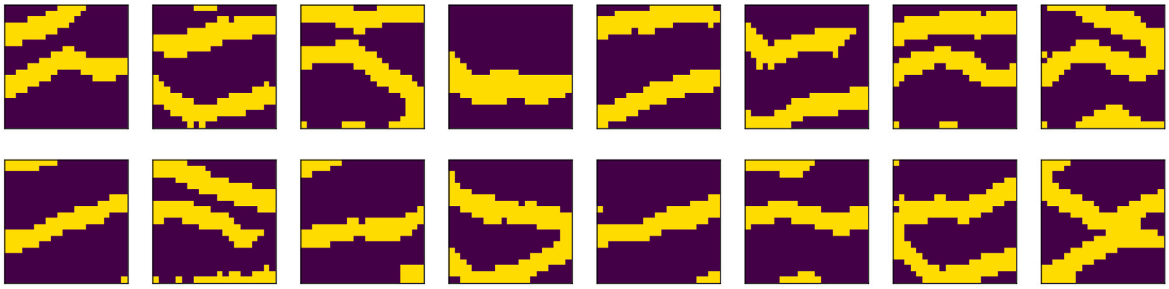

(b) patches generated by the generator

Fig. 2. (a) (b)

Table 1
The architecture of the discriminator.

	Block 1	Block 2	Block 3	FCL
layers	conv→leaky ReLU	conv→BN→leaky ReLU	conv→BN→leaky ReLU	FCL
input size	21×21	$10 \times 10 \times 32$	$5 \times 5 \times 64$	$2 \times 2 \times 128$
kernel size	4×4	4×4	4×4	
# of filters	32	64	128	
stride	2	2	2	
output size	$10 \times 10 \times 32$	$5 \times 5 \times 64$	$2 \times 2 \times 128$	1

Table 2
The architecture of the generator.

	FCL	Block 1	Block 2	Block 3
layers	FCL	TCL→BN→ReLU	TCL→BN→ReLU	TCL→ReLU→*tanh*
input size	50×1	$2 \times 2 \times 128$	$5 \times 5 \times 64$	$10 \times 10 \times 32$
kernel size		4×4	4×4	4×4
# of filters		128	64	32
stride		2	2	2
output padding		1	0	1
output size	$2 \times 2 \times 128$	$5 \times 5 \times 64$	$10 \times 10 \times 32$	21×21

– The Generator starts with a FCL followed by 3 blocks. Each block includes a transposed convolution layer (TCL) and a ReLU layer. BN is used on each TCL except the last block. A tanh activation function is applied to the ReLU layer of the last block. The architecture of the Generator is shown in Table 2.

We trained our models using the Adam optimizer with a batch size of 32 patches on a cluster server with Intel(R) Xeon(R) CPU 376 E5-2670 v3 for 500 epochs. As suggested in Radford et al. (2015), the learning rate is set to 0.0002, and the first moment is set to 0.5. The time needed for training is a fixed cost (roughly 23 min), regardless of the number of patches generated. Once trained, the Generator can then be used to generate new patches, some of which are shown in Fig. 2(b). The trained generator takes approximately 10 s to generate 3000 new patches of size 21×21.

Before using these new patches to generate stochastic realizations, we need to verify that both of new and original patches come from the same probability distribution. Visually, the new and original patches have the similar channelized structures. Fig. 2 shows that the patterns do not differ significantly between original and new patches.

Quantitatively, we assume the original patches come from a distribution F_1 and the new patches from a distribution F_2. Both of the two distributions are unspecified. Our goal is to test the null hypothesis

$$H_0 : F_1 = F_2 \tag{4.12}$$

versus the alternative $F_1 \neq F_2$. Székely et al. (2004) uses ϵ-statistic (Székely, 2003) to test equality of multivariate distributions. The ϵ-statistic (Székely, 2003) is

