TÉCNICO LISBOA

Geostatistical Subsurface Modeling Using Generative Adversarial Networks

Arthur Henrique Rodrigues Santos

Thesis to obtain the Master of Science Degree in

Petroleum Engineering

Supervisors: Prof. Leonardo Azevedo Guerra Raposo Pereira

Prof. Gustavo André Paneiro

Examination Committee

Chairperson: Profª. Maria João Correia Colunas Pereira

Supervisor: Prof. Leonardo Azevedo Guerra Raposo Pereira

Members of the Committee: Prof. Amílcar de Oliveira Soares

July, 2019

This page was intentionally left blank.

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

This page was intentionally left blank.

Abstract

Generative Adversarial Network (GAN) is a model that considers two distinct deep neural networks, i. e. a discriminator and a generator. The GAN, after trained, is capable of generating samples that reproduce the data distribution, including spatial distribution, of the training dataset. In geosciences, GAN's have been successfully applied to generate unconditional realizations of rock properties from geological priors described by training images, and within probabilistic seismic inversion methods. Here, the use of generative adversarial networks is proposed not as a model generator but as model reconstruction technique for subsurface models where we do have access to sparse measurements of the subsurface properties of interest. Sets of geostatistical realizations as training datasets combined with observed experimental data. These networks are applied to reconstruct non-stationary sedimentary channels and continuous elastic properties, such as P-wave propagation velocity, in the presence and absence of conditioning data. The application examples show the suitability of generative adversarial networks in learning the spatial structure of the data from sets of stochastic realizations, reproducing the original main statistics of first and second order and the spatial continuity pattern as expressed by a variogram model (i.e. a spatial covariance matrix). This thesis also explores the application of GAN's in a seismic inversion algorithm in attempt to incorporate non-stationary geological structures by a performing a simultaneous inversion of the seismic data to facies and acoustic impedance models. This mechanism will enable to produce reservoir models more geologically consistent and reliable.

Keywords: Generative Adversarial Networks, Subsurface Model, Reservoir Modeling, Seismic Inversion

This page was intentionally left blank.

Resumo

A *Rede Generativa Adversária* (GAN) é um modelo de rede composta de duas redes neuronais com múltiplas camadas escondidas (*deep neural networks)*, que apresentam caráter distinto, i. e. discriminativa e geradora. A GAN, após treinada, será capaz de gerar amostras que reproduzem a distribuição de dados, inclusive espacialmente, a partir de um conjunto de treino. Em geociências, o uso GAN's para geração não condicionada de modelos de propriedades de rocha por meio de imagens de treino, e em métodos probabilísticos de inversão sísmica. Nesse trabalho, a GAN é proposta como uma técnica de reprodução de modelos de subsuperfície, onde há acesso uma escassa base de dados das propriedades de interesse. Como dados de treino para a rede, são usadas realizações geoestatísticas e modelos são criados a partir de dados experimentais. A GAN é utilizada para reconstruir canais não estacionários e propriedades elásticas contínuas, na presença de dados condicionantes ou não. Os exemplos apresentados mostram a capacidade da GAN em aprender a estrutura espacial dos dados a partir de realizações estocásticas, reproduzindo as principais estatísticas de primeira e segunda ordem e os padrões de continuidade espacial, como expressado por modelos de variograma (i. e. matriz de covariância espacial). Também é explorado o uso da GAN no método de inversão sísmica na tentativa de reproduzir estruturas não-estacionárias por meio de uma inversão simultânea dos dados sísmicos para modelos de fácies e impedâncias acústicas. Tal método permitirá produzir modelos de reservatórios geologicamente mais consistentes e confiáveis.

Palavras-chave: Redes Generativas Adversárias, Modelos de Subsuperfície, Modelação de Reservatórios, Inversão Sísmica.

This page was intentionally left blank.

Table of Contents

List of Figures

List of Acronym

ANN	Artificial Neural Network
BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
DSS	Direct Sequential Simulation
GAN	Generative Adversarial Network
GSI	Global Stochastic Inversion
MPS	Multi-Point Statistics
ReLU	Rectified Linear Units
Tanh	Hyperbolic Tangent
WGAN	Wasserstein Generative Adversarial Network

This page was intentionally left blank.

1 Introduction

1.1 Motivation

The seismic reflection method is a technique used to study the earth´s subsurface. A seismic wave is propagated in the earth´s subsurface and when the wave reaches an interface between two media with different elastic properties, the wave is reflected back until the surface, where its amplitude and arrival time, given the typical relatively low frequencies, are recorded in specific sensors like geophones and hydrophones (Yilmaz, 2001). However, besides geological structure information, there is no other information that can be directly incorporated from the seismic data into reservoir models. Therefore, it is necessary to extract rock property information that is not directly present at the seismic recording, but related to the seismic response of the subsurface, like the elastic properties (Caetano, 2012).

The study of a phenomenon or a physical system is often related to the attempt to be described as a model, which is determined by a function and its parameters. The inverse problem is set when the output of the model is known, but it is necessary to find the configuration of model parameters that best fits the observed measurements (Tarantola, 2005).

To infer the spatial distribution of the subsurface petro-elastic properties from indirect measurements such as the recorded seismic section an inverse problem is set. The process of solving this inverse problem is called seismic inversion. There are two main approaches to do a seismic inversion: the deterministic, and a statistical approach. However, due to the complexity of the seismic reflection phenomena, mainly issues related to the time variant acoustic wavefield propagation, the earth´s reflectivity, noise content, earth´s filtering effect, scattering, mode conversion, it is often not possible to use a simple and deterministic approach to tackle this kind of inverse problem (Tarantola, 2005).

The inversion methods using a statistical approach enables the assess of uncertainties of the inverted models. The focus of this work is to study a particular method of the statistical approaches, which are the iterative geostatistical seismic inversions (Azevedo & Soares, 2017). To do a reliable geostatistical seismic inversion, it is necessary to have well information of the acoustic impedance enabling to compute a synthetic seismic trace for the well and compare it with the measured seismic data. In this way, it is possible to adjust the misfit between synthetic and real seismic traces by adjusting the model parameters (acoustic impedance values). These parameters can be correlated to the available well data enabling to minimize the misfit of the inverted model and creating more reliable model due to the possibility of integration of seismic data and well data (Mosser et al. 2018b).

One of the challenges of geostatistical seismic inversion is the incorporation of non-stationary sedimentary features within the inversion framework. Elastic properties, that most influence the seismic response of the subsurface are highly correlated to the sedimentary facies. However, these structures are often complex and non-stationary and their explicit inclusion within geostatistical seismic inversion, such as Global

Stochastic Inversion (GSI) algorithm (Soares et al., 2007), is not available. Therefore, an improvement to be done in the algorithm is the introduction of geological features that can enable a more reliable inversion process due to the correlation that petrophysical properties present with lithofacies.

Generative models are designed with the objective of reproducing a given data distribution. Generative Adversarial Networks (GANs) are a kind of generative models that consists in an adversarial training between two networks in a zero-sum game framework, i. e. both discriminative and generative networks, which have been shown to be able to parameterize and generate complex data distribution. Since its first formulation by (Goodfellow et al., 2014) several improvements have been made to increase GAN's performance, like the use of convolutional networks (Radford et al. 2015) and development of new cost functions as addressed by Wasserstein GAN (WGAN) (Arjovsky et al. 2017).

With the known successful application of GAN's in previous geosciences applications (e.g. Chan & Elsheikh, 2017; Mosser et al. 2018a; Mosser et al., 2018b), it is expected that the GAN can be implemented in the geostatistical seismic inversion workflow in an attempt to reproduce the non-stationary features and resulting in a more reliable inverted models.

1.2 Objective

This Master thesis proposes the use of GANs as method to generate subsurface models. The network is trained with a set of geostatistical realizations. The performance of the network is tested in different scenarios for the generation of conditioned models, using the network as a model reconstruction technique. Later, it is proposed a seismic inversion method coupling geostatistical simulation, co-simulation tools and generative adversarial networks (Goodfellow et al., 2014). This approach uses a trained artificial neural network as a generative model to reproduce non-stationary sedimentary features which are updated based on the mismatch between real and synthetic seismic data.

The main goals are:

i) Develop a deep artificial neural network, based on generative adversarial networks, able to generate discrete and continuous models;
ii) Evaluate the performance of the deep artificial neural network as a model reconstruction technique;
iii) Develop a deep artificial neural network, based on generative adversarial networks, able to generate lithofacies models;
iv) Couple the developed generative model within geostatistical seismic inversion for facies generation;
v) Implement the seismic inversion procedure developed in iv) to an application example.

1.3 Thesis Outline

This document is organized in six chapters, starting with an introduction of the problem, followed by the main theoretical background related to the proposed methodology and ends with the overview of the proposed generative adversarial geostatistical seismic inversion methodology.

The first chapter presents the objective and the motivation of this Master thesis and the structure of this document.

The second chapter presents the theory background related to the proposed methodology of this research. It is divided into two main parts. The first part addresses the seismic inversion methods, providing information about its purpose and most known methods. The second part depicts the principles of artificial neural networks, particularly deep artificial neural networks and its recent applications in geophysics.

The third chapter describes in detail the proposed methodology, coupling geostatistical inversion and generative adversarial networks. A brief description of the applied deep neural network used and the adjustment of its parameters is also presented.

The fourth chapter shows the obtained results, according to the progress around understanding the GAN methods and the generation of conditioned samples. First, some exploratory tests are introduced for illustration purposes. Then, the use of GANs as a method of model reconstruction is explored. Finally, the implementation of the GAN in the GSI algorithm is performed as an attempt to reproduce the non-stationary presented in a seismic dataset.

The fifth chapter concludes this thesis highlighting the main advantages and disadvantages of the method and future work that can be developed.

The sixth chapter contains the bibliographic references used for the development of this thesis.

This page was intentionally left blank.

2 Theoretical Background

2.1 Seismic Method

The seismic method is a conventional way to acquire indirect information about the earth´s subsurface by recording the seismic response of the subsurface to a propagated seismic wave. A branch of the seismic method is the exploration seismology, which is related to applications for exploration and production of oil fields (Yilmaz, 2001).

When a seismic wave is propagated in the subsurface and reaches an interface between two media that present different impedances, two main phenomena can occur: the reflection and the refraction of the wave. The seismic reflection technique studies only the reflected waves. The reflected energy returns to the surface as a reflected seismic wave and its amplitude and arrival time are recorded by seismic recorders. The amplitude of the reflected seismic wave is related to the properties of the earth´s subsurface, mainly to the difference of two adjacent lithologies, and the arrival time is related, not only to rock properties like density, body wave propagation velocity, porosity, but also to the formation depth. The seismic response, $S(t)$, can be described by a convolution between the propagated seismic wave, $w(t)$, and earth´s reflectivity, $r(t)$, plus a noise (or error) term $e(t)$, as seen in Equation (1) (Russell, 1988)

$$S(t) = w(t) * r(t) + e(t). \quad (1)$$

Earth´s reflectivity, $r(t)$, is the set of reflection coefficients of each subsurface interface.The reflection of the waves occur when the wave encounters a contrast in of acoustic impedance Equation (2), as can be seen in Figure 1, part of the incident wave is reflected and part is refracted to the second layer at the boundary of the layers.

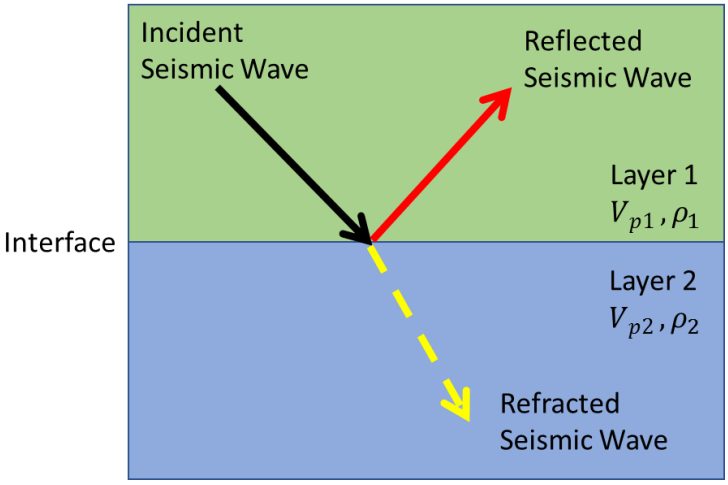

Figure 1. An incident seismic wave reaches the boundary of layers that present different acoustic impedance. Part of the wave is reflected and part is refracted. The amplitude and arrival time of the reflected portion of the wave is recorded by the geophones or hydrophones.

For normal incidence, the reflection coefficients, Equation (3) , are resulting in the contrast between acoustic impedance layers (Russell, 1988):

$$AI = V_p \, x \, \rho, \tag{2}$$

$$r = \frac{AI_2 - AI_1}{AI_2 + AI_1}, \tag{3}$$

where V_p is the P-wave propagation velocity and ρ is the density of the rock propagation medium and the indexes 1 and 2 represent the media above and below the interface.

2.1.1 Seismic Inversion Methods

Although seismic data is highly important for the determination of the reservoir structure and fracture identification, it is hard to create a reservoir model that relies only on seismic data. Reservoir models are normally created from properties measured at existing wells. However, it has been shown that the creation of reservoir models that integrates seismic data and properties measured inside wells, like porosity, permeability and saturation, results in an improvement of the model. However, the acquired seismic data cannot be directly related to well data (Caetano, 2012).

Seismic inversion is a process that enables retrieving the earth´s subsurface properties from acquired seismic data within a high spatial extent. After obtaining this property value from the seismic data it is possible to correlate them to the available well data. The important fact is that seismic inversion is able to provide property information in other regions than the well location, helping the understanding of a larger area of the reservoir which results in more reliable geophysical models for reservoir characterization (Azevedo & Soares, 2017).

Seismic amplitudes recorded during a seismic acquisition are related to the reflection at the interface between distinct lithological layers. However, for reservoir models, it is more useful information about the lithology itself, not its interface. Since the reflection coefficient is related to the difference of elastic properties between lithologies, the acoustic impedance information, a property of the lithology rather than interface, can be retrieved by an inversion procedure (Caetano, 2012). The seismic inversion method removes the effect of the convolution between the wavelet and the subsurface reflectivity to obtain the model. Expanding the convolutional model (Equation (1), the inversion would retrieve the elastic impedance by generating an initial model, computing the reflection coefficients in view of computing the reflectivity sequence and convolving it with the wavelet to generate a synthetic seismogram and compare it with the real data (Caetano, 2012).

The seismic inversion is not a simple process and there is no deterministic solution, it is an ill-posed problem with non-unique solution due seismic method limitations like seismic resolution, noise content (random or coherent), measurement errors, numerical approximations and the intrinsic error to the chosen forward model (Tarantola, 2005).

The objective of the inversion method is to define an Earth´s subsurface model that is forward modeled and produces a synthetic seismogram that has a good correlation with the original seismic data. The match between both data, synthetic and real, can be measured and minimized by defining an objective function that measures the mismatch between both data (Azevedo & Soares, 2017). One important factor is that the obtained elastic model is coherent and reproduces the available geological and petrophysical properties of the reservoir.

The first inversion methods consisted of deterministic approaches aiming to minimize an objective function related to the mismatch between the synthetic seismic, originated by perturbating an initial guess, and the recorded seismic data. Recently, seismic inversion methods rely on a statistical framework to enable uncertainty quantification. From these statistical approaches, the focus of this work is on geostatistical iterative procedures in which the perturbation of the model is done, globally, using stochastic sequential simulations like Direct Sequential Simulation (DSS) and co-simulation (Soares, 2001). These geostatistical simulation methods ensure that a given spatial continuity pattern is reproduced in the inverted elastic models (Azevedo & Soares, 2017).

2.1.2 Global Stochastic Inversion

The Global Stochastic Inversion (GSI) (Soares et al., 2007) is based on two steps: i) The generation of N simulations of acoustic impedance (AI), reproducing the spatial patterns observed in variograms; ii) The computation of synthetic seismogram from these AI simulations and combine the best matches of each model to generate a model with higher correlation. This is done in an iterative way (Caetano, 2012). Figure 2 shows an overview of this inversion method.

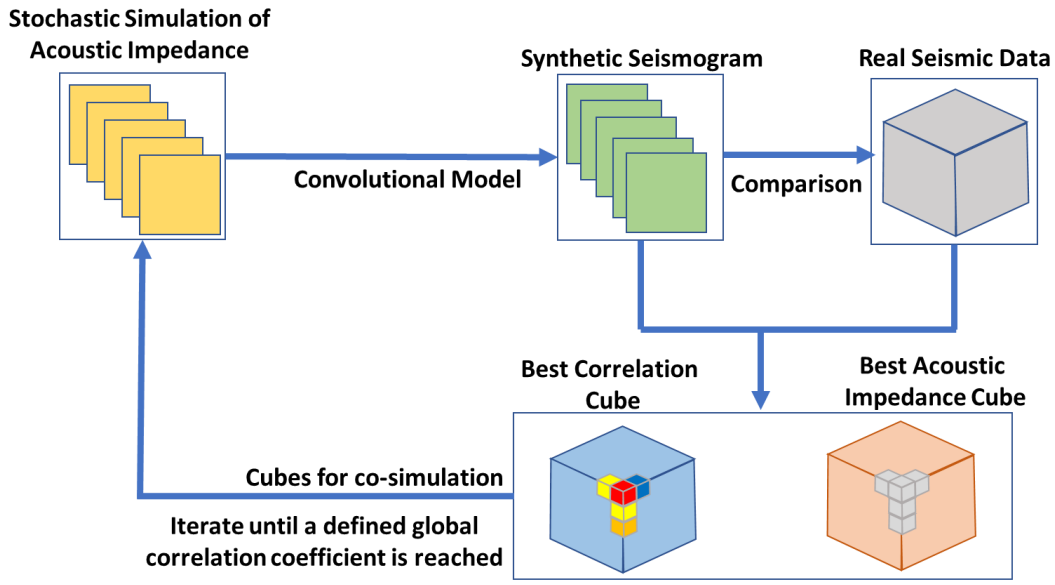

Figure 2. Workflow of the Global Stochastic Acoustic Inversion. A set of acoustic impedance models are simulated, and the synthetic seismic seismograms are computed. The synthetic models are compared to the real seismic data. This comparison method provides the best correlation coefficient and acoustic impedance cubes, to be used in the co-simulation (adapted from Azevedo & Soares, 2017).

First, a set of N acoustic impedance models are simulated using DSS and well data as experimental data and a spatial continuity pattern as described by a variogram model. For each generated model of acoustic impedance the normal incidence reflectivity coefficients are calculated (Equation (2)) and convolved with the wavelet (Equation (1), resulting in N synthetic seismic volumes

Seismic traces are compared trace-by-trace and on a layer basis, partitioning the (synthetic and real) seismic cube in discrete blocks (Figure 3). Correlations between the segments of the N simulations with the real seismic cube are computed and the best segment of each position, with higher correlation, is stored in auxiliary cubes. The value of the acoustic impedance is stored in an acoustic impedance cube and the local correlation value of the segment is stored in a correlation cube (Figure 4) (Azevedo & Soares, 2017; Caetano, 2012).

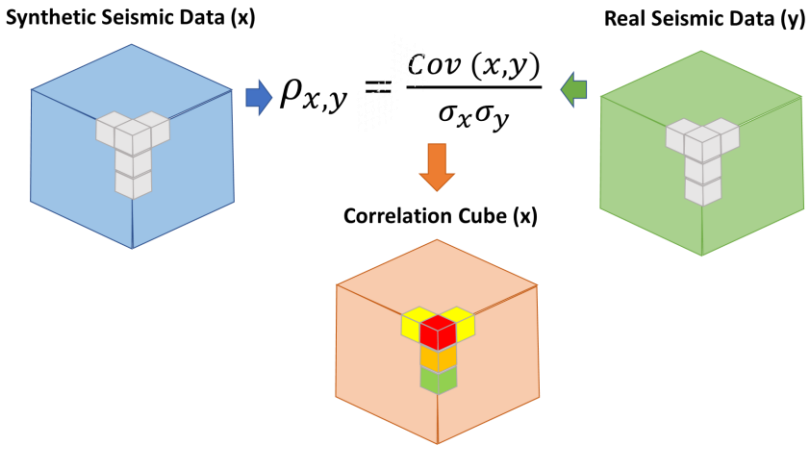

Figure 3. Comparison procedure to create the correlation cubes for each synthetic seismic cube (adapted from Caetano, 2012)

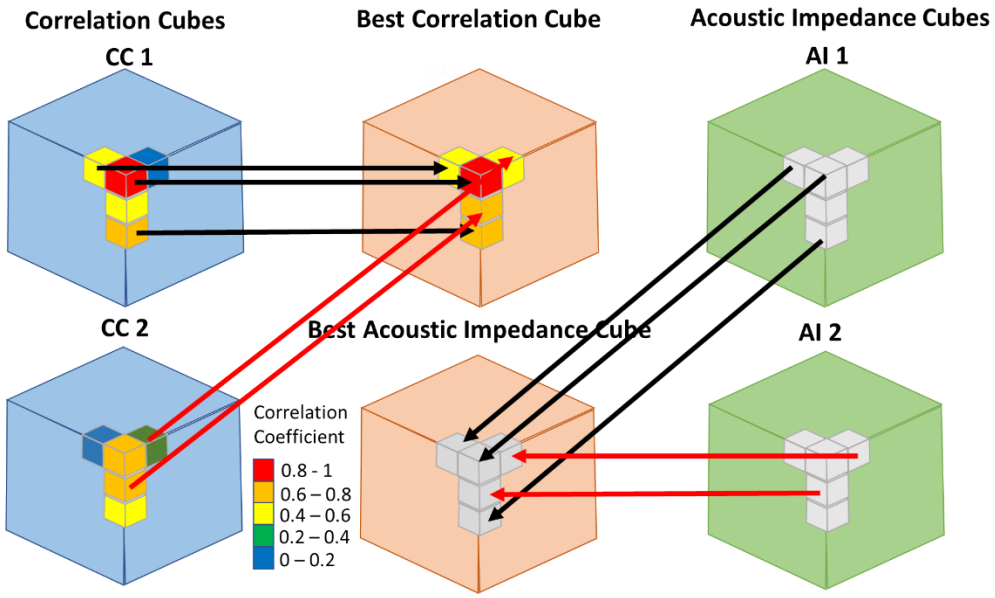

Figure 4. Creation of best correlation and acoustic impedance cubes from $N = 2$ models. For each position of the grid, it is chosen the segment of the correlation cubes that presents higher correlation values. In this case, the color of the arrows, black for cube 1 and red for cube 2, indicates the origin of the segments. Then, the cube best acoustic impedance values is built with the corresponding segments to the ones with higher correlations for each position (adapted from Caetano, 2012).

These stored cubes are then used for the co-simulation of a new set of *N* stochastic realizations of acoustic impedance. The best acoustic impedance and correlation cubes are used as a second variables for the co-simulation. This process is repeated until the overall correlation reaches a defined threshold value (Azevedo & Soares, 2017).

2.2 Artificial Neural Networks

The human brain can accomplish tasks that seem trivial for humans, but that conventional computers cannot perform. In this sense, the brain works like a complex, nonlinear and parallel computer and the key

mechanism of these systems are the neurons. An Artificial Neural Network (ANN) is a system that, in a simpler framework, tries to mimic the way brain works. This means that, through a computation process, it is able to learn and use the acquired knowledge to complete a task. An ANN is, basically, a network of interconnected processing units called neurons, that is able to learn by adjusting the weights (or parameters) of the interconnections between neurons through a learning algorithm (Haykin, 1999).

ANNs, as data-driven approaches, capture the relationship within the data and are able to handle large-scale and complex problems that conventional computing cannot. They are designed to handle nonlinear datasets, where each neuron presents an activation function that is nonlinear, which results in a nonlinear general framework of the ANN. In this sense, it is able to work with nonlinear relationships and this makes the ANN very suitable for complex problems (Haykin, 1999).

Similarly to the human brain, ANN learns by analyzing examples, called training data, of inputs with its respective outputs. In this way of learning, they are able to learn by adjusting the connection weights. The learning process depends on the applied algorithm, but they rely on calculating the difference between the actual output of the network and the expected result. In feedforward backpropagation neural networks, the error is propagated backward in such way that the weights changes and, when the input is fed again to the network, the difference to the output is reduced. This process can also be called optimization since the objective of the training is to minimize the error between the actual and the desired output. Once the network is trained, it works as a function that maps an input to an output. This input-output mechanism allows the network to generalize and calculate outputs for data that were not in the training set (Haykin, 1999). This input and output ability can work as a parameterization method for geophysical data, as seen in Chan & Elsheikh (2017).