$$\epsilon_{n_1,n_2} = \frac{n_1 n_2}{n_1 + n_2}\Big(\frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{m=1}^{n_2}\|\mathbf{p}_i - \tilde{\mathbf{p}}_m\| - \frac{1}{n^2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_1}\left\|\mathbf{p}_i\mathbf{p}_j\right\| \\ - -\frac{1}{n^2}\sum_{l=1}^{n_2}\sum_{m=1}^{n^2}\|\tilde{\mathbf{p}}_l - \tilde{\mathbf{p}}_m\|\Big) \tag{4.13}$$

where n_1 and n_2 are the number of original patches $\mathbf{p}$ and new patches $\tilde{\mathbf{p}}$ respectively. In our case, the ϵ-statistic is 14.1637. Then the permutation test approach (Efron and Tibshirani, 1994) is used to derive the

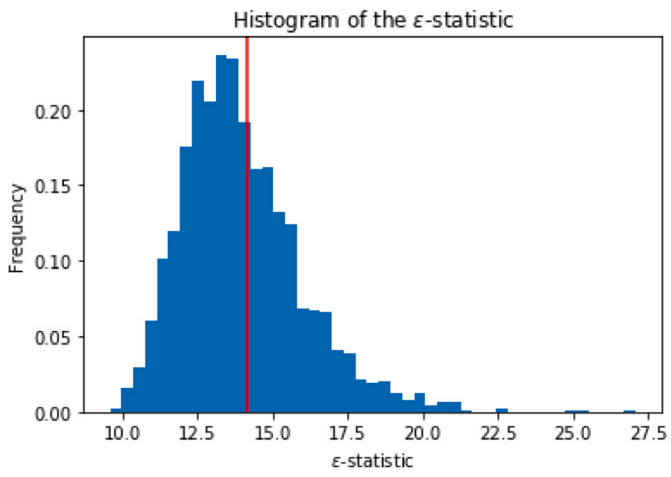


Fig. 3. Histogram of the ϵ-statistic.

distribution of the test statistic under the null hypothesis. Practically, we conduct 2000 resamples to conduct an approximate permutation test. The sampling distribution of the ϵ-statistic is shown in 3. The P-value, represented by the blue area to the right of the vertical red line, is 0.3965, which is large enough (> 0.05) to fail to reject the null hypothesis.

4.3. Hard data conditioning

In this subsection, a comparison with patch-based methods in terms of data conditioning capabilities is conducted. Fig. 4 shows some randomly generated hard data patterns of size 21×21. Given these data patterns, we cannot find from TI any patches that exactly matches these data patterns. To perfectly match these data patterns, we generate new patches using the trained Generator and obtain a certain number of eligible patches using the acceptance-rejection method. This demonstrates that there is a possibility that we cannot find eligible patches from TI, however, a number of patches that exactly match the hard data pattern can be obtained by the trained Generator

The first column in Fig. 4 shows one type of hard data pattern. The second column shows the most similar patch from TI, which is unable to exactly match this hard data pattern, leading to poor hard data reproduction. The percentage of ill-matched nodes is 0.2. Our trained Generator can be used to generate patches that honor the hard data exactly. The last 3 columns show 3 of these patches.

4.4. The problem of verbatim copy

In this section, realizations generated from the enriched pattern database will be compared with the ones obtained from the original pattern database. The comparison is conducted in terms of the occurrence of verbatim copy, which can be quantified with coherence maps. Fig. 5(a) shows the training image of size 101×101 and linearly increasing indices of the location of each node in the training image. Coherence maps can then be constructed by recording these indices during the simulation. Fig. 5(b) shows one realization of 301×301 and its coherence map generated from the original pattern database. Similarly, Fig. 5(c) shows one realization of 301×301 and its coherence map obtained from both the original and GAN-generated patches, in which the white pixels indicate these nodes of the realization come from new patches and not from the training image, limiting the occurrence of verbatim copy.

The occurrence of verbatim copy is partly due to finite size of TI, from which a limited number of patterns are extracted. Generating new patches is equivalent to increasing the size of TI. The greater the number of GAN-generated patches are used, the less chance portions of the TI are exactly copied into the simulated realizations. It is noteworthy that the effect of verbatim copy could be completely avoided under some conditions: (a) discarding all patterns from the original TI and only use GAN-generated patches during the simulation; (b) The GAN-generated patches should be different from any part of the TI, which can be quantified by a distance (or dissimilarity) function. How to properly choose distance functions to meet the condition (b) deserves further study.

4.5. Quantitative assessment

In this section, we will quantitatively compare realizations created from the enriched patch database with the ones produced from the original patches. This comparison is conducted on the basis of two criteria (Tan et al., 2014): (i) the variability between generated realizations, also called a diversity score representing a model of spatial uncertainty and (ii) the variability within the realizations, also termed an inconsistency score depicting how well the statistics of the training image are reproduced. Realizations generated by a better patch-based algorithm could be deemed to have a larger space of uncertainty and a better pattern reproduction. To quantitatively study these two types of variabilities, we use the distance function, as defined in Tan et al. (2014) to provide a measure of similarity between any two images

$$d(l_i, l_j) = \mathbf{JS}(\mathbf{MPH}(l_i), \mathbf{MPH}(l_j)) \tag{4.14}$$

where $\mathbf{MPH}(l)$ represents the multiple-point histograms (Deutsch and Gringarten, 2000) of patches extracted from the image l with the template size of 4×4; **JS** stands for the Jensen–Shannon divergence (Endres and Schindelin, 2003) measuring the similarity between two MPHs. Therefore, the variability between any two of N realizations can be represented as,

$$D_{between}(l_1, \dots, l_N) = \frac{1}{N(N-1)} \Sigma_{i=1}^{N} \Sigma_{j=1}^{N} d(l_i, l_j) \tag{4.15}$$

Similarly, the variability between the N realizations and the training image **ti** can be computed as,

$$D_{within}(l_1, \dots, l_N, \mathbf{ti}) = \frac{1}{N} \Sigma_{i=1}^{N}\ d(l_i, t) \tag{4.16}$$

Then, the ratio of the above two distances is used to rank the algorithms,

$$Ratio(l_1, \dots, l_N, \mathbf{ti}) = \frac{D_{between}(l_1, \dots, l_N,)}{D_{within}(l_1, \dots, l_N, \mathbf{ti})} \tag{4.17}$$

We start with the training image shown in Fig. 6(a) and generate two sets of 50 realizations of size 101×101: one from the enriched pattern database denoted as $\mathbb{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_{50}\}$, and the other from the original pattern database denoted as $\mathbb{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_{50}\}$. One realization with each method is shown in Fig. 6(b) and 6(c).

Then the two types of variabilities can be calculated based on (4.15) and (4.16), which are summarized in Table 3.

The first row of the Table 3 shows that our proposed method is better in terms of the spatial uncertainty (by representing a larger distance of 1 compared to 0.9058 for the original method). This result is consistent with the expectation that enriched pattern database is able to suppress the occurrence of verbatim copy (see Section 4.4), which is inversely proportional to the variability between realizations.

In the second row, the original approach represents a slightly shorter distance of 0.9823 compared to 1 for the proposed one, indicating that both of their performance in reproducing statistics of the training image is quite similar. This result is also consistent with the expectation. A possible reason could be that both of new and original patterns come from the same probability distribution (see Section 4.2). Therefore, the performance of both two methods in reproducing statistics of the training image is quite similar.