A characteristic of ANNs is the adaptivity, where the network can be retrained whenever it is necessary to be adjusted to a new context (Haykin, 1999). Another factor is the evidential response of an ANN, besides providing the output, the network also provides the confidence of the resulting output. This mechanism provides quite useful information of the training status of the ANN (Haykin, 1999). During training, the context information is something that can be handled by the network since every neuron reflects the activation state of the global structure of the network. They work like a dependent and connected system, rather than a sum of single processes (Haykin, 1999).

The ANN present a uniformity of analysis and design in its concept. Although the different training algorithms and network architectures, all of them rely on the neuron as a key mechanism. This makes possible to share theories and algorithms between applications. In the end, ANNs are versatile because it can handle different types of data, its application is varied and the knowledge acquired from one application can be transported to another area (Haykin, 1999).

The use of ANN presents some drawbacks. The ANNs is able to map a numerical input to a numerical output through series of calculations that happens between the layers of the network, however, the

process by which the network comes up with a solution cannot be explained. Usually, the ANNs are considered as “black-boxes”, where it provides no clue about how it got the actual output. Another important drawback is in terms of definition of the network architecture and hyperparameter to be used. Usually, these factors are adjusted by trial and error or knowledge from previous applications (Mijwel, 2018).

Other important limitation is that the networks works in a numerical framework, which means that the inputs and outputs of the networks represents numerical values. This is extremely important to have in mind because it is necessary to format the structure of the problem in a way that the network is able to handle (Mijwel, 2018) . for example, an image is represented as a matrix, in which each pixel of the image is represented by a position in the matrix and a respective integer value (representing the color in that point). Not only important to be considered for data input or output, but also for the determination of the cost function to be optimized and minimize the loss of the network´s output.

2.2.1 The Perceptron

The mechanism of an ANN can be described by three main elements: first is the weighted connections between neurons that feedforward the output of one neuron as an input to another neuron; the second is the sum of inputs at a given neuron, since it may receive multiples inputs, in this case, an adder is responsible for summing all the inputs received; the third is the processing of the resulting input by the neuron. This processing is defined by an activation function. The input goes to this function and the output is calculated and sent forward through a connection, restarting the cycle (Haykin, 1999).

A neuron, k, receives a vector $\boldsymbol{I}$, with m elements, that is the product between two vectors $\boldsymbol{x}$ and $\boldsymbol{w}$, where $\boldsymbol{x}, (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_m})$, is the input vector representing the data and the $\boldsymbol{w}$, $(\boldsymbol{w_{k1}}, \boldsymbol{w_{k2}}, \dots, \boldsymbol{w_{km}})$, is a vector that represents the weights of each connection between the input m and the neuron k. Inside the neuron, the elements of the vector $\boldsymbol{I}$ are summed (Equation (4), this mechanism transfers to neuron process only one input u_k.:

$$u_k = \sum_{j=1}^{m} \boldsymbol{x}_j \boldsymbol{w}_{kj}. \tag{4}$$

A bias, b_k, is summed to the output of the adder, u_k. This is done to increase or reduce the input, v_k of the neuron´s activation function, $\varphi(v)$. The bias is a trainable parameter that applies an affine transformation (linear transformations with translations allowed) in u_k. Then, v_k, called activation potential or induced local field, is inputted to the respective activation function $\varphi(v)$ and the output (Equation (5) is fed forward. Figure 5 conceptually represents the processes that happen in a neuron (perceptron) as previously described (Haykin, 1999):

$$y_k = \varphi(b_k + u_k). \quad (5)$$

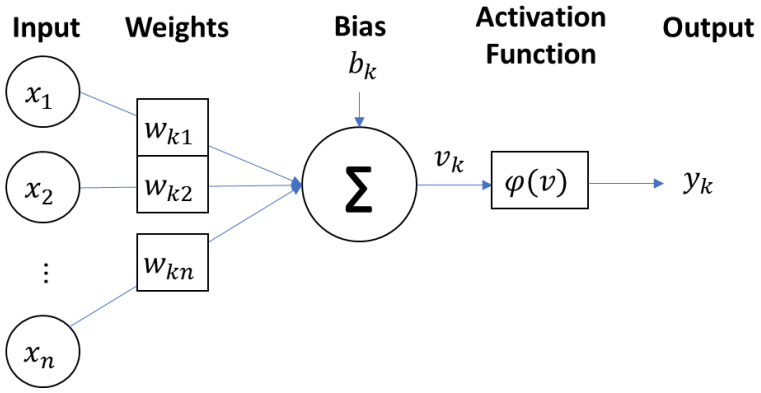

Figure 5. Schematic representation of the perceptron (adapted from Haykin, 1999).

2.2.2 Activation Function

The activation function defines the output of the neuron and, if the activation function is nonlinear, is responsible for adding the nonlinearity behavior to the network. Usually, these nonlinearities are presented in form of sigmoids, hyperbolic tangent or rectified linear units functions. Theoretically, If the structure of the ANN is large enough, even with this limited number of nonlinear functions, the ANN can approximate to any complex function. However, since the networks have a finite number of neurons, its processing and learning power are limited, the choice of the activation function will influence directly the training process and the performance of the network (Agostinelli et al., 2015).

The most basic activation function is the so called threshold function, which is similar to a firing trigger. Given a threshold, the neuron will be activated ($\varphi(v) = 1$) or not ($\varphi(v) = 0$). However, this activation function is not a continuous and differentiable function, and differentiability is an important feature for the backpropagation learning mechanism (Haykin, 1999).

As an option, sigmoid functions can be used. These functions are similar to threshold functions but are continuous and differentiable in the range [0, 1]. The sigmoid is the most used function in ANNs and presents a balance between linear and nonlinear behavior. An example is the logistic function given by Equation (6) (Haykin, 1999):

$$\varphi(v) = \frac{1}{1+\exp(-\alpha v)}, \quad (6)$$

where a determines the slope of the sigmoid function.

Sometimes, it is desirable that the output ranges from -1 to 1. For this purpose, it can be used a hyperbolic tangent (tanh) transfer function. It is similar to the sigmoid function, but the output ranges from -1 to 1 and provides a zero-centered output. Another fact is that, allowing negative values, in some cases, benefits the network training (Haykin, 1999). Figure 6 displays the sigmoid and tanh functions.

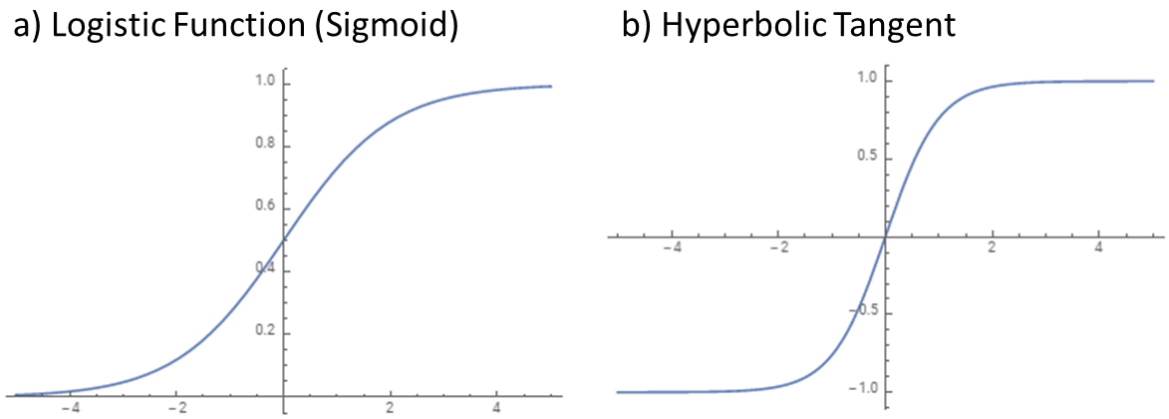

Figure 6. Graphical representation of: a) The sigmoid function bounded from 0 to 1; b) The hyperbolic tangent function, bounded from -1 to 1.

Although the sigmoid and the tanh function are widely used, if the values of the outputs are saturated in 0 or 1 (sigmoid) and -1 or 1 (tanh) the gradients provided by the backpropagation will be close to 0. This problem is known as the vanishing gradient and results in slow convergence of the training. Therefore, a set of functions called Rectified Linear Units (ReLU, Equation (**7**) (Nair & Hinton, 2010) overcome this problem (Maas et al., 2013) :

$$\boldsymbol{\varphi(v)} = \begin{cases} \boldsymbol{v \; if \; v > 0} \\ \boldsymbol{0 \; if \; v \leq 0} \end{cases}. \tag{7}$$

The ReLU function provides a gradient of 1 if $v > 0$ and 0 elsewhere. Initially, as shown in Bengio et al. 2011, the networks present better performance if the neurons are off or operating in a linear regime. This is because the ReLU induces sparsity, which means that neurons can be turned off, real zero outputs, different from sigmoid and tanh. The advantage of using ReLU functions relies on the sparsity it provides to the model and the reduced computational costs related to the linear functions. However, the ReLU functions can present the dying ReLU problem, i.e., similarly to sigmoid functions, the outputs can saturate to zero due to the negative values. Although the gradients of the activated neurons will be passed, this may cause that some neurons are never activated and not respond, generating a passive role. As an alternative, it was developed the Leaky Rectified Linear Unit function (Leaky ReLU) (Maas et al., 2013). This function provides a gradient for negative values inputs by adding a slope at the negative part of the ReLU, as seen in Equation (8) as follows:

$$\varphi(v) = \begin{cases} v \; if \; v > 0 \\ xv \; if \; v \leq 0 \end{cases}, \tag{8}$$

where x is responsible to generate a slight slope at the horizontal part of the ReLU function. Figure 7 exemplifies the difference between ReLU and Leaky ReLU functions.

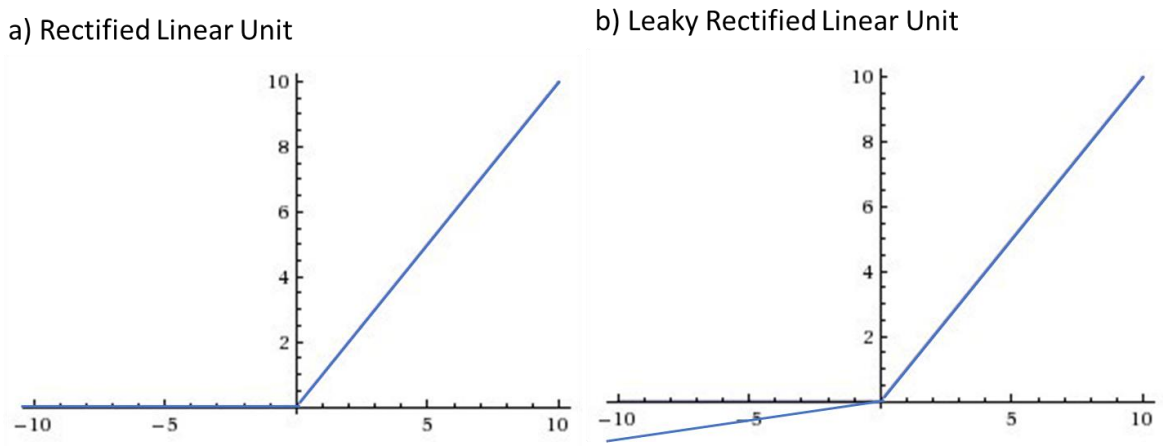

Figure 7. Graphical representation of: a) The Rectified Linear Unit; b) The Leaky Rectified Linear Unit, presenting a slight slope at the negative x axis.

2.2.3 Network Architectures

ANNs structure, represented by layers of neurons, is highly related to the learning algorithm to be used and the capacity of the network. The simplest architecture is the single-layer feedforward network that consists of one input layer directly connected to an output layer of neurons. It is called single because only one layer, the output, computes values (Haykin, 1999).

The other structure is called multilayer feedforward network. This architecture includes more neuron layers between the input and output layers, often called hidden layers. These hidden layers also perform computation and, theoretically, increase the learning capacity of the network and enable to extract higher order statistics from the data (Haykin, 1999). In this case, the input sends data to the first hidden layer, which process the data and its output is inputted to the next layer, following this sequence until the data reaches the output layer. Therefore, the result of the output layer is influenced by all previous layers, meaning that it reflects the activation state of the entire network. Normally, these layers are fully connected, because each neuron of the layer is connected to all neurons of the following layer. However, if a connection is missing, they are called partially connected layers (Haykin, 1999). Figure 8 illustrates these two network architectures.

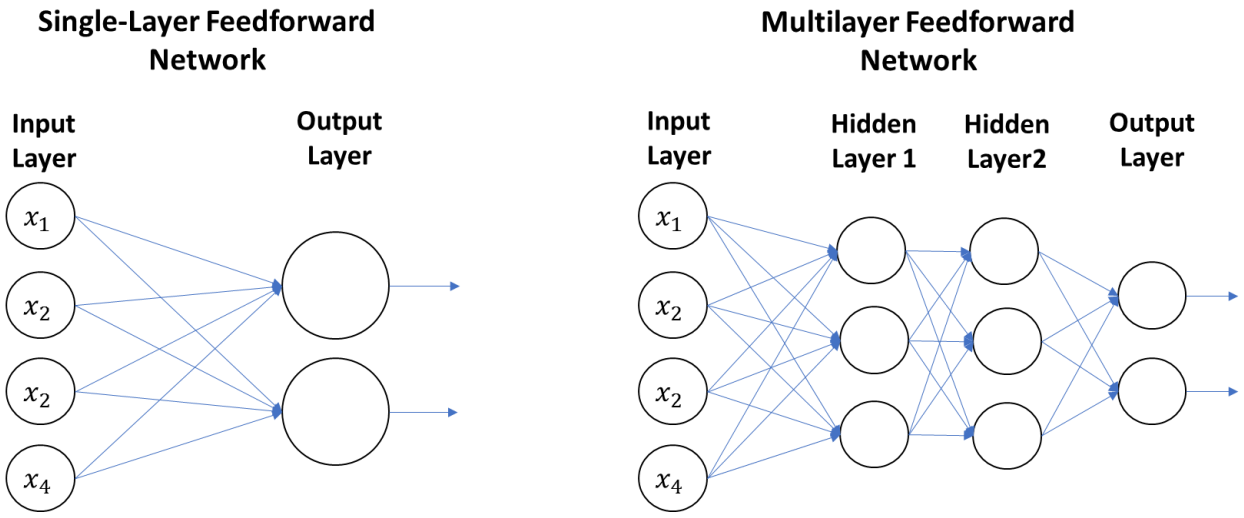

Figure 8. At the left, a scheme of a single-layer feedforward network with two output neurons. At the right, a multilayer feedforward network presenting two hidden layers, with three neurons in each one, and an output layer (adapted from Haykin,1999).

The feedforward networks propagate two kinds of signals. The functional signal, which is the propagation of the input vector through the network and the outputs of the hidden and output layer. The other is the error signal, which is calculated at the output layer and is backpropagated to adjust the synaptic weights in order to the approximate the actual output of the desired response (Haykin, 1999).

2.2.4 Learning Mechanism

ANNs learns through an adjustment of its connection weights and the biases. The learning algorithm determines how these parameters are updated. The basis of the learning mechanism of a multilayer feedforward network is the error-connection learning concept. However, this principle can be better illustrated in a single-layer network. Given an input $\boldsymbol{x}$ to the neuron k, it computes an output y_k. Its output is compared to the desired response or target value d_k. An error, e_k, is computed (Equation (9). This error provides a metric about the actual error and guides the learning mechanism in a way to adjust the synaptic weights, reducing the error e_k, therefore, approximating y_k of the desired response d_k. This adjustment is done in a step-by-step way in which each step is called iteration (Haykin, 1999):

$$e_k = d_k - y_k. \tag{9}$$

This objective is usually described as the minimization of a given cost function. In this way, the updating of the weights is done based on the application of the delta rule or Widrow-Hoff rule (Widrow & Hoff, 1960) . Considering $\boldsymbol{w_{kj}}(i)$ the values of the connection weights, denoting the connection between the output of neuron j and the input of neuron k and the elements of an input vector $\boldsymbol{x_j}(i)$, in a given iteration i. The adjustment $\Delta \boldsymbol{w}_{kj}(i)$ applied to each connection is computed by Equation (10) (Haykin, 1999):

$$\Delta \boldsymbol{w}_{kj}(i) = ne_k(i)\boldsymbol{x}_j(i), \tag{10}$$

where n denotes a positive constant called learning rate. The learning rate is a key parameter for the performance and stability of the network during the learning process and must be tunned carefully (Haykin, 1999). For lower learning rates, it provides a smooth upgrade to the weights of the network. Although the learning process becomes slower, it provides stability to the training (Haykin, 1999).

After computing the adjustments, the weights can be updated given the following rule (Equation (11)):

$$\boldsymbol{w}_{kj}(i+1) = \boldsymbol{w}_{kj}(i) + \Delta\boldsymbol{w}_{kj}(i). \tag{11}$$

The error back-propagation algorithm is the most popular training algorithm for multilayer networks. The algorithm is based on error-connection learning and consists of two passes through the network during one iteration. First, during the forward pass, the input vector is propagated to the network and to the hidden layer and output perform the computation of the values. Since the output is calculated and it can be compared to a target value, an error can be calculated. The backward pass in the network is the backpropagation of the calculated error in a way that the weights are adjusted following the delta rule (Equation (11) (Haykin, 1999).

In ANNs, the error,e, is computed in terms of a cost function C. The backpropagation algorithm computes the derivative of the cost function, C, in relation to the connection weights $\boldsymbol{w}_{kj}$, where j and k denotes two successive layers of the network. This gradient, $\partial C/\partial \boldsymbol{w}_{kj}$, is computed for each neuron in the network. The value of the gradient indicates how the cost function varies with the weights. Once the gradient is computed, it is used to do a gradient descent in the weight space, seeking $\partial \boldsymbol{w}_{kj}$ that minimizes the cost function C. Applying the gradient with the use of the delta rule can be written as Equation (12) (Haykin, 1999):

$$\Delta\boldsymbol{w}_{kj}(i) = -n \frac{\partial C(i)}{\partial \boldsymbol{w}_{kj}(i)}, \tag{12}$$

where the minus signal refers to the gradient descent approach. Equation (12) can be rewritten as:

$$\Delta\boldsymbol{w}_{kj}(i) = n\delta_j(i)y_k(i), \tag{13}$$

where $\delta_j(i)$ is the local gradient, that be described by Equation (14):

$$\delta_j(i) = e_j(i)\varphi'\left(v_j(i)\right). \tag{14}$$

If the j is an hidden neuron, the local gradient is also related to the sum of the δ_s of the k layer not as function of the error, as presented in Equation (15):

$$\delta_j(i) = \varphi'(v_j(i))\sum_k \delta_k(i)\boldsymbol{w}_{kj}(i). \tag{15}$$

The backpropagation algorithm cannot be shown to converge and a usual stopping criterion is evaluating the output of the network. This process is done with a validation dataset, a percentage of the training data

set that is not presented to the training process, which is used to test the performance of the network. As the training data set, the pair input/output of this set is known, inputting this set to the network produces an output that the error can be quantified, and the network performance evaluated (Haykin, 1999).

The network weights can be updated using several methods and one can highlight 3 mainly used:

a. stochastic gradient descent (SGD): the weights of the network are updated after each training sample is presented to the network;
b. batch gradient descent: the weights are updated after the presentations of all training samples. This method results in higher stability for the weights updates, but requires larger computational resources to be performed (Haykin, 1999);
c. mini-batch gradient descent: the weights are updated after a number m of samples, lower than the total samples of training data, is presented to the network. Theoretically, this algorithm results in higher stability than SGD and requires less resources than batch gradient descent.

Some alternatives to the traditional SGD algorithm have developed along the time in aim to decrease the training time of the network. One of the alternatives is to incorporate momentum to the SGD technique, the momentum uses the inertia of the previous iterations to accelerate the training process. The update of the weights using SGD was shown in Equation (11). The iterations of SGD with momentum can be described as (Keskar & Socher, 2017):

$$m(i+1) = \beta m(i) + \Delta \boldsymbol{w}_{kj}(i), \tag{16}$$

$$\boldsymbol{w}_{kj}(i+1) = \boldsymbol{w}_{kj}(i) + m(i+1), \tag{17}$$

where $\beta \in (0,1)$ is the momentum parameter, related to the influence of the previous gradient value to the actual value, and $m(0)$is initialized with the value 0.

The SGD algorithm uses a scalar learning rate to update all the parameters. One alternative is to use adaptive methods, which uses a vector of learning rate that are adapted as the training continues (Keskar & Socher, 2017).

The RMSprop method uses (Tieleman & Hinton, 2012) a combination of mobile average of gradients to scale the learning rate for each parameter. The process in described by Equations (19) and (20). Also, consider $\Delta \boldsymbol{w}_{kj}(i)$ as in Equation (18), without the multiplication by the learning rate.

$$\Delta \boldsymbol{w}_{kj}(i) = -\frac{\partial C(i)}{\partial \boldsymbol{w}_{kj}(i)}, \tag{18}$$

$$g(i+1) = \beta g(i-1) + (1-\beta)\,(\Delta \boldsymbol{w}_{kj}(i))^2, \tag{19}$$

$$\boldsymbol{w}_{kj}(i+1) = \boldsymbol{w}_{kj}(i) + \frac{n}{\sqrt{g(i+1)+\epsilon}} * \Delta \boldsymbol{w}_{kj}(i) \ , \tag{20}$$

where g is the term that corresponds to an average of the gradient and is responsible for the scaling of the learning rate for the update. The term ϵ is used to ensure that there will not occur a division by zero and is normally set to be 1^{-10}.

The Adam algorithm (Kingma & Ba, 2015) emerged as a method to conjugate momentum and scaling of the learning rate parameter using an average of the gradients to input the inertia of previous gradients, Equation (21) and an average of the squared gradients to scale the learning rate, Equation (22). The final update can be seen in Equation (23).

$$m(i+1) = \beta_1 m(i-1) + (1-\beta_1)\,(\Delta \boldsymbol{w}_{kj}(i)), \tag{21}$$

$$g(i+1) = \beta_2 g(i-1) + (1-\beta_2)\,(\Delta \boldsymbol{w}_{kj}(i))^2, \tag{22}$$

$$\boldsymbol{w}_{kj}(i+1) = \boldsymbol{w}_{kj}(i) + \frac{n\,(m(i+1))}{\sqrt{g(i+1)+\epsilon}} * \Delta \boldsymbol{w}_{kj}(i) \ , \tag{23}$$

The Adam algorithm widely used in machine learning tasks due to its performance and its ability to work well despite minimal tuning (Keskar & Socher, 2017).

A common term used during the training of a neural network is the word epochs. An epoch is defined when the whole training dataset is presented to the network. Usually, the network is trained during a pre-determined number of epochs, which means defining the amount of times that your whole dataset will pass through the network. This concept is different from iterations, which means when a sample, or a batch of samples, is presented and the parameters of the networks are updated.

It is worth to mention that, traditionally, like any other nonlinear optimization algorithm, ANN training starts with the initialization of the weights and biases values. The most used initialization procedure assigns random values to ANN parameters leading to multiple local minima of the cost function (Paneiro et al., 2018). To deal with this issue, multistart strategies or meta-heuristics methods (Luke, 2013) can be implemented. Also, according to Radford et al. (2015), one may condition the initial weights by assigning normal distribution with zero-mean and a constant given standard deviation.

2.3 Generative Adversarial Networks

A Generative Adversarial Network is a network configuration that consists in assembling two multilayer perceptron models, a generative network G, and a discriminative network D, that are trained in an adversarial process, in a zero-sum game framework, to estimate a generative model (Goodfellow et al., 2014). In this framework, the networks compete against each other: the G network generates samples in a way that the D network has to distinguish if the generated sample is from the original data or if it is

sampled from the generative model, the aim of the G network is to create samples that are capable of fooling the D network.

Both networks can be represented as differentiable functions: a) the G network is the function $G(\boldsymbol{z}; \theta_G)$, where $\boldsymbol{z}$ is an input noise vector, with a given distribution p_z, and θ_G represents the G network parameters (or weights); b) the D network is the function $D(y;\ \theta_D)$, where y is a sample, from original or synthetic (i.e. generated by G) data, and θ_D represents the D network parameters. The function $G(\boldsymbol{z}; \theta_G)$ maps the input vector to the data domain and the $D(y;\ \theta_D)$, given the input, outputs a single scalar that represents the probability that y came from the original data, and given the output it classifies the image as real (1) or synthetic (0), under the assumption that half of the inputs come from the original data and the other half is synthetic (Goodfellow, 2016a; Goodfellow et al., 2014). A scheme of the adversarial network can be seen in Figure 9.