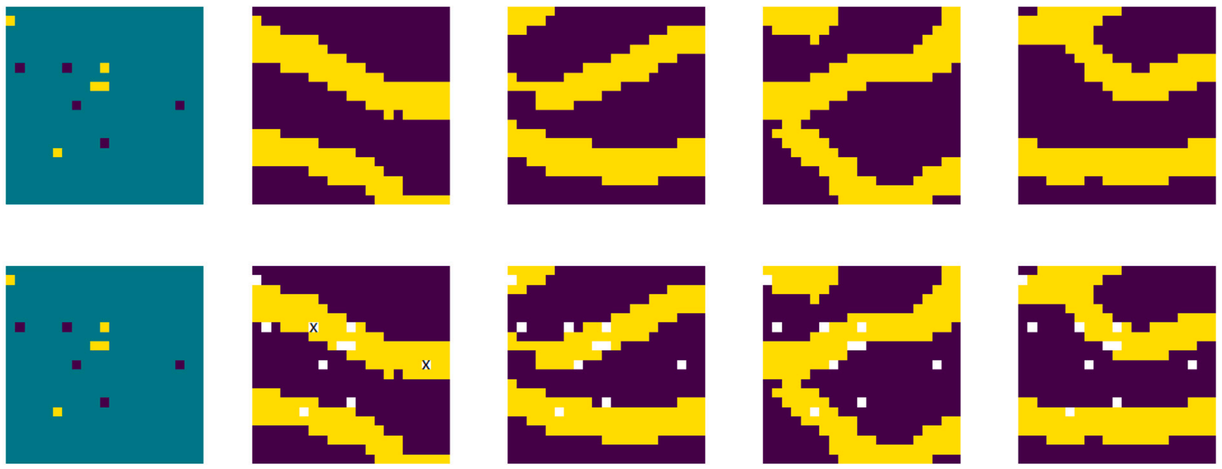


Fig. 4. patches matching the hard data pattern: (i) column 1 shows the hard data pattern which is generated outside the GAN.; (ii) column 2 shows the most similar patch from TI; (iii) Column 3 – 5 shows 3 patches obtained from our generator match the hard data pattern exactly.

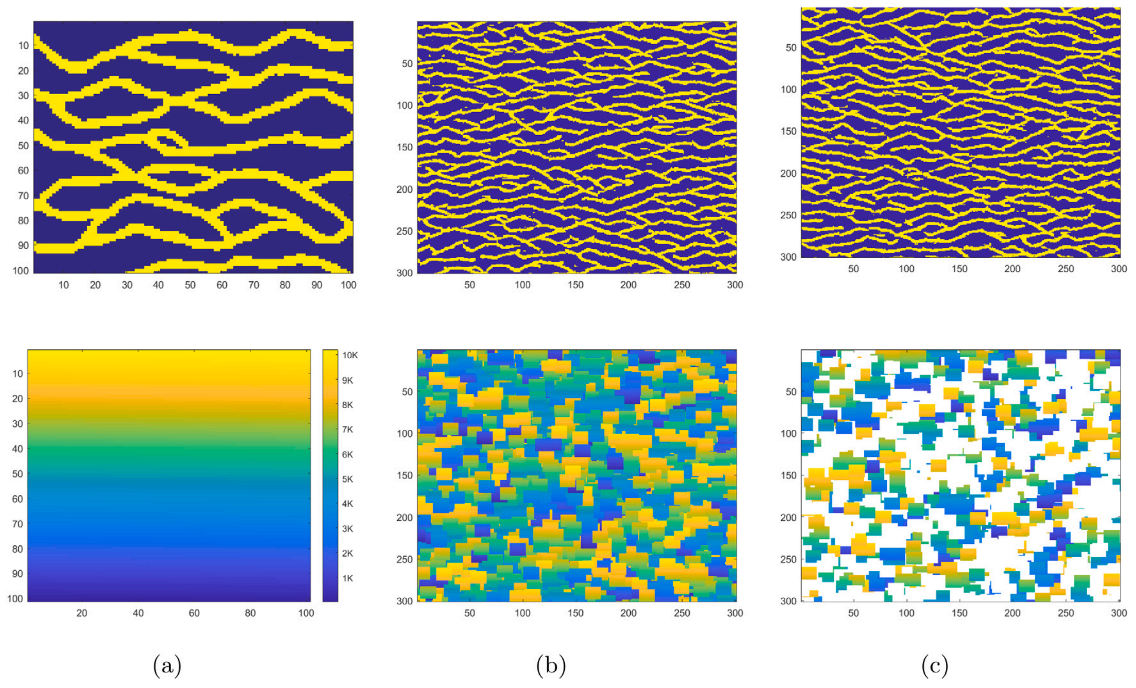


Fig. 5. Illustration of coherence maps for two different cases. Both cases use patches of size 21 × 21. Realizations of 301 × 301 generated with (b) 1681 original patches; (c) 1681 original and 3200 GAN-generated patches.

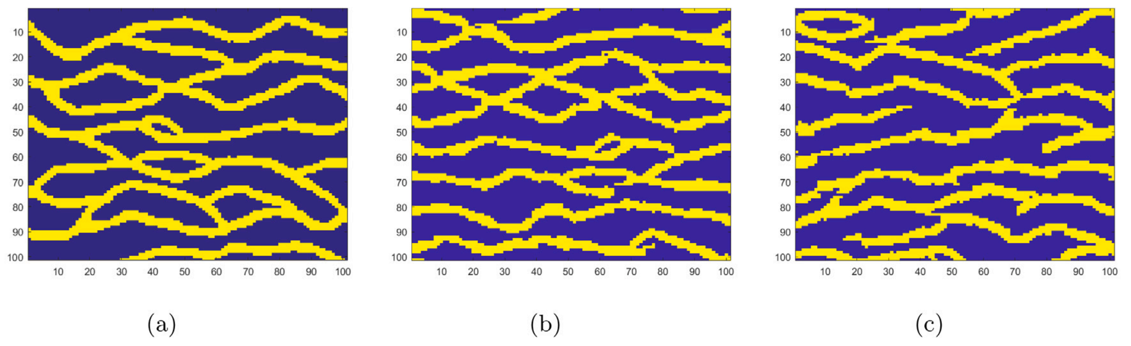


Fig. 6. (a) training image of size 101 × 101; (b) one realization of size 101 × 101 generated with 1681 original patches (original method); (c) one realization generated with 1681 original and 3200 GAN-generated patches (proposed method); Both cases use patches of size 21 × 21.

Table 3
Ratios of the within and between realization variability and total ratio for the original method (using original patch database $\mathbb{Y}$) and our proposed one (using enriched patch database $\mathbb{X}$).

Between-realization variability	$D_{between}(\mathbb{Y}) : D_{between}(\mathbb{X},)$	0.9058 : 1.0000
Within-realization variability	$D_{within}(\mathbb{Y}, \mathbf{ti}) : D_{within}(\mathbb{X}, \mathbf{ti})$	0.9823 : 1.0000
Ratio (between/within)	$Ratio(\mathbb{Y}, \mathbf{ti}) : Ratio(\mathbb{X}, \mathbf{ti})$	0.9221 : 1.0000

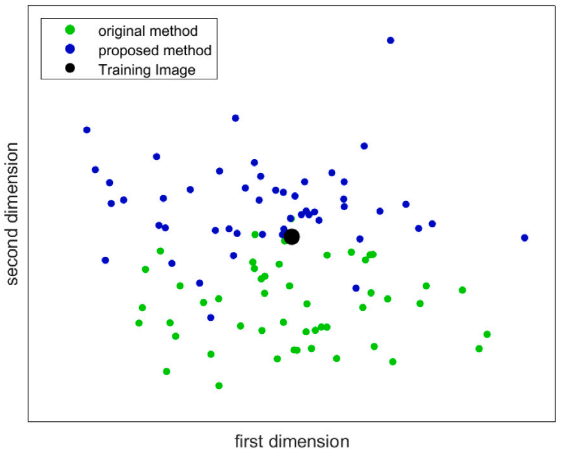


Fig. 7. two dimensional representation of realizations and TI using MDS.

The last row of the Table 3 shows that our proposed method performs better than the original one with a quantitative rank of $1 : 0.9221$ (proposed method : original method). This result is as expected on the basis of the above two rows.

Having computing the ratio of these two counteracting distances used to compare our proposed method and the original one, we use Multidimensional scaling (MDS) to visualize these distances.