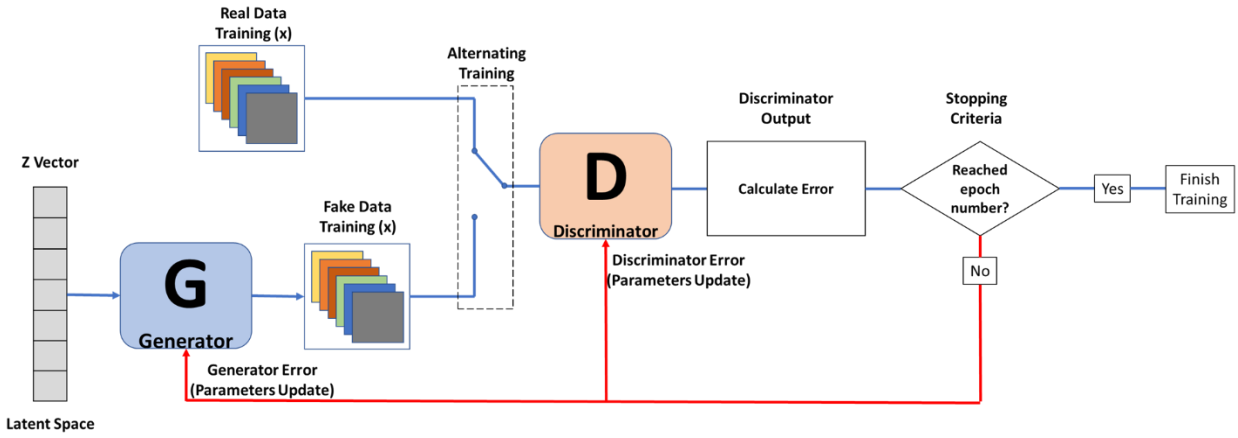

Figure 9. Generative Adversarial Network scheme.

Both networks are simultaneously trained, D is trained aiming to maximize the probability that it labels the input correctly, i.e. if it is from the original data or generated from G. G is trained with the objective to create sample a with a distribution p_g that is similar to the training data distribution p_{data} (Chan & Elsheikh, 2017). During the training, G and D play a minmax game between each other where the value function $V(D, G)$ is described in Equation (24):

$$\min_G \max_D V(D,G) = \left\{\mathbb{E}_{y\sim p_{data}}\, log\, D(y) + \mathbb{E}_{\boldsymbol{z}\sim p_z}\, log\left(1 - D\big(G(\boldsymbol{z})\big)\right)\right\}. \tag{24}$$

This means that the objective of G is to minimize Equation (25), this results that $D(G(\boldsymbol{z}))$ outputs approximately 1, meaning that D identified $G(\boldsymbol{z})$ as real

$$\min_G V(D,G) = \left\{\mathbb{E}_{\boldsymbol{z}\sim p_z} \log\left(1 - D\big(G(\boldsymbol{z})\big)\right)\right\}. \tag{25}$$

However, since that in early learning G did not learn p_{data}, it results that D is able to recognize that the synthetic sample came from G, meaning that Equation (26) saturates and does not provide sufficient

gradient to G to learn because this results in vanishing gradient problem (Goodfellow et al., 2014,Goodfellow, 2016b):

$$\log(1 - D(G(\boldsymbol{z}))) \rightarrow 0. \tag{26}$$

It was proposed that, instead of minimizing $\log(1 - D(G(\boldsymbol{z})))$, the network should be trained to maximize $log\ D(G(\boldsymbol{z}))$, as seen in Equation (27). This change provides stronger gradients during the early steps of the network training (Goodfellow et al., 2014)

$$\max_G V(D,G) = \left\{\mathbb{E}_{\boldsymbol{z}\sim p_z} \log\left(D(G(\boldsymbol{z}))\right)\right\}. \tag{27}$$

On the other hand, the objective of D is to minimize Equation (28). This means that $\log D(y)$ outputs approximately 1, identified original data as real and that $D(G(\boldsymbol{z}))$ outputs approximately 0, meaning that it identifies $G(\boldsymbol{z})$ as synthetic data

$$\max_D V(D,G) = \left\{\mathbb{E}_{y\sim p_{data}} \log D(y) + \mathbb{E}_{\boldsymbol{z}\sim p_z} \log\left(1 - D(G(\boldsymbol{z}))\right)\right\}. \tag{28}$$

In Goodfellow et al. (2014) it is proven that this problem has a global optimum for $p_g = p_{data}$ which can be achieved by the minibatch stochastic gradient descent algorithm optimizing the Equation (24).

In this case, both players present cost functions that are optimized during the training and that can be represented in terms of the standard cross-entropy. The cross-entropy cost for D can be calculated as in Equation (29), which is aimed to be minimized (Goodfellow, 2016b)

$$J_D(\theta_D, \theta_G) = \left\{-\frac{1}{2}\mathbb{E}_{y\sim p_{data}} \log D(y) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z} \log\left(1 - D(G(\boldsymbol{z}))\right)\right\}. \tag{29}$$

Similarly, the cost of G can be written as in Equation (30), which is aimed to be maximized (Goodfellow, 2016b)

$$J_G(\theta_D, \theta_G) = \left\{-\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z} \log\left(1 - D(G(\boldsymbol{z}))\right)\right\}. \tag{30}$$

The training between both networks can be summarized as a zero-sum game: if one wins, the other loses. The solution to this game is when the Nash equilibria is reached, meaning that a local minimum for $G(\boldsymbol{z};\theta_G)$ in terms of θ_G is found and that it is found a local minimum for $D(y;\ \theta_D)$ in terms of θ_D. The Nash equilibrium of this game means that: a) $G(\boldsymbol{z})$ is able to produce samples distribution $p_g = p_{data}$ and b) D is not able to distinguish if the sample is original or synthetic, meaning that $D(x) = \frac{1}{2}$, for any x (Goodfellow, 2016b).

However, Equation (30) would result in vanishing gradient, as seen in Equation (26), so the cost function of G is rewritten as maximization of Equation (31) (Goodfellow, 2016b)

$$J_G(\theta_D, \theta_G) = \left\{ -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z} \log\left(D\left(G(\boldsymbol{z})\right)\right) \right\}. \quad (31)$$

The optimization process can be summarized as following: first, the training data is separated in minibatches containing m examples and is created a minibatch of vectors $\boldsymbol{z}$ containing m examples that, are inputted to $G(\boldsymbol{z})$ and return m outputs in the data space; then, D is trained with both minibatches, trying to identify if it is real or fake. Since it is known which minibatch is from the original data or is synthetic, the parameters θ_D can be optimized using a gradient-based optimization. Then, another minibatch of $\boldsymbol{z}$ containing m examples is sampled These vectors are inputted to $G(\boldsymbol{z})$ and return m outputs in the data space. In this step, the samples are inputted to the D and it provides a feedback to the G if the generated samples were able to fool or not the network, since it is known that these samples were generated by the G.Then, the parameters θ_G are optimized according to the feedback of D. In theory, the equilibrium of this problem is achieved when the D cannot distinguish between data from the training dataset or that are generated by G. However, in practice, this equilibrium is hard to be achieved, therefore, the training is normally limited by defining a number of training epochs.

2.3.1 Deep Convolutional Generative Adversarial Network

Geological models present a fundamental property that is the spatial distribution of the data along a determined grid (2D or 3D). This spatial data distribution implies that each data value is assigned to a specific location along the grid. Therefore, it is essential that the network designed to work with this kind of model is able to incorporate this spatial data distribution. These kinds of data, that present spatial distribution, are normally seen as images (2D or 3D) and a specific kind of neural network was designed to operate with kind of dataset.

At the first formulation of GANs, both discriminator and generator consisted of fully connected layers, however CNNs are much more suited to handle image data (Creswell et al., 2017). The research is done in Radford et al. (2015) proposed a new type of architecture called Deep Convolutional Generative Adversarial Network (DCGAN), in which both, the discriminator and the generator are represented by CNNs.

The development of Convolutional Neural Networks (CNNs) is related to machine learning techniques which are mainly designed to work with pattern recognition systems and when there is correlated structure in the data (LeCun et al.1998). Although multilayer feedforward networks can be used to perform such tasks, there are problems related to the structure using fully connected layers.

As matter of expression, the spatial distributed data will be treated as an image data, with width, W, corresponding to the x axis; height, H, corresponding to the y axis and depth,D, corresponding to the z axis, is composed of $H * W * D$ pixels. If the image data is used as an input to a fully connected layer containing N neurons, each neuron of the layer is connected to the input data, meaning that the total number of connections is equal to $H * W * D * N$. This means that, to properly train the network, it is

required a large dataset due to the number of parameters. Also, this structure is influenced by the position of the features in the image, meaning that to recognize the same feature in different positions it is necessary to insert these space variations in the training data. In convolutional networks, this problem is overcomed by replication of weights across space (LeCun et al., 1998).

Another limitation of fully connected networks is that they do not consider the topology of the input data. However, when handling with image data, the relative position between two or more features is an important factor for the pattern recognition task. The convolutional networks are able to extract and combine local features to identify more complex features, while fully connected networks cannot (LeCun et al., 1998).

The convolutional networks mechanism for pattern recognition is based on local receptive fields and shared weights. Each unit of a convolutional layer receives the input of a given window of the previous layer. The use of a local receptive field enables the network to work with feature extraction (edges, corners) within these windows, rather than working with each pixel value. For the next layers, the inputs will be combinations of the previous layers, which results that the deeper layers are able to recognize more complex image structures by combining simpler features of the previous layer (LeCun et al., 1998).

The feature that is identified by a unit can be present in different spatial positions of the image. Therefore, the convolutional layers use the concept of shared weights, in which a plane of the convolutional layer consists on units that share the same set of weights, responsible for the specific feature identification, distributed along the plane and perform the same feature identification, but in different positions along the image. These planes are called feature maps and each convolutional layer is made by a set of them (LeCun et al., 1998). The feature map scans the input data within each unit, performs the calculation and stores the output. This scanning process is similar to performing a convolution on the image with a given kernel which the values are given by the set of weights.

In the original formulation of CNNs (LeCun et al., 1998), it was used sub-sampling layers after each convolutional layer. These sub-sampling layers perform an average calculation in a given window and reduces the dimension of the input. The convolutional network is composed of a successive application of convolutional and sub-sampling layer. It is important to mention that each application of convolutional layer increases the number of feature maps and each application of sub-sampling layer reduces the dimension of the image.

In the most general form, a neural network is constructed considering L layers. The action of the i-th layer of this model can be denoted by the map $f_i : E_i \times H_i \to E_{i+1}$, being E_i, E_{i+1} and H_i the inner product spaces of the input output and connection weights and biases respectively, being $x_i \in E_i$ the input data and $\theta_i \in H_i$ the correspondent weights and biases. The network prediction can be simply represented by (Caterini & Chang, 2016b):

$$F(x;\theta) = (f_L \circ \cdots \circ f_1)(x), \tag{32}$$

where $\circ$ represents the composition operator and F the output of the multi layered network.

CNNs are a particular architecture that considers the cropping, embedding and mixing operators that defines convolution, as well as pooling (Caterini & Chang, 2016a). The training of the network is then characterized by the optimization of the loss function J, with respect to the weights θ that are updated for each iteration (backpropagation). Considering the gradient descent optimization algorithm, the loss function can be represented by:

$$J(x;\theta) = \frac{1}{2}\|y - F(x;\theta)\|^2, \tag{33}$$

and, for cross-entropy loss function:

$$J(x;\theta) = -\langle y, L\big(F(x;\theta)\big)\rangle - \langle \mathbf{1} - y, L\big(\mathbf{1} - F(x,\theta)\big)\rangle, \tag{34}$$

where $\mathbf{1}$ represents a vector of ones of appropriate length and L is an elementwise function with elementwise operation log.

In terms of handling spatially correlated data, CNNs have been widely used for supervised tasks, but not in unsupervised tasks. The development of DCGAN (Radford et al., 2015) attempts to overcome this barrier and demonstrates that the architecture constraints enable the model to learn the data distribution. The core of the DCGAN model relies mainly on four characteristics.

The first characteristic is the use of a set of convolutional layers in all network, removing pooling layers as proposed in Springenberg et al. (2015). This approach enables to reach state of the art performance with a simpler model, consisting only of convolutional layers. For the discriminator, it is used a set of strided convolutional layers to provide the downscaling of the features, for the generator is used a fractional strided convolutional network to enable features upscaling (Radford et al., 2015). The second characteristic is the elimination of fully connected layers linked to convolutional layers, the use of fully connected layers may increase training stability, but decrease the convergence speed of the optimization process (Radford et al., 2015), therefore increasing the time necessary to train the network. Figure 10 presents a scheme of the DCGAN generator network. It is possible to see that the network is composed only by convolutional layers, except at the input, which is performed a matrix operation to convert the input vector $\boldsymbol{z}$ to a convolutional representation.

In order to define the dimensions of the network layers, the value of the dimension of the output data is taken as reference and the dimension of the previous layers is calculated through a backward process. As a characteristic of this operation the quantity of features maps, also called depth, of the layer is increased by a factor of 2 and the height and width are decreased by a factor of 2, until the height and width of the first layer is equal to size of the kernel, in this case, $4x4$. Figure 10 provides an example of this

architecture. Also, it is necessary a reshape operation for the **z** vector, since the data needs to fit the dimension of the first layer.

The third is the use of batch normalization, this procedure stabilizes the network training by normalizing the output of the activation function to have zero mean and unit variance in each layer that it is applied per mini-batch. This method accelerates the training of the networks by allowing higher learning rates. The only layers that batch normalization is not applied are: at the output of the generator network and the output of the discriminator network (Radford et al., 2015), this result in training instability.

The fourth is the change in the activation function of each network. For the generator, except in the output layer, it was used a Rectified Linear Unit (ReLU) activation function (Section 2.2.2). For the output layer it was used a tanh activation function (Section 2.2.2). For the discriminator, except in the output layer, it is used a Leaky ReLU function (Section 2.2.2).

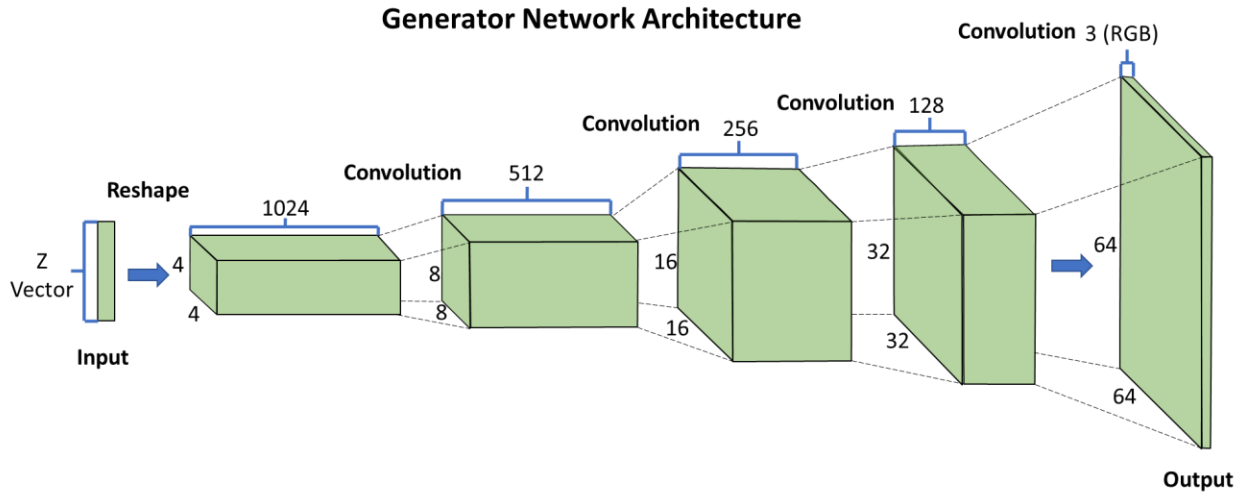

Figure 10. DCGAN Generator architecture. The numbers indicate the dimensions of each layer. For example, the first layers presents dimension of 4x4 for the height and width and a depth of 1024. the last layer is block of 64x64 (HXW) in RGB (3 channels), representing a colored image of 64x64 pixels (adapted from Radford et al., 2015).

2.3.2 Wasserstein Generative Adversarial Network

The optimization of the minimax game (Equation (24)) that results in $p_g = p_{data}$, although it can be reached in theory, it is hard to be achieved in practice (Chan & Elsheikh, 2017). At first, it was seen that training the discriminator until optimal and then updating the generator parameters is not a valid approach because this process results in vanishing gradients (Equation (26)). As highlighted in Arjovsky & Bottou (2017), the GAN, in its first formulation, computes and tries to minimize the Jensen-Shannon divergence between training data distribution and the generated distribution. If the distributions are far away, the resulting gradient of these divergences is almost zero, which means that the generator is not able to learn from the gradient that is backpropagated. On the other hand, if the D is not trained until optimality, the

gradient update of the generator may be inaccurate because the discriminator´s feedback is not appropriate since it was not fully trained to distinguish between original and synthetic data.

To avoid the gradient vanishing problem, the cost function for the generator was substituted by a similar one, given by Equation (27). Although this new formulation solved the problem of the vanishing gradients, network training became very unstable. As shown in Arjovsky & Bottou (2017), the new formulation results that the gradients follow a Cauchy distribution with zero-mean and infinite variance. The infinite variance results in training instability, and the zero-mean distribution ensues that, at average, the feedback is 0.

To improve GAN training and avoid the above problems, Arjovsky et al. (2017) compared popular probability distances and divergence (e.g. total Variation distance (Müller, 1997), Kullback – Leibler divergence (Kullback & Leibler, 1951), Jensen-Shannon divergence (Lin, 1991) and Wasserstein distance (Villani, 2009)) that are used to compare distributions, showing that using Wasserstein distance provides better results for optimization process since it is continuous and provides a usable gradient everywhere.

The Wasserstein distance is described in Equation (35):

$$W(p_{data}, p_G) = \inf_{\gamma \in \prod(p_{data}, p_G)} \mathbb{E}_{(y_1, y_2)\sim\gamma} \|y_1 - y_2\|, \tag{35}$$

where $\prod(p_{data}, p_G)$ denotes the set of all joint distributions $\gamma(y_1, y_2)$ with marginal distributions p_{data} and p_G, correspondingly (Chan & Elsheikh, 2017).

However, minimizing the Equation (35), in practice, is highly intractable. Due to the Kantorovich-Rubinstein duality it is used Equation (36) instead:

$$W(p_{data}, p_G) = \sup_{\|f\|_L \leq K} \{\mathbb{E}_{y\sim p_{data}} f(y) - \mathbb{E}_{\boldsymbol{z}\sim p_z} f\big(G(\boldsymbol{z})\big)\}, \tag{36}$$

where $\|f\|L \leq K$ is a real- valued K-Lipschits functions. If f is a parameterized family $\{f_\omega\}_{\omega\in W}$, then it is possible to rewrite the problem as Equation (37):

$$\max_{\omega\in W} \{\, \mathbb{E}_{y\sim p_{data}} f(y) - \mathbb{E}_{\boldsymbol{z}\sim p_z} f(G(\boldsymbol{z}))\}. \tag{37}$$

In Wasserstein GAN (WGAN), the objective is to minimize the Wasserstein distance. With this new objective function the discriminator, which outputs a probability, is now replaced by a function f, that is called critic, that outputs a real value that correlates with the sample quality (Arjovsky et al., 2017).

The main improvement of WGAN is that, since Wasserstein distance is differentiable and continuous, it is possible to train the critic until optimality before updating the gradients of the generator. This should be done to provide a more reliable optimization of the generator. As a result, WGAN improves the stability of the training process. A comparison between the training of a GAN discriminator and a WGAN critic is illustrated by Figure 11. When the original data and synthetic data are different, the gradient of the GAN

discriminator vanishes. However, the gradient curve provided by WGAN critic is smoother and linear, which ensures the generator to learn by backpropagation of the gradient.

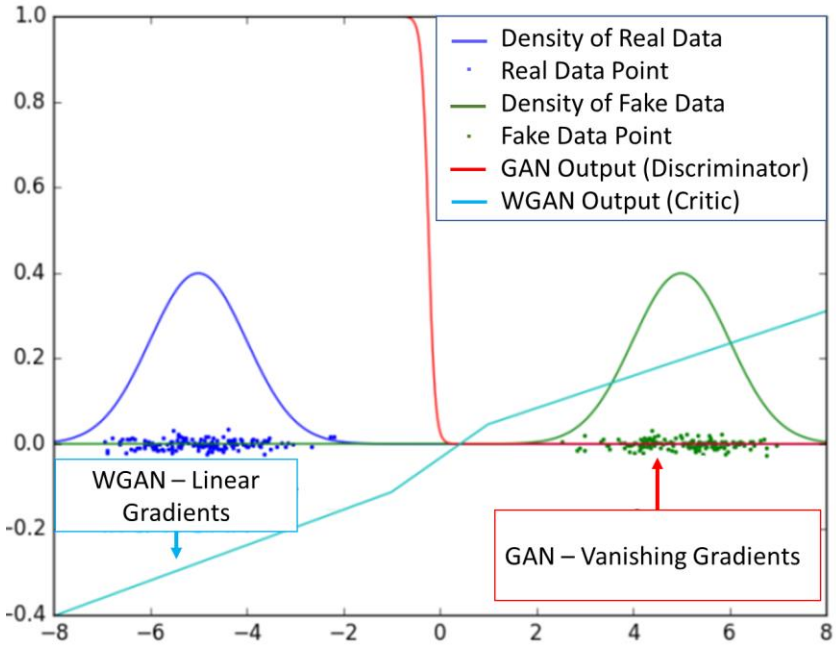

Figure 11. GAN and WGAN training process. The blue line represents the data distribution and the green data represents the synthetic data distribution. The light blue line represents the gradients of the WGAN critic, it is possible to see that it provides usable gradients. However, represented by the red line, it is possible to visualize the vanishing gradient if the GAN discriminator is trained till optimality (adapted from Arjovsky et al., 2017).

The training process of the WGAN is very similar to the GAN, with some particularities due to the replacement of the D by the critic. Due to the fact that f is a K-Lipschitz function parameterized by $\{f_\omega\}_{\omega \in W}$, to approximate this in the neural network, the weights ω are restricted to a compact space W, as proposed in (Arjovsky et al., 2017), $W = [-0.01, 0.01]$, after each iteration of the critic. The critic and G are updated in alternating steps. First, the critic is trained until optimality, or for n steps, following Equation (37). Then, G is updated using the gradient presented in Equation (38)

$$\nabla W(p_{data}, p_G) = -\mathbb{E}_{\boldsymbol{z} \sim p_z} \nabla f\big(G(\boldsymbol{z})\big). \tag{38}$$

The main improvement of WGAN is that it allows the critic to be trained untill optimality. But, as shown in (Arjovsky et al., 2017), the estimation of the Wasserstein distance correlates well with the visual quality of the synthetic data, which may provide a meaningful loss metric during the training.

2.3.3 Generating Unconditional Subsurface Geological Models with GANs

Geological models can be described as a representation of a property spatially distributed in a grid. The values of the property can be assigned to two main different domains: a discrete domain i.e. lithological facies or a continuous domain i.e. petro-elastic properties, It is common that the subsurface information is sparse, which means that, to create a geological model that represents the subsurface, it is necessary a

method to fill grid points with data values from the hard data set distribution i. e. measured points. Usually, for the creation of geological models with discrete data values, a Multi Point Statistics (MPS) algorithm is used and, for continuous data values, a common method is the use of stochastic simulations.

To create models with MPS algorithms it is necessary to create a geological model presenting repetitions of the patterns that are of interest to be reproduced on the geological model (Tahmasebi, 2018). In the other hand, to create geological model through stochastic simulations it is necessary to define a variogram model derived based on the experimental variogram model. This implies that the geological models generated through these methods are extremely dependent of the defined parameterization.