Fig. 7 shows the two most important coordinates of our realizations and training image. Even though 2D projection of our data may result in loss of information, it is still a good way of visualizing the relative positions of them. It can be seen from Fig. 7: (1) a wider spread of (blue) points for our proposed method, indicating a larger space of uncertainty and (2) the performance of both methods in reproducing statistics of the TI (black dot) is visually no obvious difference. These two observations are consistent with the results shown in Table 3

5. Discussion and conclusion

In this paper, we propose a novel approach to improve patch-based algorithms by addressing the problem of conditioning to point data and the occurrence of verbatim copy. The results show that we can generate new patches which have the same probability distribution as the original patches. We also show that given some hard data patterns no such patch from TI can be found but a number of matching patches can be found with our method. Lastly, we also show that our method can ameliorate the verbatim copy problem by generating new patches.

The ultimate goal is always to generate “good models”. One must acknowledge that “good models” cannot be dealt with in a wholly objective way. We will need to choose some measure of “good models” and such selection is often problem-specific and can be questioned. In our paper the measure of “good models” is based on two questions below: (a) whether the proposed method, compared to its corresponding patch-based algorithm, has a higher point-data conditioning capability; (b) whether the proposed approach is able to alleviate the effect of verbatim copy in comparison with the original patch-based algorithm. The experimental results indicate that models generated by our method are better than its patch-based counterpart in terms of the two above issues.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Arpat, G. Burc, Caers, Jef, 2007. Conditional simulation with patterns. Math. Geol. 39 (2), 177–203.
Ashby, F. Gregory, Ennis, Daniel M., 2007. Similarity measures. Scholarpedia 2 (12), 4116.
Caers, Jef, Zhang, Tuanfeng, 2004. Multiple-Point Geostatistics: a Quantitative Vehicle for Integrating Geologic Analogs into Multiple Reservoir Models. AAPG Special Volumes.
Chan, Shing, Elsheikh, Ahmed H., 2017. Parametrization and generation of geological models with generative adversarial networks. arXiv preprint arXiv:1708.01810.
Chatterjee, Snehamoy, Dimitrakopoulos, Roussos, Mustapha, Hussein, 2012. Dimensional reduction of pattern-based simulation using wavelet analysis. Math. Geosci. 44 (3), 343–374.
Chen, Mei, Wu, Shenghe, Bedle, Heather, Xie, Pengfei, Zhang, Jiajia, Wang, Yunlong, 2022. Modeling of subsurface sedimentary facies using Self-Attention Generative Adversarial Networks (SAGANs). J. Pet. Sci. Eng. 214, 110470.
Deutsch, C.V., Gringarten, E., 2000. Accounting for multiple-point continuity in geostatistical modeling. In: 6th International Geostatistics Congress, Vol. 1. pp. 156–165.
Dupont, Emilien, Zhang, Tuanfeng, Tilke, Peter, Liang, Lin, Bailey, William, 2018. Generating realistic geology conditioned on physical measurements with generative adversarial networks. arXiv preprint arXiv:1802.03065.
Efron, Bradley, Tibshirani, Robert J., 1994. An Introduction to the Bootstrap. CRC Press.
El Ouassini, Ayoub, Saucier, Antoine, Marcotte, Denis, Favis, Basil D, 2008. A patchwork approach to stochastic simulation: a route towards the analysis of morphology in multiphase systems. Chaos Solitons Fractals 36 (2), 418–436.
Emery, Xavier, Lantuéjoul, Christian, 2014. Can a training image be a substitute for a random field model? Math. Geosci. 46 (2), 133–147.
Endres, Dominik Maria, Schindelin, Johannes E., 2003. A new metric for probability distributions. IEEE Trans. Inform. Theory 49 (7), 1858–1860.
Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, Bengio, Yoshua, 2014a. Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680.
Goodfellow, Ian J., Shlens, Jonathon, Szegedy, Christian, 2014b. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
Guardiano, Felipe B., Srivastava, R. Mohan, 1993. Multivariate geostatistics: beyond bivariate moments. In: Geostatistics Troia’92. Springer, pp. 133–144.
Honarkhah, Mehrdad, Caers, Jef, 2010. Stochastic simulation of patterns using distance-based pattern modeling. Math. Geosci. 42 (5), 487–517.
Hu, L.Y., Chugunova, T., 2008. Multiple-point geostatistics for modeling subsurface heterogeneity: A comprehensive review. Water Resour. Res. 44 (11).
Jetchev, Nikolay, Bergmann, Urs, Vollgraf, Roland, 2016. Texture synthesis with spatial generative adversarial networks. arXiv preprint arXiv:1611.08207.
Laloy, Eric, Hérault, Romain, Jacques, Diederik, Linde, Niklas, 2018. Training-image based geostatistical inversion using a spatial generative adversarial neural network. Water Resour. Res. 54 (1), 381–406.
Mariethoz, Gregoire, 2018. When should we use multiple-point geostatistics? In: Handbook of Mathematical Geosciences. Springer, pp. 645–653.
Mariethoz, Gregoire, Caers, Jef, 2014. Multiple-Point Geostatistics: Stochastic Modeling with Training Images. John Wiley & Sons.
Mariethoz, Gregoire, Lefebvre, Sylvain, 2014. Bridges between multiple-point geostatistics and texture synthesis: Review and guidelines for future research. Comput. Geosci. 66, 66–80.
Mariethoz, Gregoire, Renard, Philippe, Straubhaar, Julien, 2010. The direct sampling method to perform multiple-point geostatistical simulations. Water Resour. Res. 46 (11).
Mosser, Lukas, Dubrule, Olivier, Blunt, Martin J., 2017. Reconstruction of three-dimensional porous media using generative adversarial neural networks. Phys. Rev. E 96 (4), 043309.
Pan, Wen, Torres-Verdín, Carlos, Pyrcz, Michael J., 2021. Stochastic pix2pix: a new machine learning method for geophysical and well conditioning of rule-based channel reservoir models. Nat. Resour. Res. 30 (2), 1319–1345.
Peredo, Oscar, Ortiz, Julián M., 2011. Parallel implementation of simulated annealing to reproduce multiple-point statistics. Comput. Geosci. 37 (8), 1110–1121.