A recent work done by Chan & Elsheikh (2017) has shown that GANs are a powerful mechanism to generate parameterized geological models. In this work, GAN networks were trained in two different permeability datasets: i) data of semi-straight patterns channelized structures and ii) meandering channelled structures. Both datasets are represented by a binary variable in which 1 represents high permeability channels and 0, low permeability channels. Each training dataset was created by cropping a $250\ x\ 250$ pixels image of the patterns in smaller windows of $50\ x\ 50$ pixels (Figure 12), providing more than $40\,000$ training images for each application example. The network architecture for this work was based on DCGAN (Radford et al., 2015).

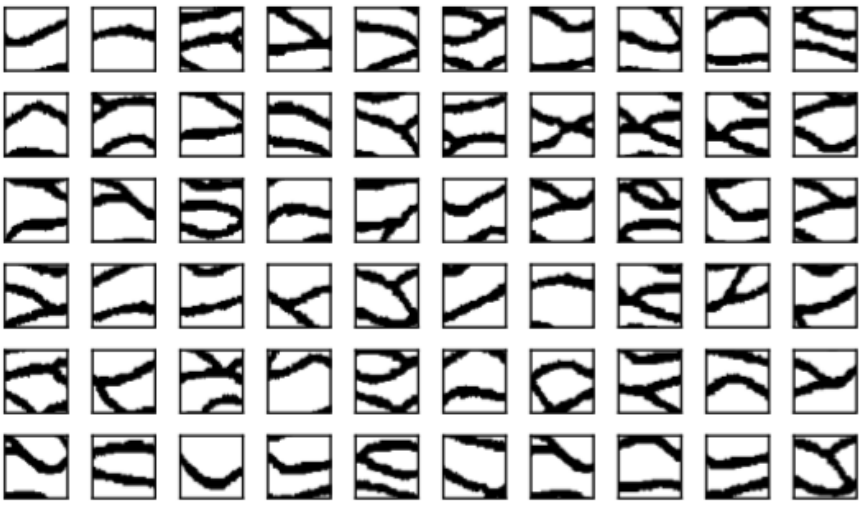

Figure 12. Original realizations of semi-straight channel (Chan & Elsheikh, 2017).

At the first stage, a visual comparison between the training dataset and generated samples (Figure 13) was done, and the generated samples presented highly plausible patterns, based on each training data. However, the presence of some artifacts was observed, like pixel noise and the channels that presents endpoints, ignoring one of the main characteristic of the dataset that is the continuity of channels. Following the visual comparison, the original and generated data were statistically compared and it was verified that the statistical data were also similar. The interesting point is that the generator network became able to reproduce parameterized models, which means that the network itself learned the

parameterization and it is not wrong to think that the parameterization of the network is as reliable as those from a variogram (case of stochastic simulation) or a single training image (case of MPS) due to the total number of parameters that the network possess, i. e. weighted connections and activation functions.

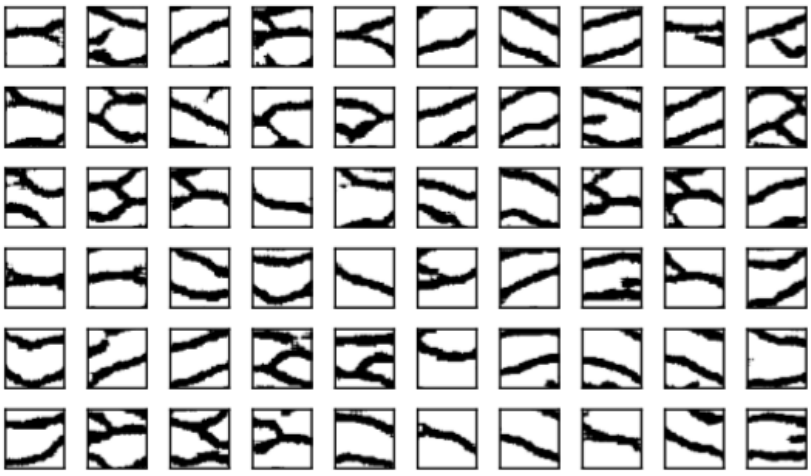

Figure 13. Generated realizations of semi-straight channels. It is possible to some artifacts like pixel noise and channels´ endpoints (Chan & Elsheikh, 2017).

Then, a deeper study was done in terms of comparison. Uncertainty propagation study was performed by analyzing the response of a simplified subsurface flow problem. This test was applied in four different datasets, each containing 5000 samples: i) original data of the semi-straight structure, ii) synthetic data of the semi-straight structure, iii) original data of the meandering structure, iv) synthetic data of the meandering structure. Then, the statistics of the following variables of the flow problems were compared: water breakthrough times, water-cut curves and saturation in different time steps. As a result of the comparison, it was verified that the GANs can be used not only to parameterize complex geological patterns and structures, but that the response to the flow problem was close to the response of the original data, meaning that it can reproduce characteristics that are not directly present in the data. An example can be seen in Figure 14, where the histograms of the saturation, calculated during the flow simulation, for the original data and the realizations from GAN are compared.

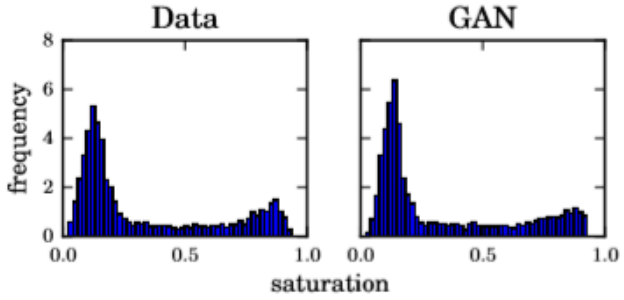

Figure 14. Saturation histograms for the flow simulation. The picture depicts the time t=0.25 of pore volume injected (Chan & Elsheikh, 2017).

Therefore, the GAN has been shown to be a powerful tool in as an alternative to generate unconditioned subsurface models that resembles the patterns and data distribution of the training dataset, which in this case was a facies dataset, but that in theory can be more diverse, for example a continuous dataset.

2.3.4 Generating Conditional Subsurface Geological Models with GANs

A recent application of GANs is the generation of synthetic samples conditioned to existing data. In Yeh et al. (2017), GANs are used to perform a semantic image inpainting task, i.e. to fill-in a portion of a corrupted image based on the available data of the image. Yeh et al. (2017) showed that this task can be performed by a trained GAN conditioned by the available data.

The generator of the adversarial network can be seen as a mapping function $G(\boldsymbol{z}; \theta_G)$ that maps the input $\boldsymbol{z}$ to an output, in this case, the image (Goodfellow et al., 2014), the $\boldsymbol{z}$ is an encoding of the output in the latent space of the generator (Yeh et al., 2017). In Yeh et al. (2017) it is proposed that the problem of semantic image inpainting can take advantage of this encoding mechanism by looking for the closest encoding $\hat{\mathrm{z}}$ of the corrupted image y at the latent space and use $\hat{\mathrm{z}}$ as input of the generator network to generate an image $\hat{\mathrm{y}}$, which corresponds to the output of the closest encoding in the latent space and, in theory, generating the image $\hat{\mathrm{y}}$ that is the most similar to image y. The closest encoding $\hat{\mathrm{z}}$ is defined by weighted context loss, defined by a mismatch between the actual output and conditioning data, and a perceptual loss, which is related to penalization of unrealistic images.

The method proposed by Yeh et al. (2017) resembles an "inversion" of the generator network, based on the backpropagation to the input data, solving an optimization problem. This approach is defined by the optimization of a specific loss function, presented in Equation (39) as follows:

$$\hat{\mathrm{z}} = \arg\min_{z}\{\mathcal{L}_c(\boldsymbol{z}|y, M) + \mathcal{L}_p(\boldsymbol{z})\}, \tag{39}$$

where $\mathcal{L}_c$ denotes the context loss, $\mathcal{L}_p$ denotes the perceptual loss and M denotes the binary mask of the corrupted image and indicates where the image has data or not.

The role on the context loss component, $\mathcal{L}_c$, is to acquire information of the available data to condition the generated sample. This loss is calculated by a weighted l_1 norm between the corrupted image and the generated image. The importance of the weighting term W is that pixels near corrupted ones provide more useful information for the semantic inpainting points than pixel locations that are far from the corrupted region. Then, the context loss is defined by Equation (40) as follows (Yeh et al., 2017):

$$\mathcal{L}_c(\boldsymbol{z}|y, M) = \|W \odot (G(\boldsymbol{z}) - y)\|_1, \tag{40}$$

where $\odot$ denotes an element wise multiplication.

The weighting term, W, is defined by Equation (41) (Yeh et al., 2017):

$$W_i = \begin{cases} \sum_{j \in N(i)} \frac{(1 - M_j)}{|N(i)|} & if\ M_i \neq 0 \\ 0 & if\ M_i = 0 \end{cases}, \tag{41}$$

where i denotes each pixel location, W_i denotes the weight of each pixel location and $|N(i)|$ represents the number of pixels in a given window centered in i (Yeh et al., 2017).

The other component of the loss function, the prior loss, corresponds to a penalization to images that are unrealistic, meaning that they are not similar to the ones presented at the training data (Yeh et al., 2017). This approach resembles the discriminator trying to identify fake samples, therefore the prior loss is similar to the GAN discriminator, as seen in Equation (42):

$$\mathcal{L}_p(\boldsymbol{z}) = \lambda \log\Big(1 - D\big(G(\boldsymbol{z})\big)\Big), \tag{42}$$

where λ is defined as a parameter to balance between the two losses. The objective is to generate realistic images by trying to fool the discriminator with the generated samples.

Once the loss function is defined, $\hat{\mathbf{z}}$ can be found by randomly starting the process with a given $\boldsymbol{z}$ and optimizing Equation (39).

Mosser et al. (2018a) introduced a GAN application to generate samples conditioning the output of a trained GAN to a lower dimensional data. Mosser et al. (2018a) presented two different cases for GAN conditioning: i) conditioning a 3D output of the Maules Creek alluvial aquifer to a 1D section corresponded to a well segment (Indicator variable). ii) Conditioning of a 3D pore scale output of Ketton limestone to 2D centered orthogonal sections of data (Continuous variable).

In both cases the network structure was based on DCGAN (Radford et al., 2015) and to perform sample conditioning it was applied a similar approach to semantic image inpainting based on Yeh et al. (2017), i.e. the same concept of minimization of the context and prior losses. The stopping criteria for optimization were: i) in the case of a binary variable, the matching of all data within the conditioning region. ii) In the case of a continuous variable, it was defined that the content loss should be lower than 0.001.

3 Methodology

This section is divided in two parts: the first part introduces the GAN as a method to create unconditioned and conditioned subsurface models; the second part describes a seismic inversion method that aims to invert seismic data simultaneously for acoustic impedance and facies models, in which the facies models are created by a trained GAN.

3.1 Generation of Subsurface Models

Here, the procedure used for the generation of subsurface models using a GAN is described. The process starts with the selection of the dataset to be used as training data and required pre-processing tasks. Then, it is presented how network architecture and parameters were set. Following, the training procedure is explained. Once the network is trained, it is ready to generate conditioned and unconditioned subsurface models.

3.1.1 Dataset Configuration

The first step is to create a database of subsurface models (discrete or continuous) to be used as the GAN training dataset.

The second step determines the size of the training dataset and generated models of the network. As a starting point, the dimensions of training and generated models are the same. This is an important step because the network architecture is defined in a way that it works with models of same dimension (height and width) and, once the network is designed, the training models must have the same size as configured for a proper network training. Related to the dataset, the training data size must preserve the features of the dataset for the network to be able to reproduce the data distribution. In this study, the size of the models was defined as $64\ x\ 64$ or $128\ x\ 128$ pixels. In cases where the training dataset was composed by models with dimensions greater than the ones defined for data input, a sliding window technique in the training dataset was used in order to create cropped models that fits the defined dimension for the input data. Using a sliding window also increases the number of training samples due to the image cropping in images with smaller dimension, this is an important fact since the size of the training dataset is increased.

The third step is to normalize the training dataset in the tanh domain ([-1,1]), since the output function of the generator is a tanh function. This ensures that the input data domain (for the discriminator) is the same as the output data domain (of the generator).

Finally, the fourth step is to separate the normalized training models in batches for the network training procedure. Also, to ensure a better training, batches are shuffled after each iteration before being inputted to the discriminator.

3.1.2 Generative Adversarial Network Architecture

The applied generator network architecture is based on the DCGAN (Radford et al., 2015). The number of layers was calculated depending on the size of the output from the generator, which in this work, present the same dimension of the data from the training dataset. Also, the training dataset was composed of images with one channel, therefore are represented as images in grayscale for continuous variables and black and white for discrete ones. The process of layers dimensioning was explained in Section 2.3.1. For the case models generation with dimension $64\ x\ 64$, the initial depth is 1024, containing 5 convolutional layers, including the one necessary for the input vector reshape. For the case of models generation with dimension $128\ x\ 128$, the initial depth is 2048, containing 6 convolutional layer.

It is important to notice that it is possible to add extra layers in both networks, the generator and discriminator. However, this has not been tested under the scope of this thesis.

3.1.3 Hyperparameter Adjustments

For samples generation that resemble the training data distribution and its visual pattern it is necessary to train the GAN. At first, it is necessary to define the network parameters to be used, related to its architecture and data input and output. GAN's training parameters follow those described in Arjovsky et al. (2017). In the scope of the current work, several GAN parameters were considered. Nevertheless, for the application examples shown herein, GANs were set with the following parameters Table 1.

Table 1. Common parameters for all four networks.

Parameter name	Value
Input Image Size	64×64 or $128\ x128$
Output Image Size	64×64 or $128\ x\ 128$
Nº of Image Channels	1 (Grayscale)
Discriminator Activation Function	Leaky ReLU (slope 0.2)
Generator Output Transfer Function	$tanh$
Optimizer	Adam
Learning Rate	0.0002
Beta 1	0.5
Beta 2	0.999
Weight Clipping Upper	0.01
Weight Clipping Lower	-0.01
Critic Iterations	5
Size of Latent Vector	100
Generator Activation Function	ReLU

In the application examples shown below, the input and output model and the batch number are different depending on the example.

3.1.4 Generative Adversarial Network Training

The DCGAN training follows the same algorithm proposed by Goodfellow et al. (2014). After each training epoch, the performance of the generator was evaluated comparing its visual characteristics and the histogram with the ones of the training data set.

3.1.5 Generation of Unconditioned Models

Once the GAN is trained, the method to produce subsurface models (i.e. samples) is by inputting a random z-vector to the generator network. In this case, the network generated unconditional models.

3.1.6 Generation of Conditioned Models

In subsurface modelling and characterization, the major interest is in producing conditioned models that honor existing experimental data. To produce conditioned models, the same network that produces unconditioned models can be used since the optimizations occur in the values of the z-vector (actual location in the latent space), not in the network weights, adjusted during the training of the network. The first step is to define the conditioning data. The conditioning data are selected as a set of data values and location that are contained in a training image. For the generated conditioned models, it is expected that the conditioning data is reproduced within the defined locations. For the work, the conditioning data was chosen by selecting random models from the training dataset , and then, selecting regions within these models as the conditioning data location and values. Therefore, a mask matrix composed of the values 0 and 1 is created. In positions where it is desired to place a conditioning value the value of the mask is set to 1, otherwise, the value is kept as 0. Then, this matrix, with same dimension as the training dataset model, multiplies the data, leaving 0 values outside the conditioning data regions and the data values in location where the mask value was 1.

After the conditioning data is set, a random z-vector that produces an unconditioned model is sampled. Then using the algorithm described by Yeh et al. (2017), the z-vector is optimized in order to find the z-vector that produces a model that best matches the conditioning data.

An important observation is that after the training and generation of the models, for simplification purposes, the training and generated data are normalized, using a min-max normalization procedure (Patro & Sahu, 2015), to the range [0,1] for statistical analysis.

3.1.7 Subsurface Model Reconstruction

This part of the work was developed to assess GANs performance in producing geological models under several scenarios. The performance was checked in two different datasets: a discrete, which in geosciences most of the times represents geological facies; and a continuous dataset, which may represent any property of the subsurface.

In the first phase, unconditional models are produced for each dataset. These models are used to verify if the generative model is able to produce geologically reliable models by doing two comparisons: a visual

comparison between the generated models and the ones from the training dataset; and a geostatistical comparison, comparing the histograms and variograms between the generated and training models.

In the second phase, the ability of the GAN to produce conditioned models was tested in three different conditioning scenarios:

i) Model reconstruction using a small region within the simulation grid as conditioning samples ("square"). This scenario reproduces an extreme case where one have access to the information about the subsurface geology in a small region. This can be a place where seismic inversion converged locally (i.e. where the samples were kept) or corresponding to a seismic data volume much smaller than the area of interest. This might happen in early exploratory stages where the area of interest (e.g. an exploration block) is much larger than the available data;

ii) Model generation conditioned to well data ("wells"). This is a classical spatial inference problem where the subsurface properties are known at few and sparse locations often tackled using geostatistical modelling tools.

iii) Model generation conditioned to scattered points across the grid ("points"). This scenario may reproduce cases where sampled values present along the grid are known.

For the scenarios dealing with discrete variables and conditioned to data points it was used a combined loss function given by a BCE function for the content loss plus the discriminator loss, for the perceptual loss (Mosser et al., 2018b). The stopping criteria are defined as a percentage of the match between the values of the actual output and the conditioning data.

For the conditioning of continuous variables, the sum of the squared error between the actual output, the conditioning data and the perceptual error were used as content loss function. Since the network works in normalized values, the values of squared errors can be directly translated to percentage errors, for example: a mean squared error of 10^{-2} is an error of $\pm 10\%$. We tested the following MSE: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$ that corresponds to an error of approximately 32%, 10%, 3.2%, 1%, 0.32% and 0.001% of the data span respectively. The MSE of each generated samples the sum of the squared errors divided by the number of conditioning pixels.

3.2 GAN Geostatistical Seismic Inversion

As the first step, it is necessary to create the training dataset following the procedures described in Section 3.1.1. It is important that the facies dataset reflects the patterns of the features of the seismic environment, therefore a specific dataset for that seismic dataset is used. These datasets can be created through MPS algorithms or by conversion of attribute maps to facies. Once the dataset is ready, the network was designed as in Section 3.1.2 and the parameters were adjusted as in 3.1.3. The network training followed the procedure presented in Section 3.1.4.

Erro! Fonte de referência não encontrada. illustrates the workflow after training. First a set of u nconditioned facies model is generated. For the optimization of the facies model it was decided to use 3 types of losses. The first two, already presented, the content loss, given by the mismatch of facies values and the conditioning data, and perceptual losses, inputting the models into the trained discriminator, are directly calculated from the facies model as described previously. The third loss is related to the correlation between the synthetic seismic generated though the Adapted GSI Algorithm and the original seismic. The process to computed each loss are shown later. These three losses links the information that necessary to optimize the facies model. Once the three losses are computed, it is used a weighted sum to compute a total loss.

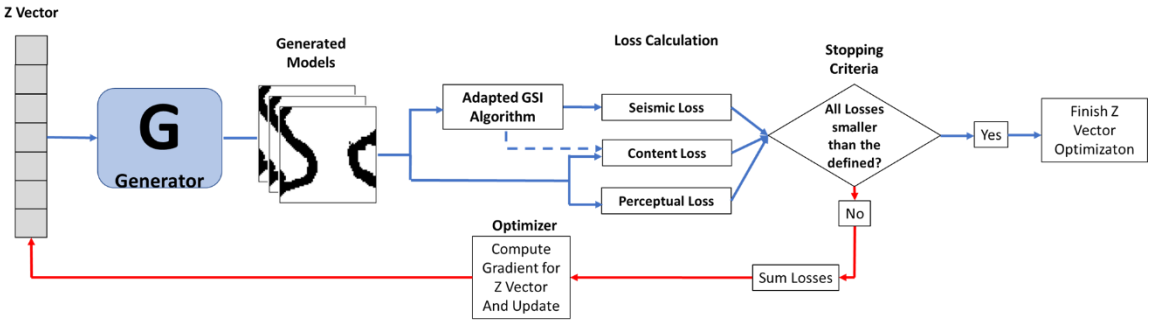

Figure 15. Workflow of the GAN Geostatistical Seismic Inversion after training.

The total loss is propagated backwards though the generator and gradient is computed for the z-vector in order to minimize the loss function. For the optimization of the loss function in terms of the z-vector, it was used the Adam optimizer with a learning rate of 1. Then, the z-vector is optimized and a new set of facies model is produced. The optimization process occurs a defined threshold value is achieved for each loss.

The proposed method of using GAN in a seismic inversion can be summarized in the following sequence of steps:

1) Creation of a database containing the expected spatial pattern of the facies. This database can be created through MPS or attribute maps of the seismic converted to binary maps with k classes, $k = 1, 2, \dots K$, where K is the total number of facies. This database will be the training database of the GAN.
2) Train the GAN with the training dataset built in 1).
3) Create *N* facies models with the GAN trained in 2). These models will comprise a certain number of seismic traces, W, and vertical samples, H.
4) Perform stochastic sequential simulation of the acoustic impedance conditioned to the facies model using DSS with multi-local distributions (Nunes et al., 2017) for the entire grid. The well data will be used as experimental data and variogram will describe the spatial continuity pattern.
5) For each acoustic impedance model computed the synthetic seismogram by calculating the reflection coefficients, which are convolved by a wavelet.

6) For each model, it will be computed a correlation between the synthetic seismic and the real seismic, at the same location. To each model it will be assigned a mismatch value.
7) For each model, it will be computed a perceptual loss by inputting the samples in the trained discriminator. The perceptual loss is the output of the discriminator.
8) For each iteration, based on the correlation coefficient, it is created a best facies model to be used as conditioning data.
9) Calculation of the misfit between the generated facies model and the conditioning data. Then, penalize the mismatch of the models.
10) Computation of a combined objective function relating the correlation in 6), the loss in 7) and the mismatch in 9) for each model.
11) Use the objective function to create new facies models with GAN. This can be done using the optimizer of GAN or using a simple conjugated gradient.
12) Iterate until a threshold defined mismatch value is achieved.

3.2.1 Adapted GSI Algorithm

The adapted GSI algorithm carries the main idea of the GSI algorithm (Section 2.1.2): the combination between stochastic simulations and genetic algorithm. But here, this algorithm is implemented with two distinct objectives:

i) Create a synthetic seismogram that correspond to each one of the generated facies model
ii) Create a best facies model using the best local correlation coefficients between all the models (similar to the creation of the best acoustic impedance models, but for facies values).

Once the facies models are generated by the GAN, they are inputted to this algorithm. The first part of the algorithm, related to the objective i), is shown in Figure 16. For each facies model, using information of the facies values at each location, the acoustic impedance realizations are simulated conditioned to the facies values, using DSS with multi-local distributions. Therefore, it is necessary to create variogram models for each class of facies in the model. As in the conventional GSI, it is produced a best acoustic impedance model. For the GAN Geostatistical Seismic Inversion, this best acoustic impedance model is used to create a best synthetic seismic model which is used to compute the seismic loss. This process is repeated for the N facies models created.

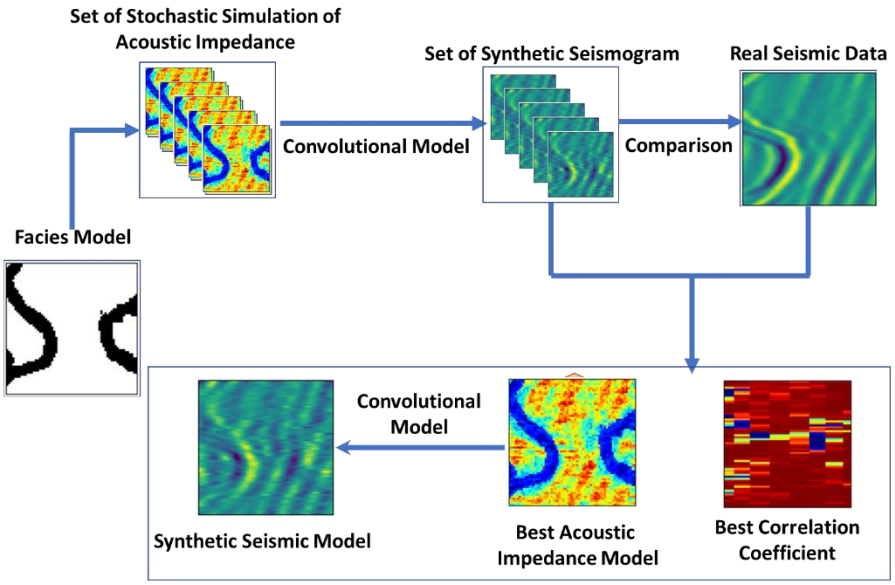

Figure 16. Workflow of the first part of the Adapted GSI Algorithm.