Radford, Alec, Metz, Luke, Chintala, Soumith, 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Rezaee, Hassan, Marcotte, Denis, Tahmasebi, Pejman, Saucier, Antoine, 2015. Multiple-point geostatistical simulation using enriched pattern databases. Stoch. Environ. Res. Risk Assess. 29 (3), 893–913.

Rezaee, Hassan, Mariethoz, Gregoire, Koneshloo, Mohammad, Asghari, Omid, 2013. Multiple-point geostatistical simulation using the bunch-pasting direct sampling method. Comput. Geosci. 54, 293–308.

Song, Suihong, Mukerji, Tapan, Hou, Jiagen, 2021. Geological facies modeling based on progressive growing of generative adversarial networks (GANs). Comput. Geosci. 25 (3), 1251–1273.

Strebelle, Sebastien, 2002. Conditional simulation of complex geological structures using multiple-point statistics. Math. Geol. 34 (1), 1–21.

Székely, Gábor J., 2003. E-Statistics: The energy of statistical samples, Vol. 3, No. 05. Bowling Green State University, Department of Mathematics and Statistics Technical Report, 3, (05), pp. 1–18.

Székely, Gábor J., Rizzo, Maria L., et al., 2004. Testing for equal distributions in high dimension. InterStat 5 (16.10), 1249–1272.

Tahmasebi, Pejman, Hezarkhani, Ardeshir, Sahimi, Muhammad, 2012. Multiple-point geostatistical modeling based on the cross-correlation functions. Comput. Geosci. 16 (3), 779–797.

Tan, Xiaojin, Tahmasebi, Pejman, Caers, Jef, 2014. Comparing training-image based algorithms using an analysis of distance. Math. Geosci. 46, 149–169.

Wei, Li-Yi, Lefebvre, Sylvain, Kwatra, Vivek, Turk, Greg, 2009. State of the art in example-based texture synthesis. In: Eurographics 2009, State of the Art Report, EG-STAR. Eurographics Association, pp. 93–117.

Zhang, Tuanfeng, Switzer, Paul, Journel, Andre, 2006. Filter-based classification of training image patterns for spatial simulation. Math. Geol. 38 (1), 63–80.