Observing Figure 16, it is possible to see that for each facies model inputted, the algorithm outputs: a best correlation coefficient, a best acoustic impedance and a synthetic seismic model. These models are assigned to that facies model.

The second part of the algorithm, related to the objective ii), is to create the best facies model taking in account all the pairs of models of best correlation coefficient and facies models. A best facies model is created by selecting, position by position, from all the models, the facies value assigned to the facies model that present the highest correlation coefficient. This process is shown in Figure 17. The idea of this process is the same used to create the best acoustic impedance cube in GSI algorithm.

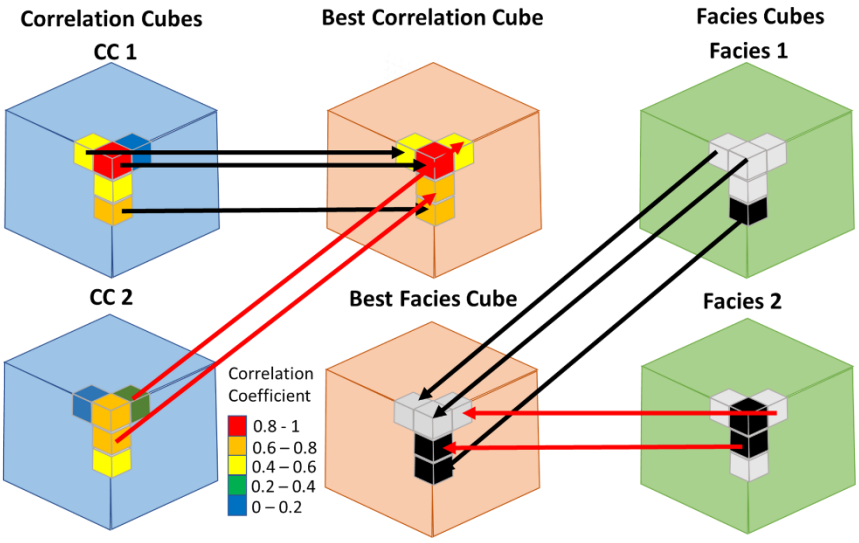

Figure 17. Creation of best correlation and best facies cubes from $N = 2$ models. For each position of the grid, it is chosen the segment of the correlation cubes that presents higher correlation values. In this case, the color of the

arrows, black for cube 1 and red for cube 2, indicates the origin of the segments. Then, the model of best facies values is built with the corresponding segments to the ones with higher correlations for each position.

The best facies model created will then be used as conditioning data for the optimization process. At the end of each iteration, the best facies model is stored and also the assigned correlation coefficient at each position. At the next iteration, this best facies model is loaded and it is only updated if the correlation coefficient value of a model in the actual iteration is higher than the stored values.

3.2.2 Content Loss Calculation

In the optimization context, the content loss provides a quantification of the mismatch in terms of facies values between the conditioning data and the actual generated facies model. For this method, it was used two different kinds of conditioning data: i) data of facies values from wells located and ii) the facies values of the best facies model obtained from the adapted GSI algorithm. Both conditioning data combined in just one conditioning model. Important to say that the conditioning data originated from well data overlaps that the best facies model in this region, since these values (from well measurements) present higher certainty values than the one originated from the best facies model. The two kinds of conditioning data can be seen in Figure 18.,

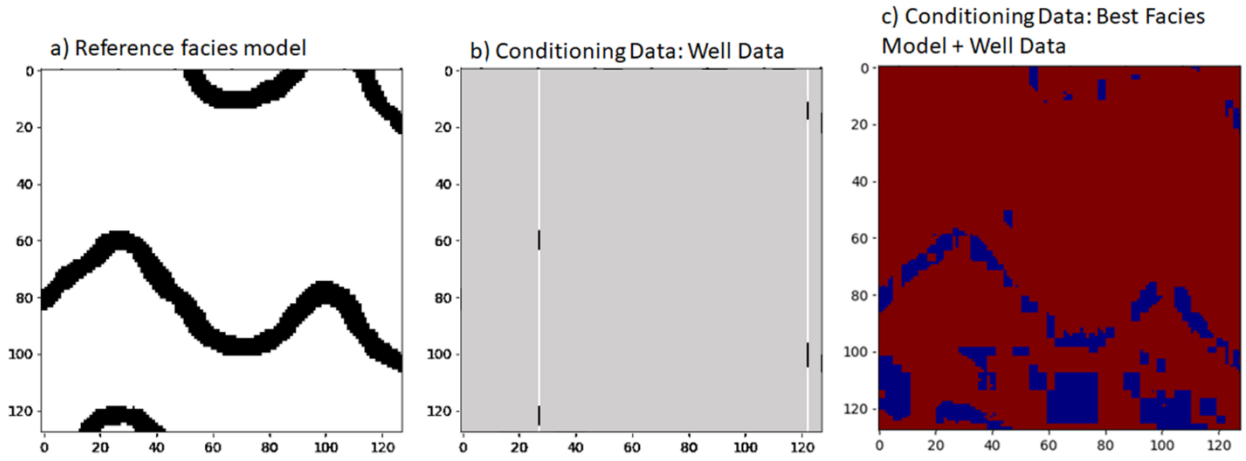

Figure 18. a) Reference facies model. The values of the facies at the well locations are based on these models. b) the facies values selected as conditioning data at well locations (x=28, x=123). c) example of best facies model obtained from the Adapted GSI algorithm, however this model already incorporates the well data. At the well locations, it is possible to recognize the blue traces representing the channels facies. The use of different colors just represent that the data have different origin.

After some tests, it was observed that the best way to compute the content loss was dividing the content loss in two components: i) the mismatch between where in the conditioning data the facies values are channels (1), but in the generated facies model are non-channel (0) ii) the opposite, the mismatch between where in the conditioning data the facies values are non-channels (0), but in the generated facies model are channels (1). To do this, it was necessary to create masks using the conditioning data as base. The process is shown in Figure 19. The idea of the mask is to multiply for the facies model and

conditioning data. In this way, for the regions containing zero values no mismatch will be calculated only for regions of the mask where the value is one.

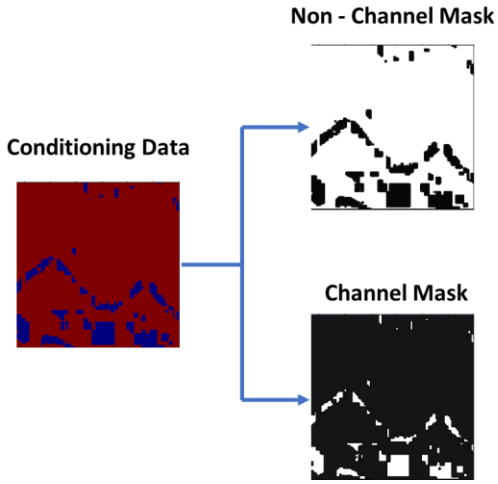

Figure 19. Computation on masks from the conditioning data. The white represents values of 1 and the black values of 0.

After the calculation of the masks the two content losses are computed. Figure 20 shows the computation of the content loss using the non-channel mask (Figure 19). This content loss quantifies the mismatch between regions where, comparing to the conditioning data, non-channel facies are channels in the generated model. For the computation of the loss, it used a BCE between the models, this computation provides a scalar value to the non-channel content loss.

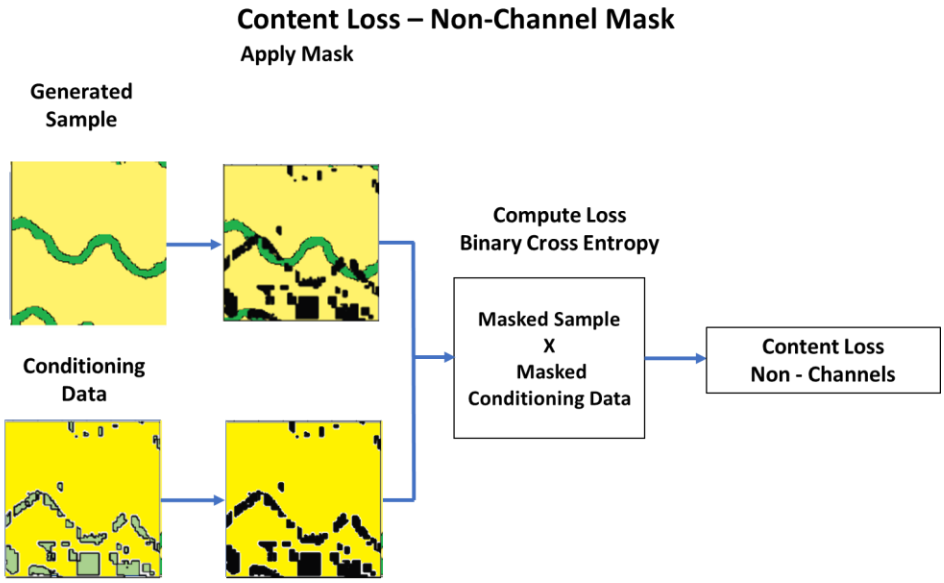

Figure 20. Computation of the non-channel content loss. The non-channel mask is applied to the generated model and to the conditioning data. Then, a BCE is computed between the models.

Figure 21 shows the computation of the content loss using the channel mask (Figure 19). This content loss quantifies the mismatch between regions where, comparing to the conditioning data, channel facies are non-channels in the generated model. Here a BCE between the models is also used.

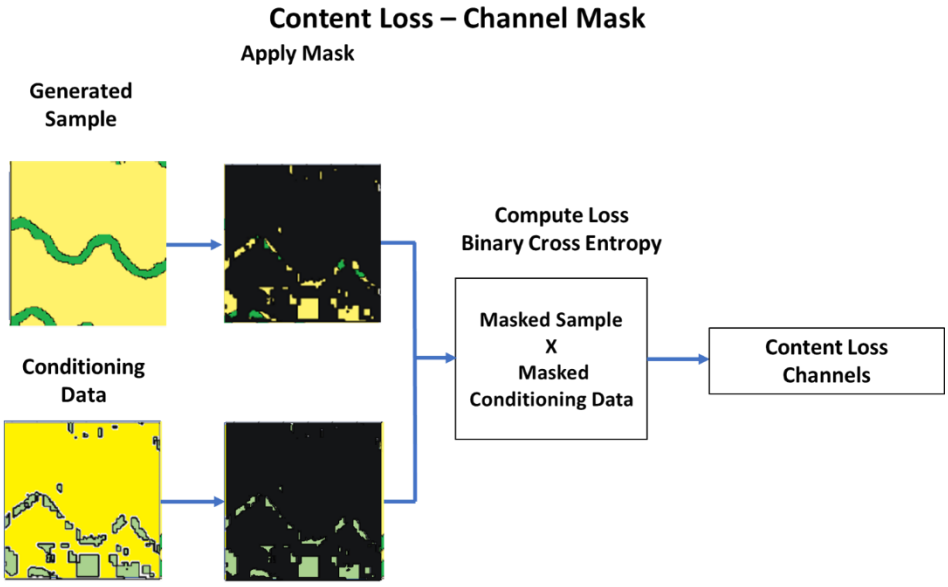

Figure 21. Computation of the channel content loss. The channel mask is applied to the generated model and to the conditioning data. Then, a BCE is computed between the models.

The final content loss used is the weighted sum of both losses shown previously. The weights are used due to the fact that, in the beginning, there are not many conditioning data related to facies values of channels, the predominant facies is non-channel. Therefore, the channel content loss is multiplied by a factor of 10 in order to penalize more the realizations where there are not present channels where it is expected to be, since the objective is to create a facies model where the channels structure are reproduced. And the non-channel content loss works in a way that channels realizations that present channels outside the channels mask, are penalized. Also, according to the tests, it has been seen that dividing the content loss by the number of pixels provided better results. The final content loss is given by:

$$Content\ Loss = \frac{10 * BCE_{Channel} + BCE_{Non-Channel}}{H * W}, \tag{43}$$

where H and W are the height and width of the model, respectively.

3.2.3 Perceptual Loss Calculation

The perceptual loss (Figure 22) is used to compute how similar the data distribution of the generated model is compared to the one of the training dataset. To perform this calculation, it is used the output of the discriminator network used during the training process. The discriminator has the role to distinguish between training data and generated data, therefore it can be used to evaluate if the generated model is similar to the training data or not.

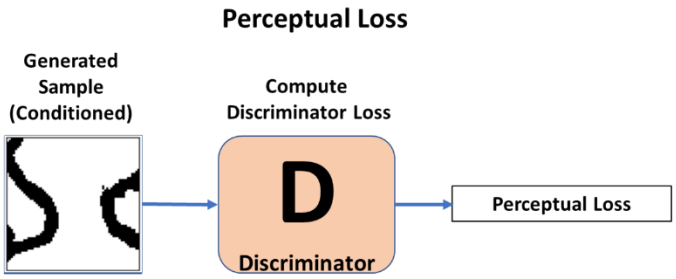

Figure 22. Calculation of the Perceptual Loss of the model.

3.2.4 Seismic Loss Calculation

The seismic loss computes the correlation between the computed synthetic seismic and the real seismic values. The process to calculate this loss is presented in Figure 23. Through the Adapted GSI Algorithm, it is calculated a synthetic seismic that represents the given facies model. Then it is computed a correlation coefficient between the values of the synthetic seismic and original seismic. The value of the correlation coefficient is given by the following expression (Azevedo & Soares, 2017):

$$Rho = \frac{2 * \sum_{i=0}^{n} xy}{\sum_{i=0}^{n} x^2 + \sum_{i=0}^{n} y^2,} \tag{44}$$

where x represents the flattened array of the original seismic, y represents the flattened array of the synthetic seismic and n represents the total amount of pixels in the picture ($H * W$).

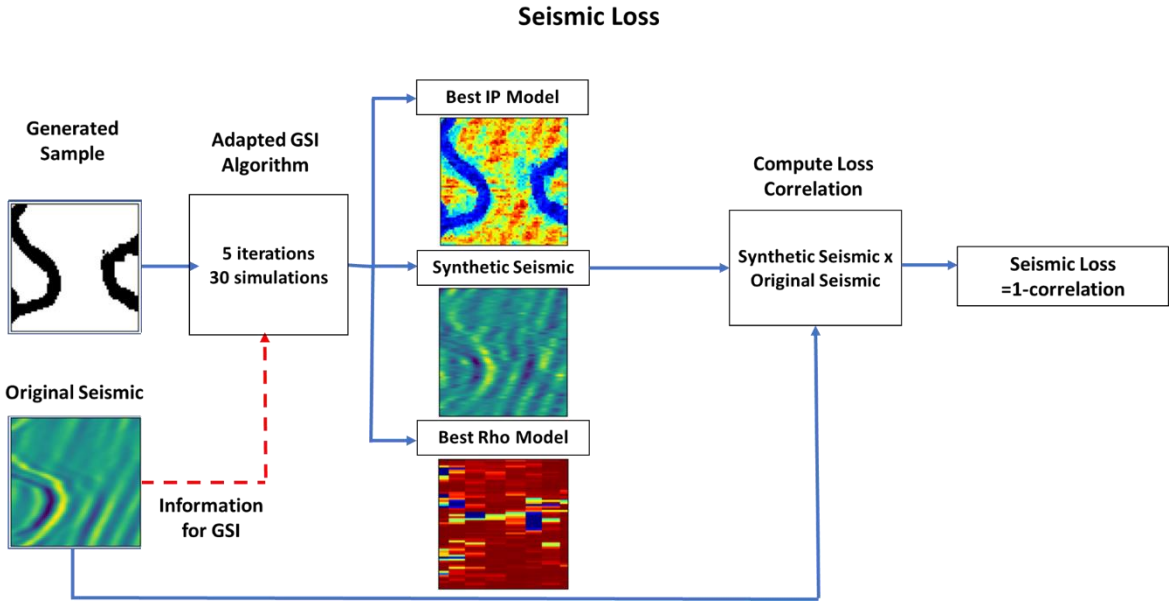

Figure 23. Calculation of the Seismic Loss of the model.

In order to input the correlation in the objective function, it was used the value $1 - Rho$. This is because the optimization process is structured in a way to minimize the values of the loss. According to this

modification, As the value of Rho increases, which presents the maximum of 1, the value of the loss decreases and vice-versa.

4 Results and Discussion

4.1 Exploratory Tests

The GAN is a new technique and the first relevant works that present an application in the geosciences field dates from 2017. Therefore, it was necessary to develop exploratory tests around this technique. These tests are described in the subsequent sub-sections. These tests were performed training the GAN considering continuous and discontinuous variables: i) discrete dataset containing a facies model presenting a channelized structure, ii) continuous dataset presenting realizations of p-wave velocity.

4.1.1 Dataset Description

The facies training dataset is composed by cropping a well-known training image with semi-straight channels (Strebelle, 2002) with a sliding window with $64\ x\ 64$ pixels. A total of 34,969 models were used to train the network (Figure 24). In this example, the black facies (0) correspond to shales, representing low permeability, while the white (1) to sand channels, presenting high permeability. The use of this dataset enables to test the capability of the GAN in producing the channels configuration and its connectivity.

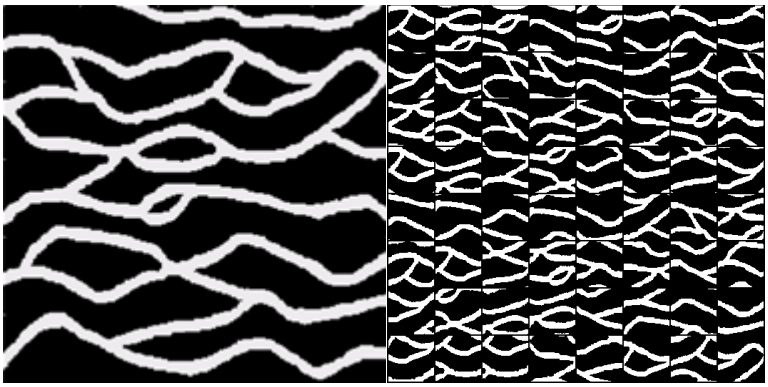

Figure 24. At the left side, the original training image of semi-straight channels. At the right side, the training images obtained after applying the sliding window.

The continuous dataset is composed by 1000 realizations of P-wave propagation velocity created through Direct Sequential Simulation (Soares 2001) using a spherical variogram model with range of 5 pixels for the horizontal direction and range of 10 pixels for the vertical direction. The P-wave velocity values range between 3730 m/s and 5526 m/s (Soares, 2001). All models within this dataset share the same histogram and spatial continuity pattern as shown in Figure 25, this is an intrinsic property of models produced with stochastic sequential simulation. For this case, the data was normalized in the range [0,1] and a statistical summary of the dataset is presented in Table 2.

Table 2. Statistical summary of the continuous dataset used in the original and normalized domains.

	Continuous data	
	Original	Normalized
Mean	4915.83	0.660
Std. Dev.	444.87	0.248
Min	3730.98	0.000
25%	4751.95	0.569
50%	5039.11	0.729
75%	5216.48	0.828
Max	5526.12	1.000

Examples of a single realization can be seen in Figure 25, the histogram and variograms are displayed for further comparison with the statistical data of the generated models.

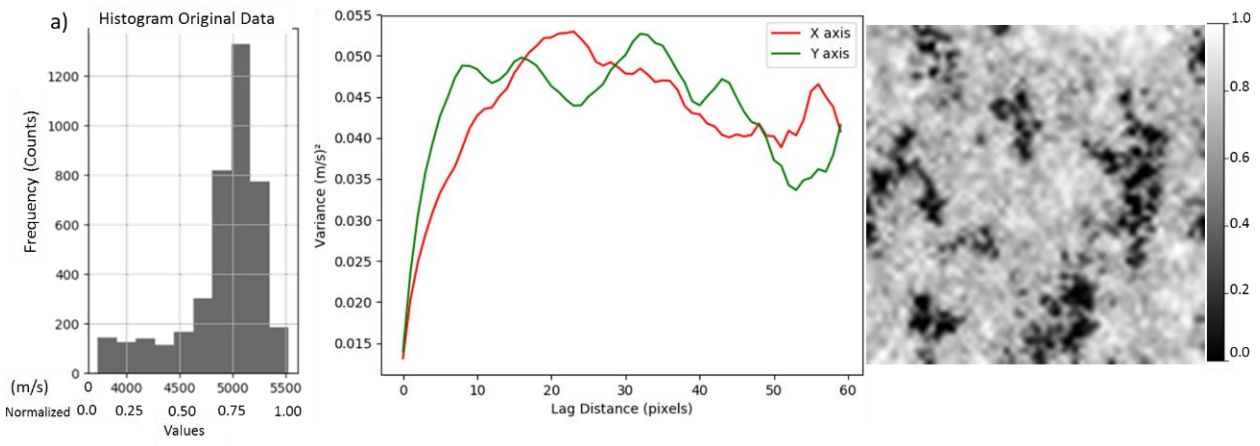

Figure 25. a) Histogram of P-wave velocity in normalized and original domains and b) the correspondent variograms for two perpendicular directions; c) a single realization of P-wave velocity.

4.1.2 Generation of models

The generation of unconditioned models had two main objectives: a) verify the quality of the generated models and b) verify if the statistical data of the generated models are similar to the training dataset. These two procedures were also used to determine the GAN that provided the best samples.

Here the results of the unconditioned facies models are presented. In Figure 26 shows a batch of training models (left) and a batch of generated models (right). Comparing both batches of models, it is possible to say that the GAN is able to generate models with similar spatial patterns as those inferred from the

training dataset, including the continuity, which is an essential feature to be reproduced in terms of channels modelling. Figure 27 compares the histograms between trained and simulated models, which reinforces the ability of the GAN to reproduce not only the visual patterns, but also the intrinsic geostatistical parameters of the training dataset. Important to point to the fact that both histograms were computed using the entire training dataset composed by 34969 realizations, not only the samples presented in this batch.

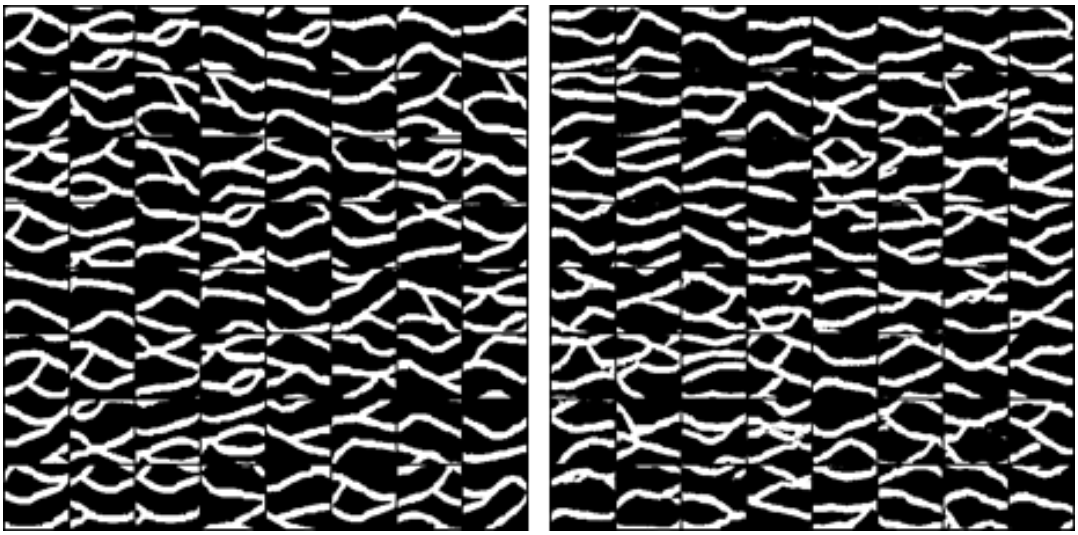

Figure 26. At the left side, sliding windows obtained from the original dataset, a training batch. At the right side, realizations of the GAN.

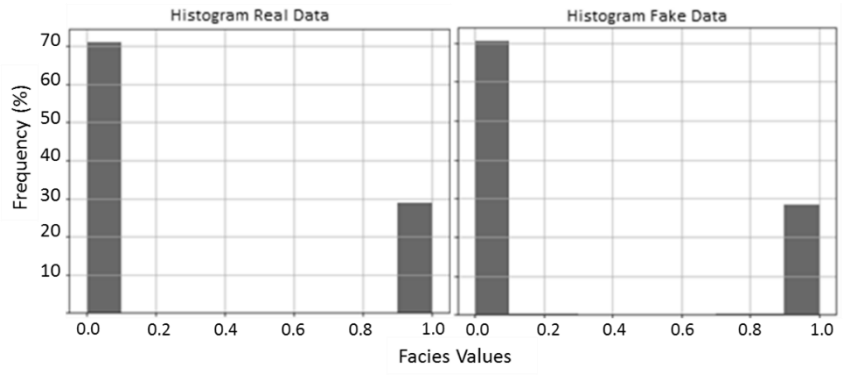

Figure 27. At the left side, the histogram composed of data from all the sliding windows – 34969 windows, the percentage of data representing channels, 1, is 28.97%. At the right, histogram composed of data from 34969 realizations of the GAN, the percentage of data representing channels, 1, is 28.85%.

One important aspect to highlight is the consistency of the resulting models in terms of the continuity of channels. The spatial connectivity is an important feature as it has a dramatic impact when running fluid flow simulation for reserve estimation. Also important is the fact that these models were generated from a training dataset with images of the same size of $64\,x\,64$ pixels, which is not possible with conventional geostatistical simulation conditioned to MPS. Usually, MPS geostatistical simulation methods require large

training images with a considerable amount of repetitions of the geological features of interest (Tahmasebi, 2018).

The ensemble of V_P models comprising the training dataset is composed of 1000 models. Figure 28a shows an example of the training dataset after being normalized between 0 and 1 (Table 1). In Figure 28b, the generated samples by GAN are presented. The comparison shows that the spatial patterns of the training dataset are reproduced in the generated models.

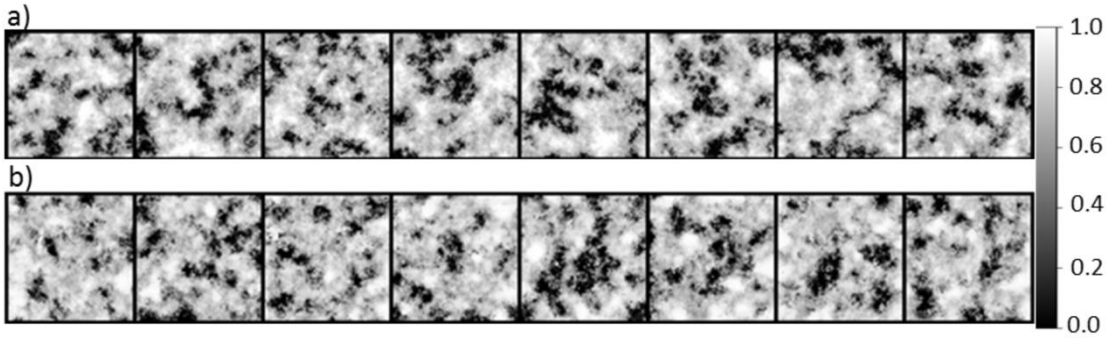

Figure 28.a) Real realizations of P-wave velocity of the training dataset. b) Realizations generated by the GAN.

4.2 Model Reconstruction

4.2.1 Facies Model Reconstruction

The dataset used for the facies model reconstruction was the semi-straight channels presented in section 4.1.1.1 and, the network, the same as the one used to produce the unconditioned samples.

After defining the experimental data to be used, it was developed a series of tests by changing two main parameters. The first parameter is the minimum acceptable accuracy between the conditioning data points and the actual output values, it was used 3 different accuracies: 90%, 95% and 100%. The second parameter is the region comprised by the conditioning data, it was considered three different situations: “square”, “wells” and “points”.

For each test 100 models were generated in order to compute the statistics and compare the statistics between training and generated data. From these 100 realizations, one realization was randomly selected to illustrate and discuss the results.

4.2.1.1 Generation of Conditioned Samples: Square

In this example, a reference image was pulled from the training dataset. Then, a square of fifteen by fifteen pixels was defined as conditioning data leaving the remaining pixels of the image to be simulated by the GAN (). This region represents pixels where we are confident about the underlying geology (e.g. samples with good match between synthetic and observed seismic data after seismic inversion).

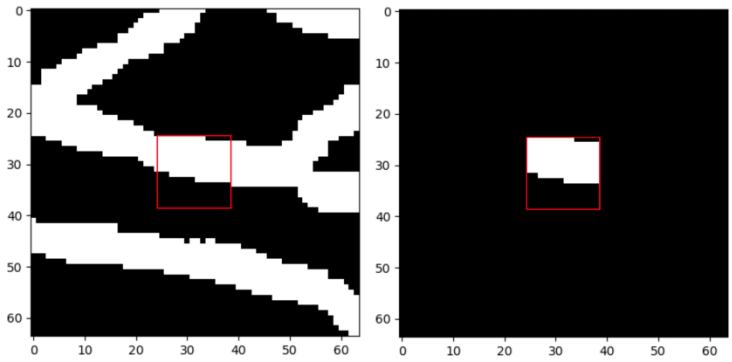

Figure 29. At the left, a random sliding window, which is the source for the conditioning data. At the right, the conditioning data, comprised in a 15x15 square.

Figure 30 shows two models generated with a minimum of 90% and 95 % match respectively. The reconstructed models are able to reproduce the channel connectivity around the set of conditioning data and also for the remaining of the image. These channels show the same spatial pattern as the training dataset. Unmatched samples are preferable located along the borders of the channels representing locations with higher uncertainty. The same effect can be interpreted from the facies probability model calculated by the occurrence of the sand facies (represented as white color) in a set of 100 models produced by the conditioned GAN (Figure 31). Also, high uncertain regions are preferable located along the limits of the channels. From a geological point of view, these results are consistent as the limits between both facies do represent areas of mixture between facies and therefore of high uncertainty. The GAN is also able to reproduce the main statistics of the reference image as depicted in Figure 32.

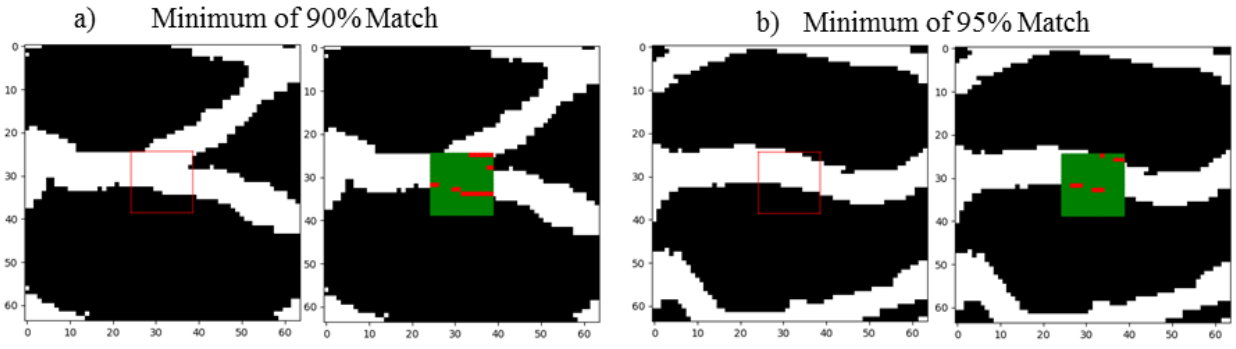

Figure 30. a) At the left, the realizations of the network with, at least 90% of accuracy between conditioning data and output. At the right, the indication of match (green) or mismatch (red) between data. b) At the left, the realizations of the network with, at least 95% of accuracy between conditioning data and output. At the right, the indication of match or mismatch regions.

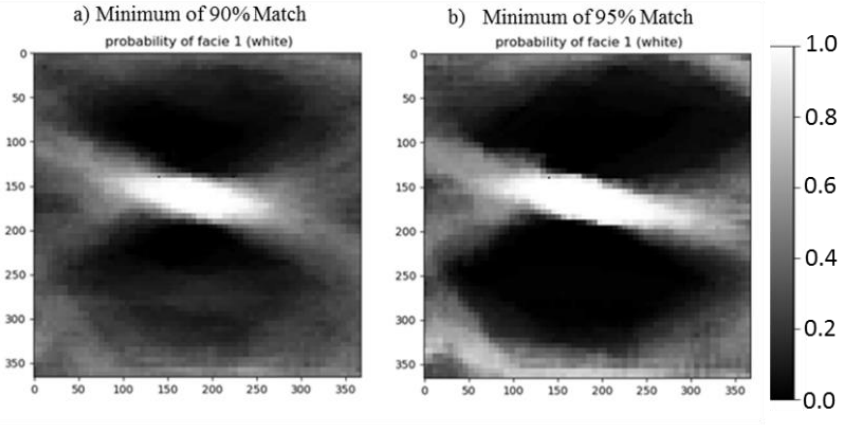

Figure 31.Location probability of appearing a facies with value one (white). a) for the 90% accuracy case and b) for the 95% accuracy case.

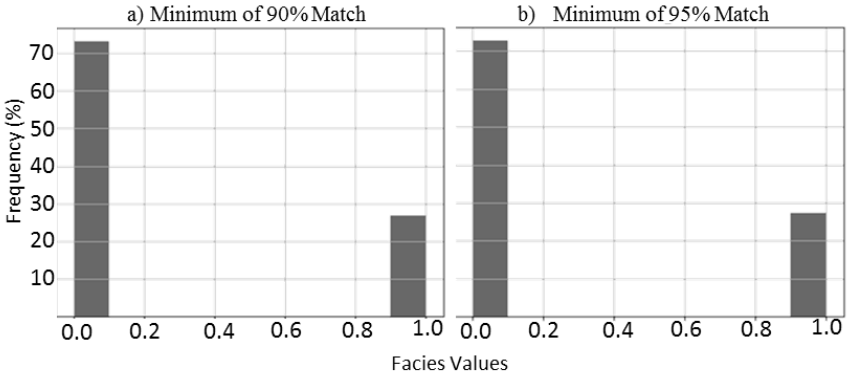

Figure 32. a) Histogram calculated for the 100 realizations with a minimum of 90% match, present a frequency of 26.92% for facies 1. b) Histogram calculated for the 100 realizations with a minimum of 95% match, present a frequency of 27.27% for facies 1.

4.2.1.2 Generation of Conditioned Samples: Wells

For model reconstruction conditioning to well data, two vertical wells located far from each other and close to each border of the model were considered (location 5 and 50 along the horizontal axis) (Figure 33). This is a typical example of spatial inference problem, where geostatistical simulation algorithms are the preferable tools in geosciences using existing well data as experimental data. In these methods we need to explicitly provide a spatial continuity model either a variogram model or a training image. Here, we intend to show the ability of GANs to generated equivalent models using a much more richer training dataset when compared with the traditional approaches.

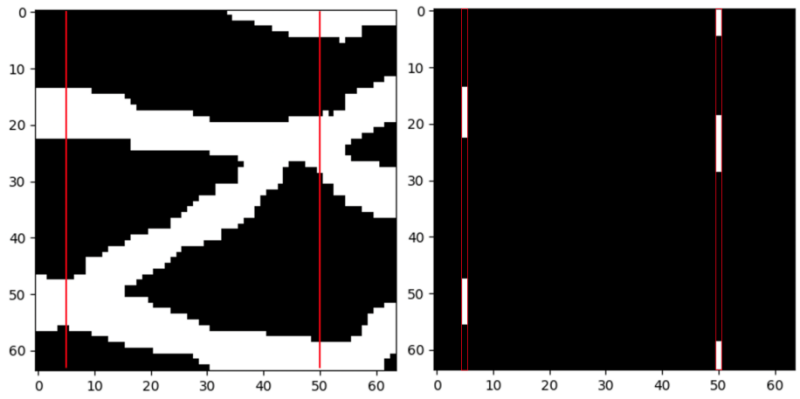

Figure 33. At the left, a random sliding window, which is the source for the conditioning data. At the right, the conditioning data, two wells located at the 5 and 50 positions on the x axis.

Figure 34 illustrates three different realizations for a minimum of accuracy of 90%, 95% and 100%, respectively. While the match is considerably good in all three realizations, the models are also able to reproduce the geological continuity of the channels and reproduce the main spatial patterns existing in the training dataset. This fact is highlighted in the probability model of facies occurrence computed from a set of 100 realizations (Figure 35). The use of well information does not impact the reproduction of the main statistics (Figure 36).

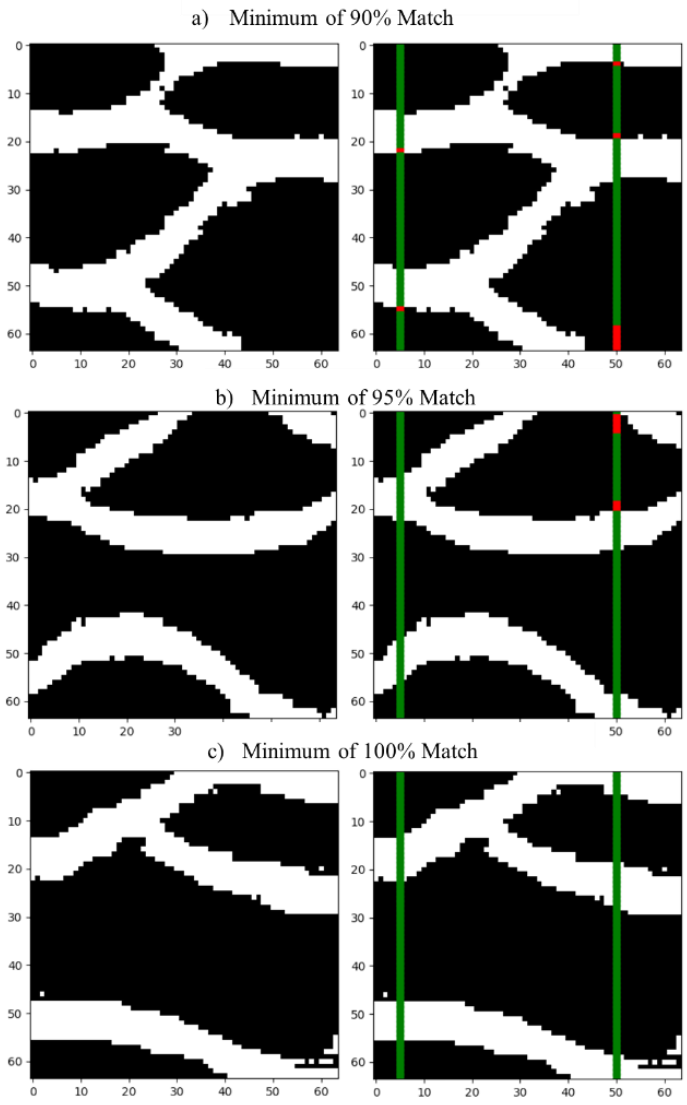

Figure 34. a) At the left, the realizations of the network with, at least 90% of accuracy between conditioning data and output. At the right, the indication of match or mismatch. b) Realizations of the network with, at least 95% of accuracy between conditioning data and output. At the right, the indication of match or mismatch. c) the realizations of the network with 100% of match. At the right, the indication of the perfect match along the well.

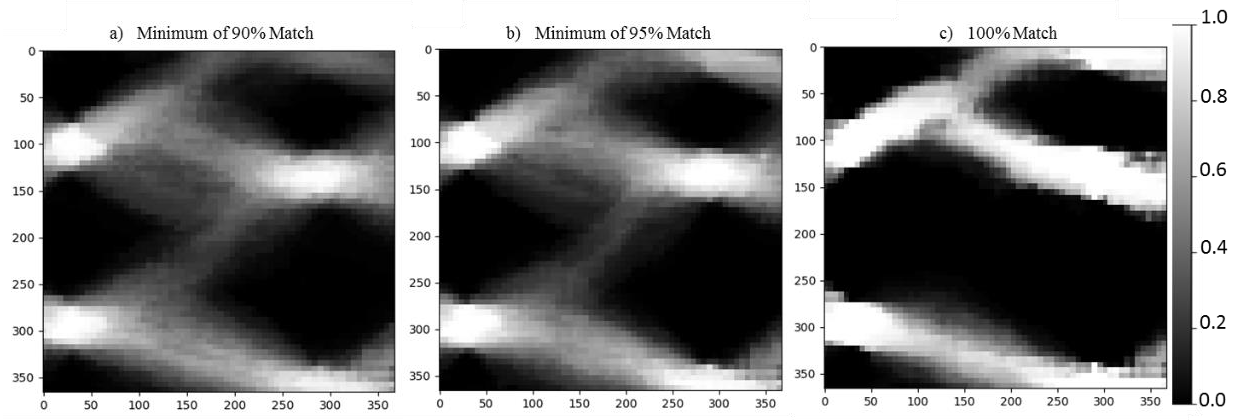

Figure 35.Location probability of appearing a facies with value one (white). a) for the 90% accuracy case, b) for the 95% accuracy case and c) for the 100% match case.

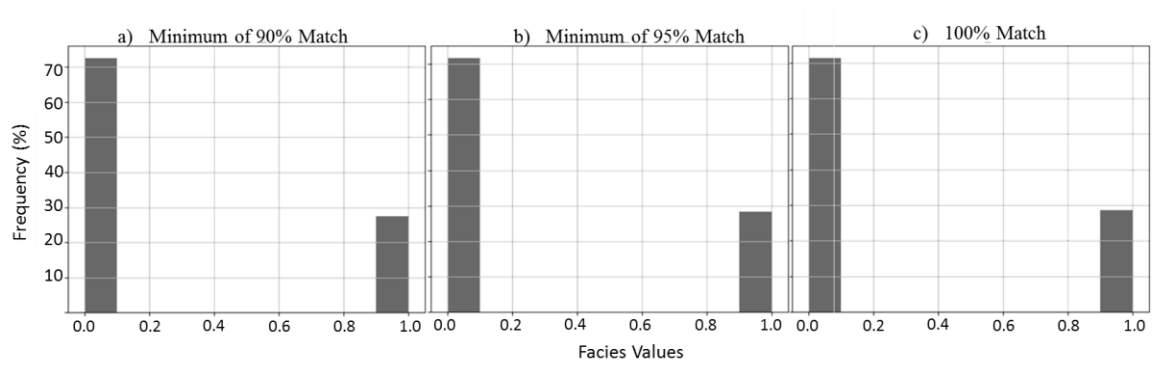

Figure 36. a) Histogram calculated for the 100 realizations with a minimum of 90% match, present a frequency of 27.51% for facie 1. b) Histogram calculated for the 100 realizations with a minimum of 95% match, present a frequency of 28.39% for facie 1. c) Histogram calculated for the 100 realizations with 100% match, present a frequency of 28.67% for facie 1.

4.2.1.3 Generation of Conditioned Samples: Points

In this scenario 0.5% of the grid (21 pixels) were used as experimental data. Figure 37 shows the conditioning data with the 21 data points randomly selected from a training images for 90%, 95% and 100% accuracy tests. Figure 38 shows realization for accuracy 90%, 95% and 100%.

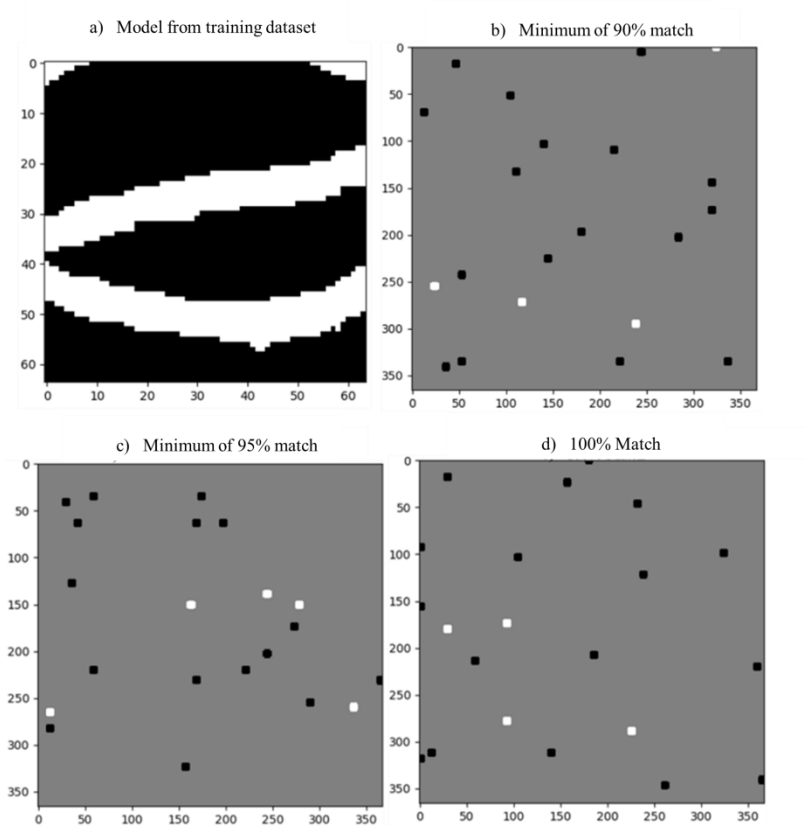

Figure 37. a) a random sliding window, followed by the conditioning data of each test. b) Conditioning data for a minimum of 90% accuracy test. c) Conditioning data for a minimum of 95% accuracy test. d) Conditioning data for 100% accuracy test.

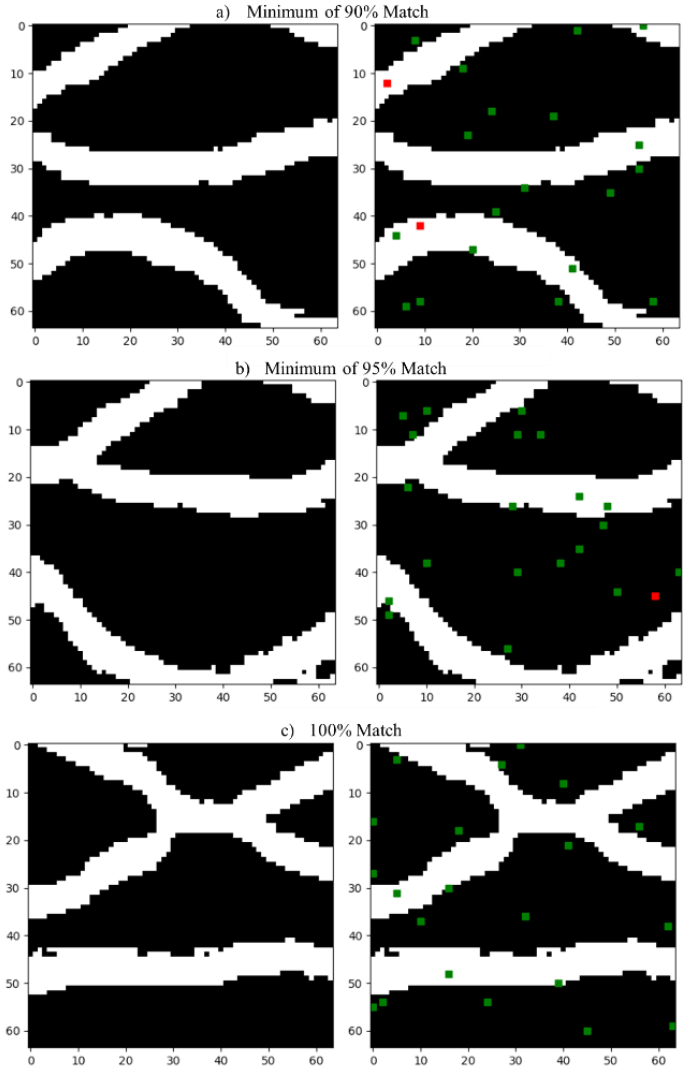

Figure 38. a) At the left, the realizations of the network with, at least 90% of accuracy between conditioning data and output. At the right, the indication of match or mismatch. b) Realizations of the network with, at least 95% of accuracy between conditioning data and output. At the right, the indication of match or mismatch. c) the realizations of the network with 100% of match. At the right, the indication of perfect match for all conditioning points.

While the match is considerably good in all three realizations, the models are also able to reproduce the geological continuity of the channels and reproduce the main spatial patterns existing in the training dataset. This fact is highlighted in the probability model of facies occurrence computed from a set of 100 realizations (Figure 39). The use of well information does not impact the reproduction of the main statistics (Figure 40).

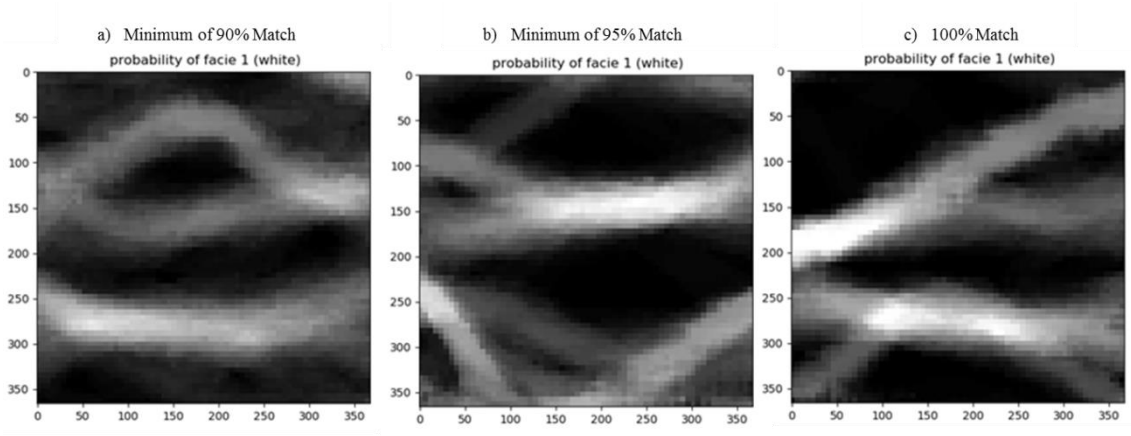

Figure 39.Location probability of appearing a facies with value one (white). a) for the 90% accuracy case, b) for the 95% accuracy case and c) for the 100% match case.

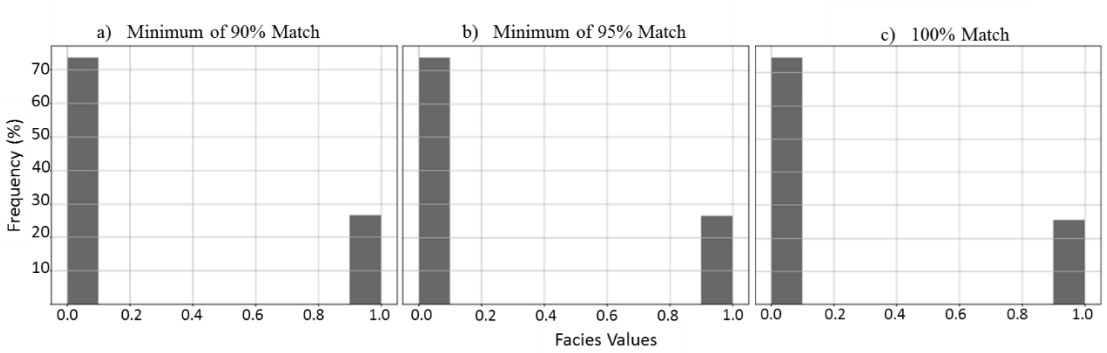

Figure 40. a) Histogram calculated for the 100 realizations with a minimum of 90% match, present a frequency of 26.51% for facies 1. b) Histogram calculated for the 100 realizations with a minimum of 95% match, present a frequency of 26.33% for facies 1. c) Histogram calculated for the 100 realizations with 100% match, present a frequency of 25.51% for facies 1.

4.2.2 P-Wave Velocity Model Reconstruction

The dataset used for continuous model reconstruction was the P-wave velocity models presented in section 4.1.1.2 and, the network, the same as the one used to produce the unconditioned samples.

The tests for the generation of conditioned samples of continuous variables followed a procedure similar to the tests involving the discrete data set. The conditioning regions are the same, the difference is the stopping criteria for each case. Since it is a problem that deals with continuous data, it was chosen a squared error criterium. To evaluate the performance of the network in each scenario, it was performed test with six different stopping criteria: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$. For each scenario, it was performed 5 or 4 tests, each one with different conditioning data, to evaluate the performance of the network.

4.2.2.1 Generation of Conditioned Samples: Square

In Figure 41, it is shown the conditioning data of ten by ten pixels, with a MSE of 10^{-3} meaning that the maximum squared error of the actual output should be equal to 10^{-1} due to the 100 conditioning points. In addition to the small number of samples, the square only has information about high Vp values introducing a potential bias in the predictions. This is a classical example where geostatistical modelling tools would struggle as they are developed to reproduce exactly the information provided by the set of experimental data.

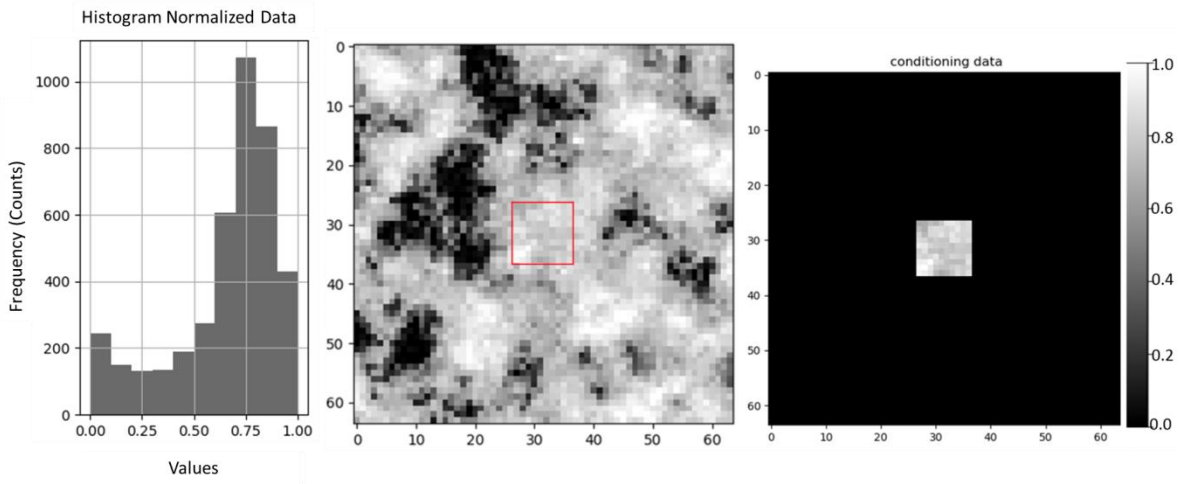

Figure 41. (left), The histogram of a random sliding window and the respective model, which is the source for the conditioning data. At the right, the conditioning data, comprised in a 10x10 square.

Figure 42 shows a generated model and the respective histogram and variograms. It is important to notice that the data distribution was preserved. From the histogram, which is more complex than the discrete database, the distribution is similar.

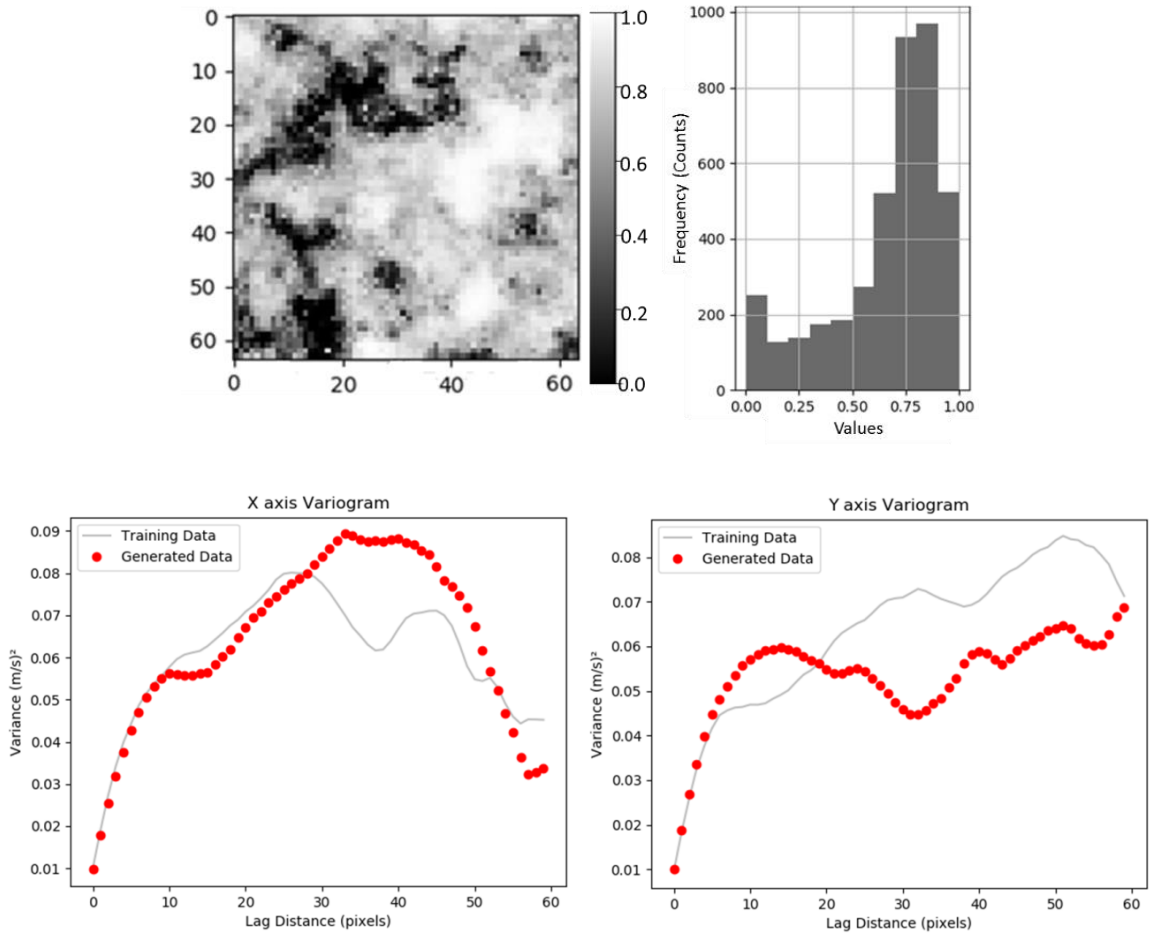

Figure 42. Example of a generated model and the respective histogram and variograms.

Figure 43 shows the squared error of the generated image and the difference between the desired and actual output.

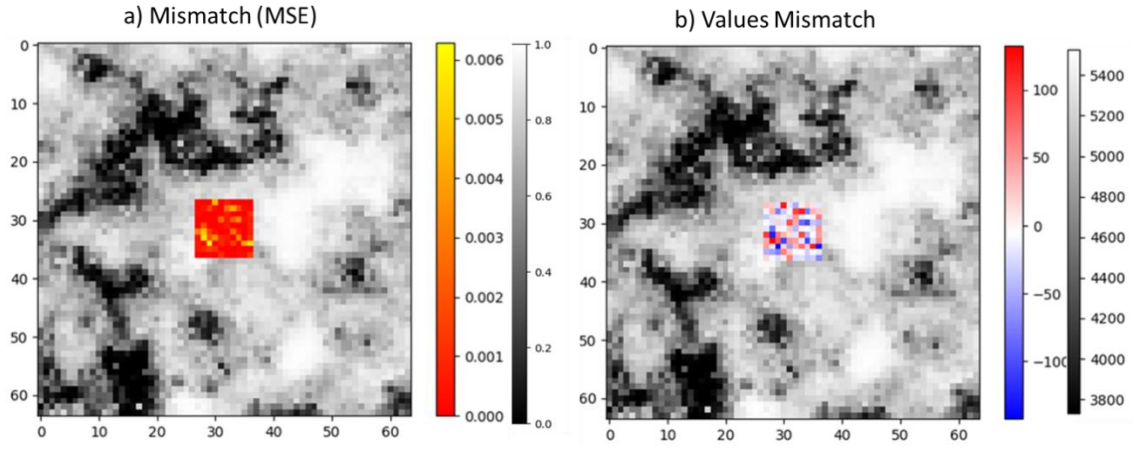

Figure 43. At the left, the mean squared error between the conditioning data and the actual output value. At the left, the difference, in velocity scale (m/s), between the conditioning data and the output.

Figure 44 shows the mean and the variance of the 100 realizationa of the test. The mean model resembles a kriged model, or the E-type model, of a set of geostatistical realization where the predicted values far from the location of experimental data tend to be close to the average value of all data. Also, the variance has a similar behavior to conventional geostatistical simulation with low variance close to the locations of the conditioning data.

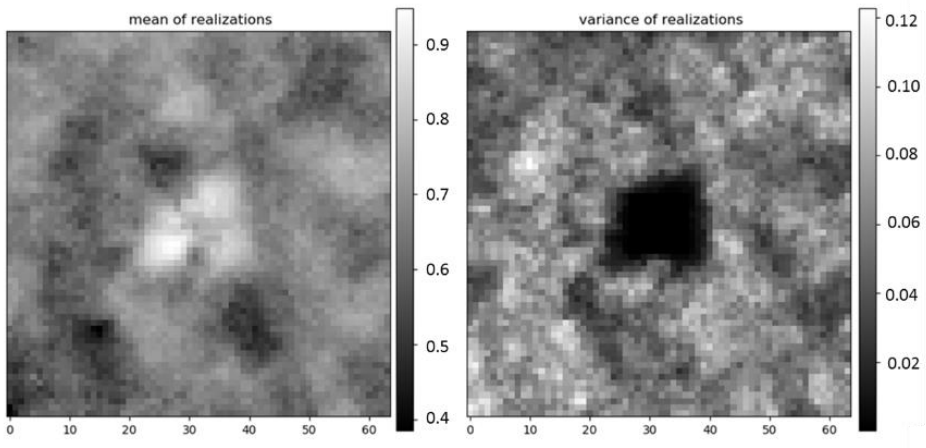

Figure 44. At the left, the mean of the realizations. At the right, the variance of the realizations.

4.2.2.2 Generation of Conditioned Samples: Wells

In total, it was developed 28 tests for this conditioning region, resulting from a combination of the MSE and the 3 different set of well locations.

Figure 45 shows the conditioning data with two wells located near the border of the model, in position x=5 and x=50, with a MSE of 10^{-2} meaning that the squared error of the actual output should be equal to 1.28 due to the 128 (2 wells with a length of 64 pixels) conditioning points. Figure 46 introduces the generated model and the respective histogram. Figure 47 shows the MSE of the generated image and the difference between the desired and actual output. Figure 48 shows the mean and the variance of the 100 realizations of the test.

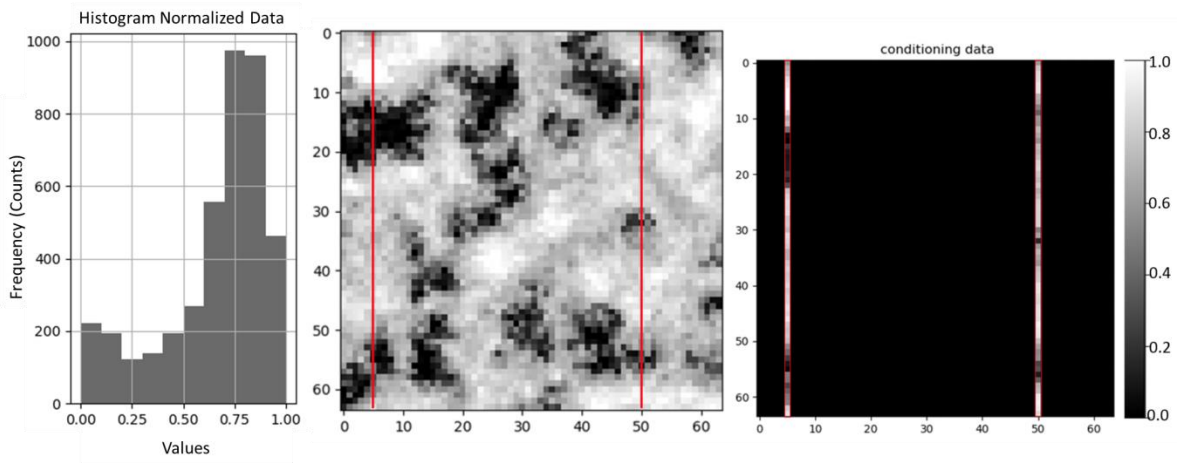

Figure 45. (left), The histogram of a random sliding window and the respective model, which is the source for the conditioning data. At the right, the conditioning data, two wells located at the 5 and 50 positions on the x axis.

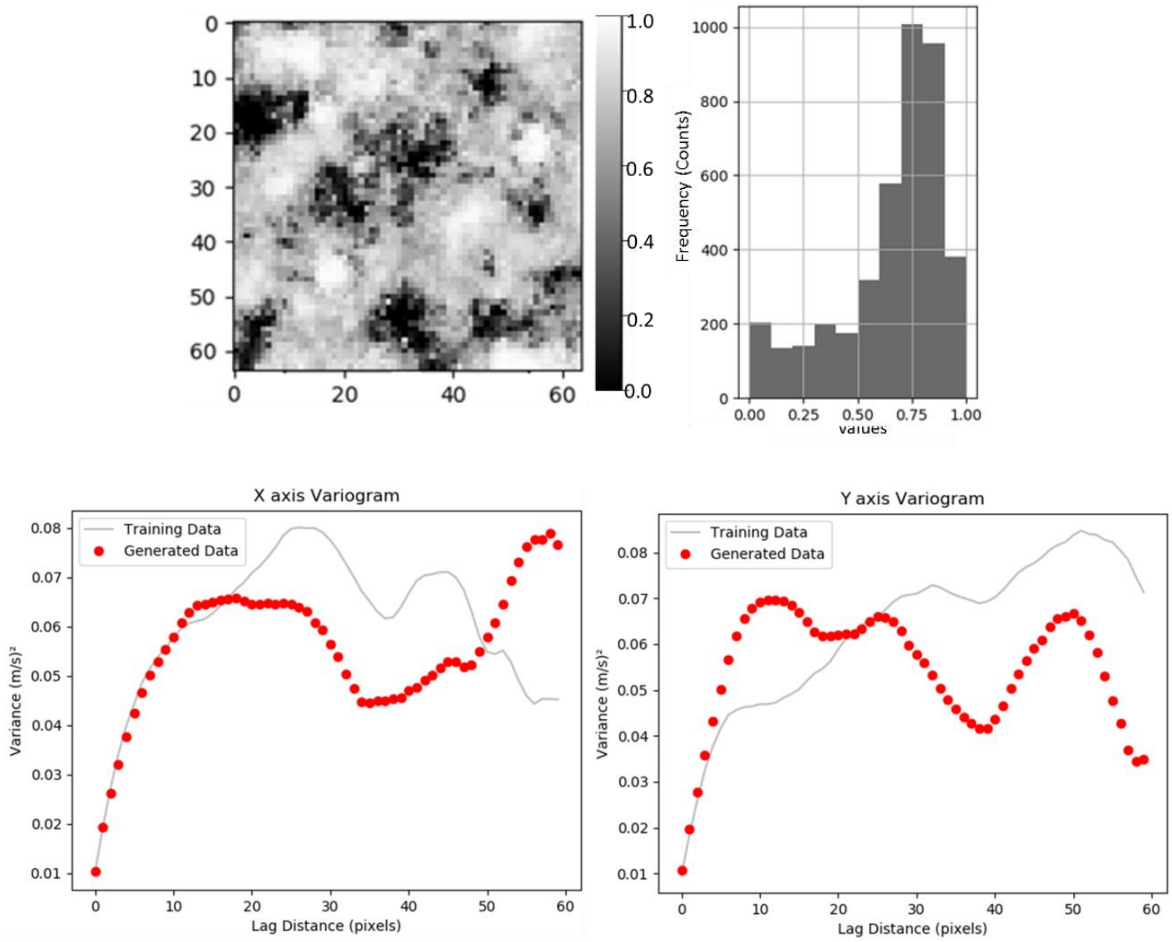

Figure 46. Example of a generated sample and the respective histogram and variograms.

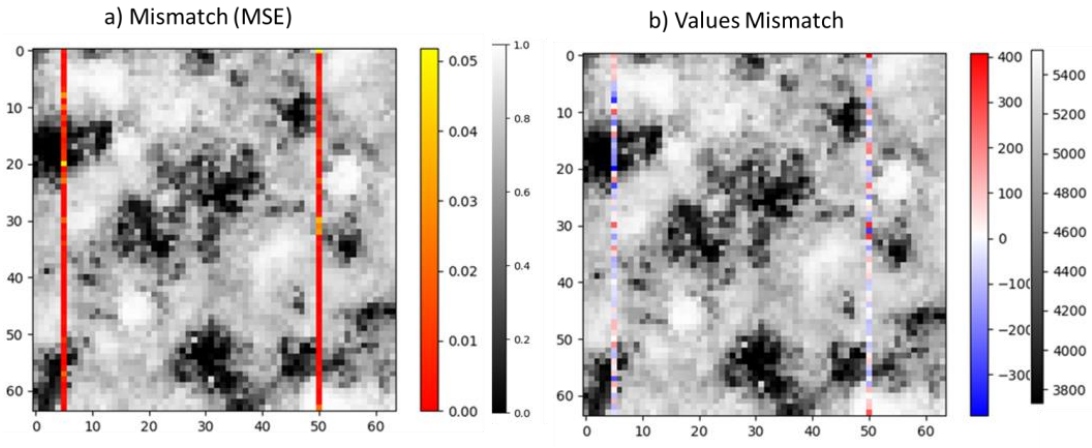

Figure 47. At the left, the mean squared error between the conditioning data and the actual output value. At the left, the difference, in velocity scale (m/s), between the conditioning data and the output.

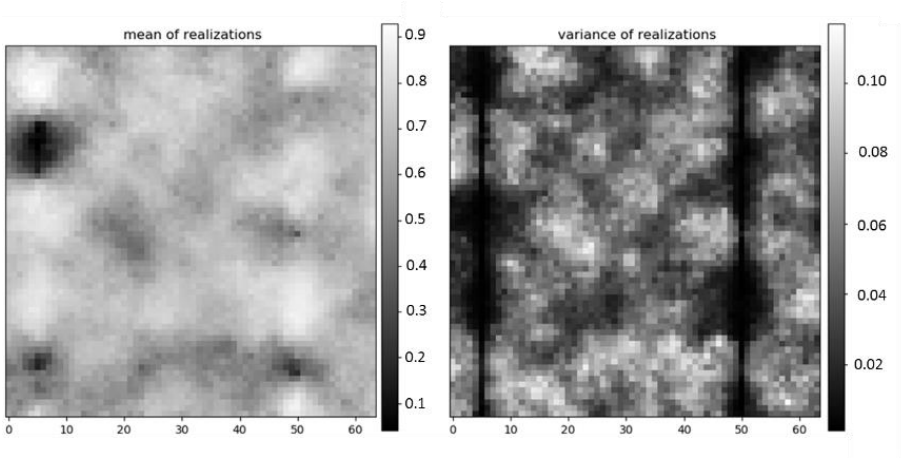

Figure 48. At the left, the mean of the realizations. At the right, the variance of the realizations.

4.2.2.3 Generation of Conditioned Samples: Points

Figure 49 shows the conditioning data with 1% percent of the image points as conditioning data, with a MSE of 10^{-5} meaning that the maximum squared error of the actual output should be equal to $4.1 * 10^{-4}$ due to the 41 conditioning points. In Figure 50, it is presented a generated sample and the respective histogram. Figure 51 shows the MSE of the generated image and the difference between the desired and actual output. Figure 52 shows the mean and the variance of the 100 realizations of the test,

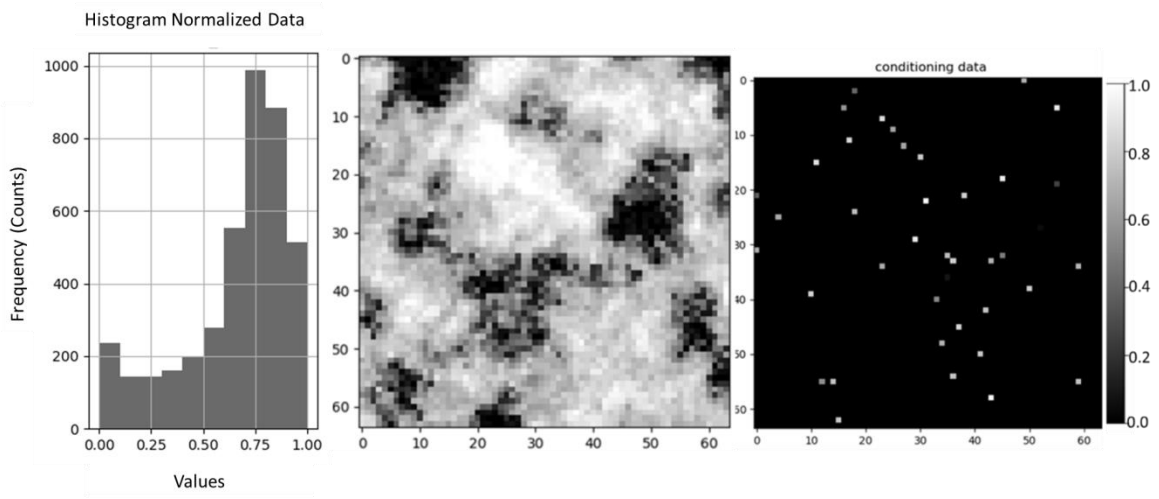

Figure 49. (left), The histogram of a random sliding window and the respective model, which is the source for the conditioning data. At the right, the conditioning data, 41 randomly located points.

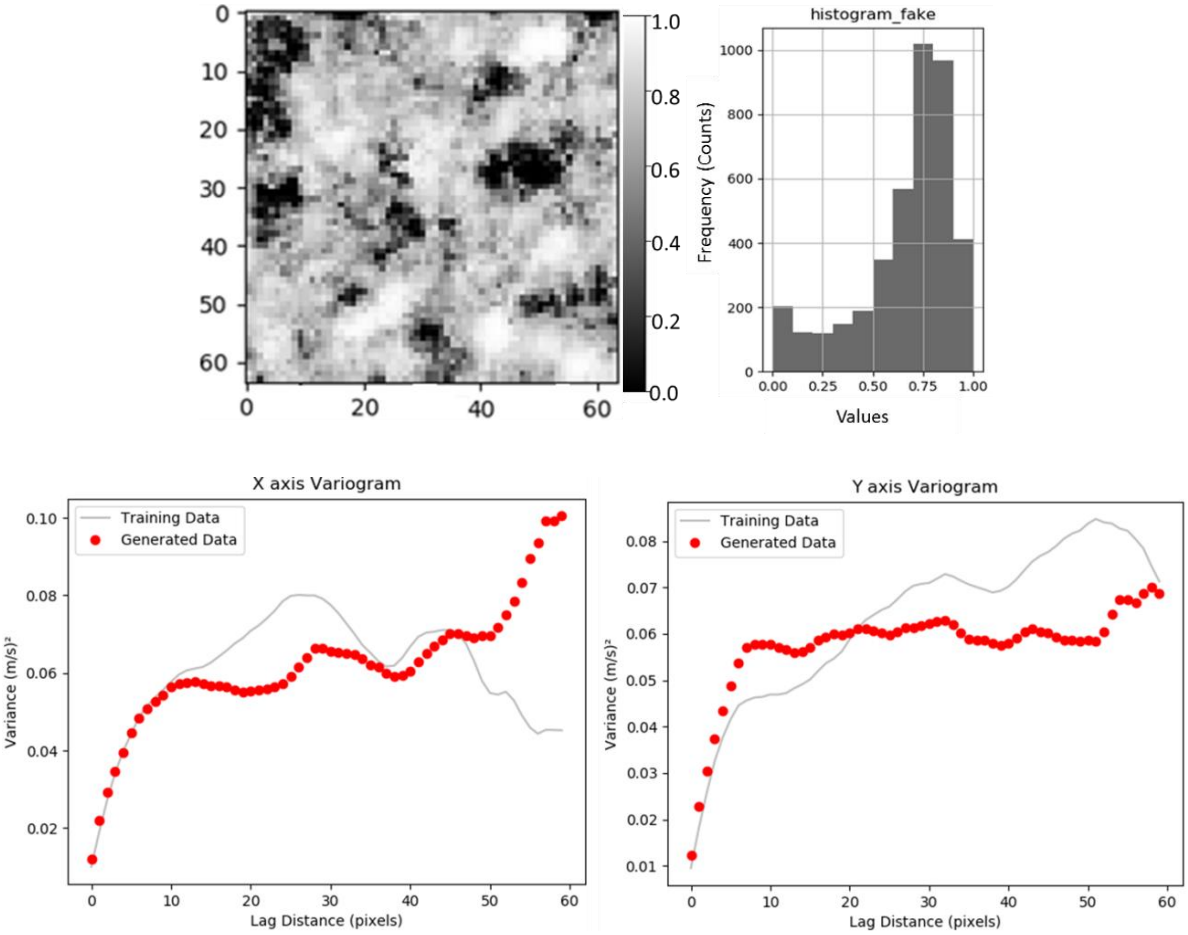

Figure 50. Example of a generated sample and the respective histogram and variograms.

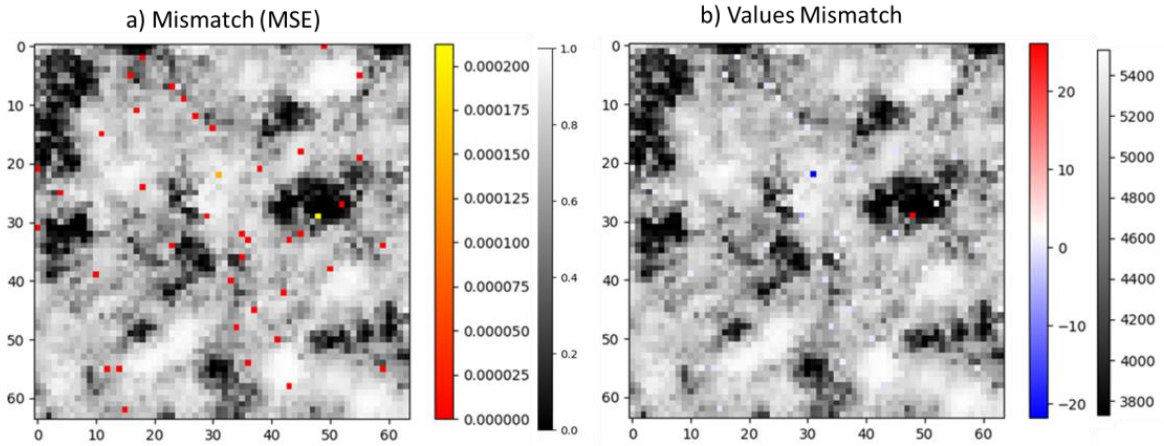

Figure 51. At the left, the mean squared error between the conditioning data and the actual output value. At the left, the difference, in velocity scale (m/s), between the conditioning data and the output.

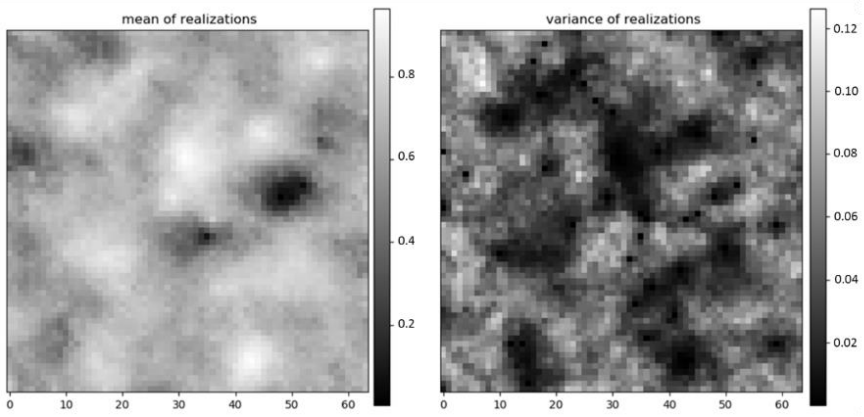

Figure 52. At the left, the mean of the realizations. At the right, the variance of the realizations.

Analyzing the final models produced by the GAN (Figure 42, Figure 46, Figure 50) it is possible to assess the reliability of the GAN in reproducing the main statistics and the spatial continuity pattern of the set of realizations comprising the training dataset. The final models, presented in Figure 43, Figure 47 and Figure 51 display a maximum error of plus or minus 500m/s for the conditioning points, which is a tolerable deviation for the characterization of the subsurface velocity. The display of the mean values and the variances of the 100 realizations in (Figure 44, Figure 48, Figure 51) shows that outside the conditioning regions the pixels have more values with higher variance than around the conditioning region. Also, the mean of the values of the 100 realizations in regions outside the conditioning area is around the mean of the dataset, these characteristics are also observed in models produced by stochastic simulations. Regarding the variograms, it can be seen that until the range of the variograms models are reached, the experimental variograms of the generated model follow the experimental variogram of the training sample from which the conditioning data was selected.

4.3 GAN in Seismic Inversion

This section of the thesis introduces a geostatistical seismic inversion method that uses GAN as facies model perturbation. The proposed approach consists of an iterative procedure. First, the trained GAN generates an unconditioned facies model that is then optimized using the algorithm proposed in section 3.2

4.3.1 Dataset Description

The dataset is composed of three main data: the seismic model, the set of experimental data of acoustic impedance and the facies model. It is important to point to the fact that this is a synthetic case study.

To train the network, it was used a facies data of dimensions $200\,x\,200$. The model has two generic lithological facies: the sand channels, in black, and, in white, the background shaly facies. The network was trained using sliding windows of $128\,x\,128$ over the true facies model. Some examples can be seen in Figure 53.

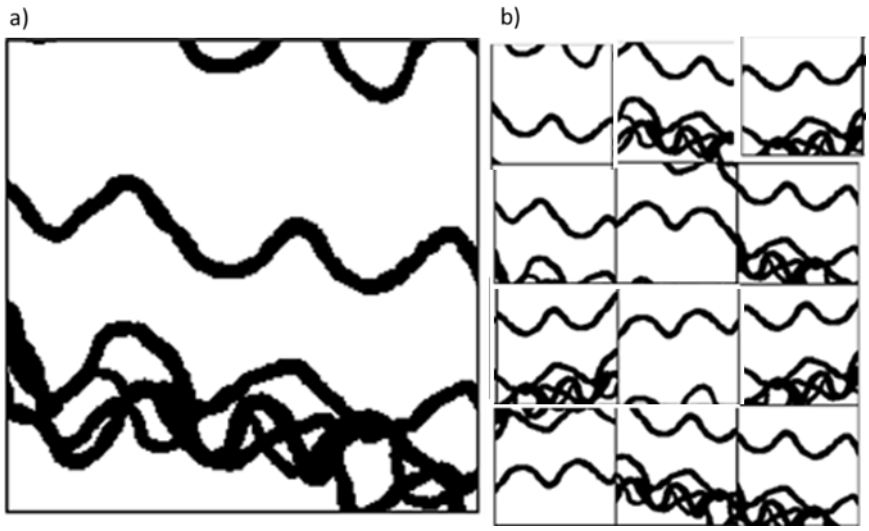

Figure 53. a) Facies Model. b) Sliding windows of dimension $128x128$.

The seismic model presents a dimension of $200\,x\,200$ pixels. Figure 54 presents the seismic model, this seismic section is referent to the facies model shown in Figure 53.Therefore, this section was used during the case studies and, when necessary, it was sliced in windows of $64\,x\,64$ or $128\,x\,128$. The values of the seismic, original or synthetic, is always presented as a normalized value.

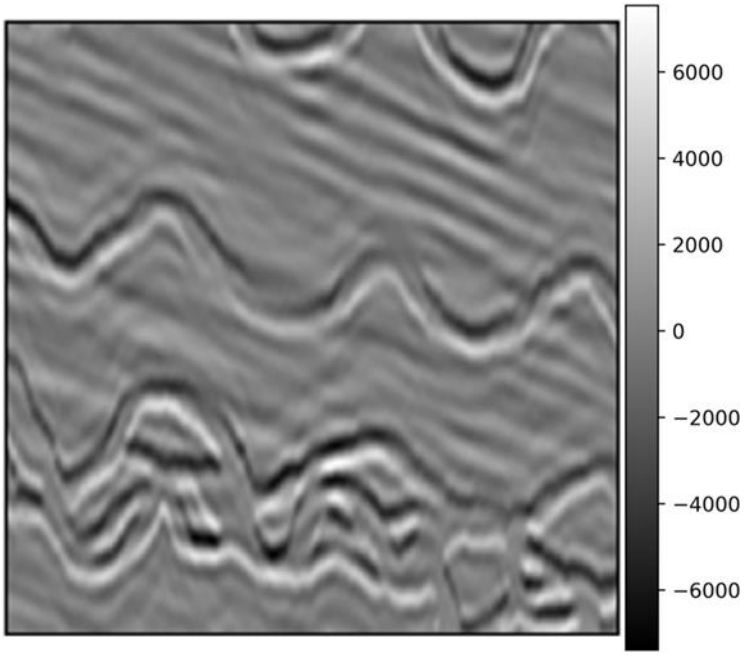

Figure 54. Seismic model.

Another data necessary is the acoustic impedance data. This data was filtered by facies value using the facies information of the facies model previously presented in order to be able to adjust variogram models for each facies class. Therefore, it was created two distinguished hard data sets, one corresponding to the Ip data of the channels and one to the second lithological facies. In Figure 55 is presented the acoustic impedance data related to the section above and from which the seismic section was computed.

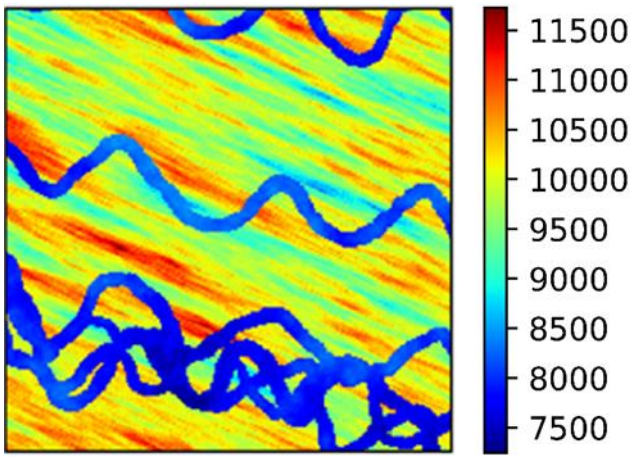

Figure 55. Synthetic acoustic impedance of the section.

4.3.2 Case Study - 128x128

This section presents the results of applying the methodology described in section 503.2 to a model of dimension 128 x 128 pixels. Figure 56 show the data used for this case, 128 by 128 pixels models cropped from the original dataset.

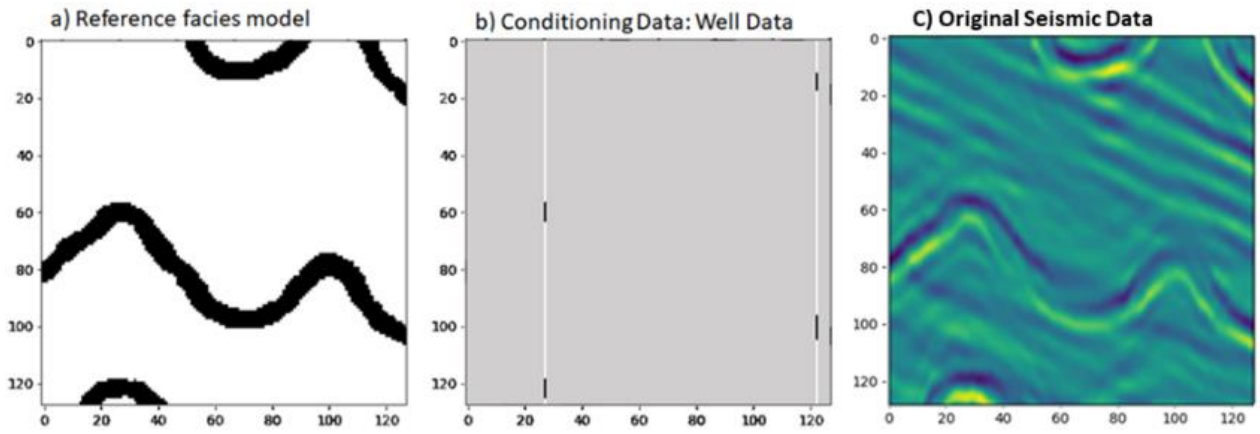

Figure 56. Presentation of the a) facies model, b) the conditioning data of the well locations (x=28, x= 123) and c) the seismic amplitudes of the selected region of the model.

The first step was to use the GANs to created a set of 50 unconditioned realizations of facies models. The set of unconditioned realizations are shown in Figure 57.

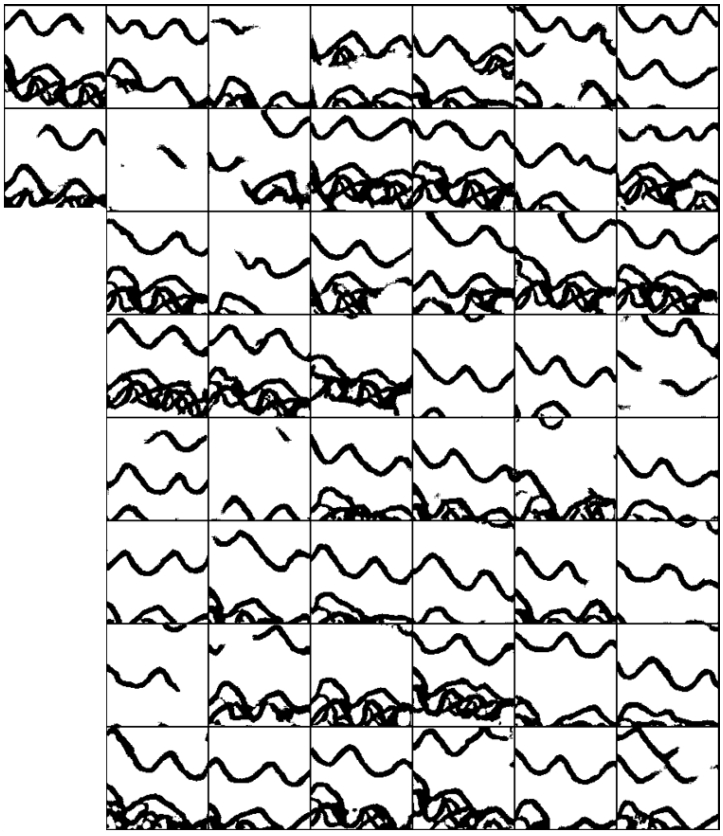

Figure 57. Set of 50 unconditioned realizations of facies models (models are rotated 90º to the left).

The next step was to input each one of the created model through the Adapted GSI algorithm. For each seismic was created a respective synthetic seismic. As part of the Adapted GSI algorithm each facies

model is assigned with a best correlation model. Using these models, plus the conditioning data of the well, it is created a conditioning model for the next iteration (Figure 58).

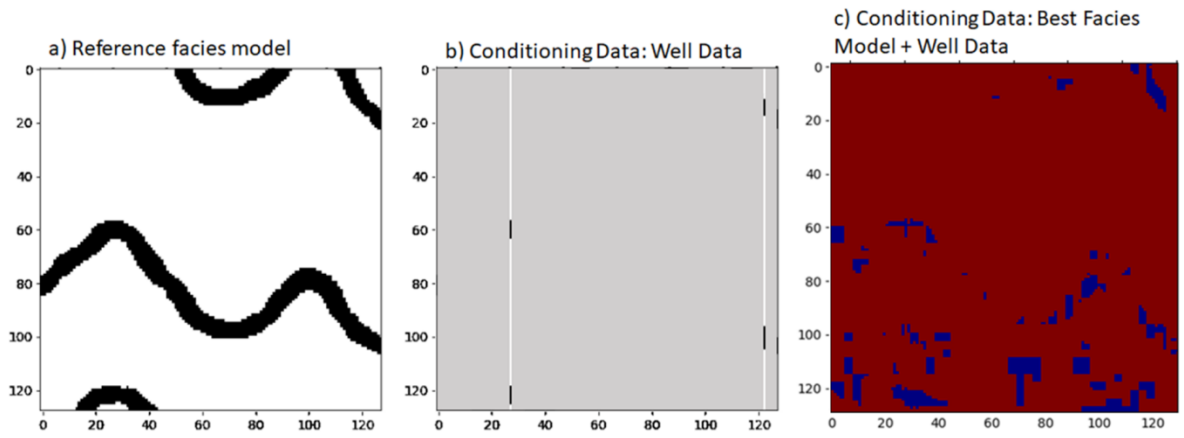

Figure 58. a) Reference facies model. The values of the facies at the well locations are based on these models. b) the facies values selected as conditioning data at well locations (x=28, x=123). c) best facies model created during first iteration. This model will be used to calculate the content of the current facies models.

Then, it is computed the losses of all the facies model. As it is complicated to evaluate the losses of all the models, it was chosen to show in detail the model that presents lower seismic loss, higher correlation between original and synthetic seismic.

For this optimization, the threshold values defined were 90% of accuracy for the facies values in well locations and 90% of correlation between synthetic and original seismic. However, the accuracy threshold for facies values was not achieved, only for the seismic correlation. In Figure 59, it is possible to see the evolution of the optimization of the facies models along the iterations. In Figure 60, it is shown the best facies model, with lowest seismic loss.

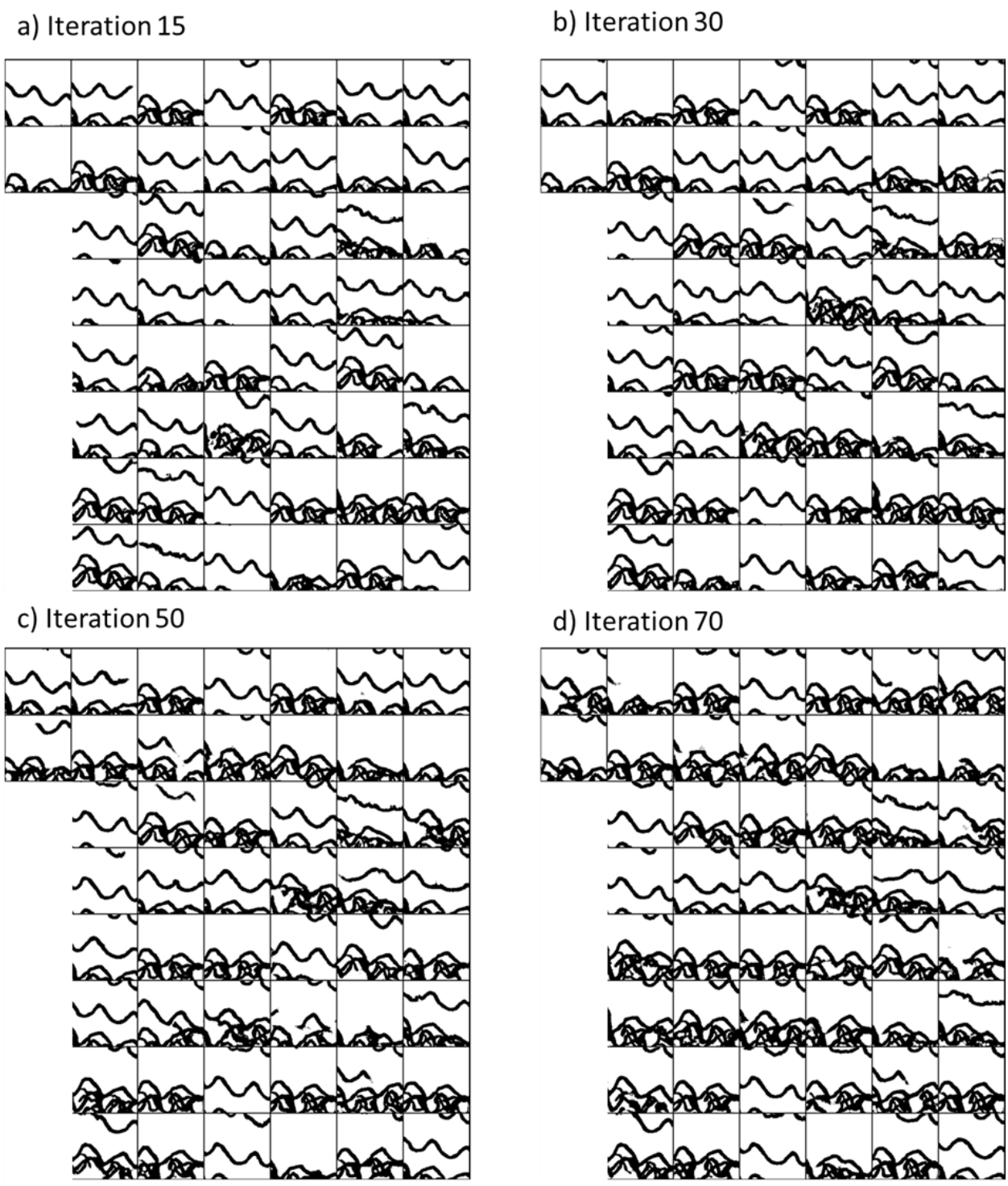

Figure 59. Examples of the optimization of the initial facies models during: a) iteration 15, b) iteration 30, c) iteration 50 and d) iteration 70 (models are rotated 90º to the left).

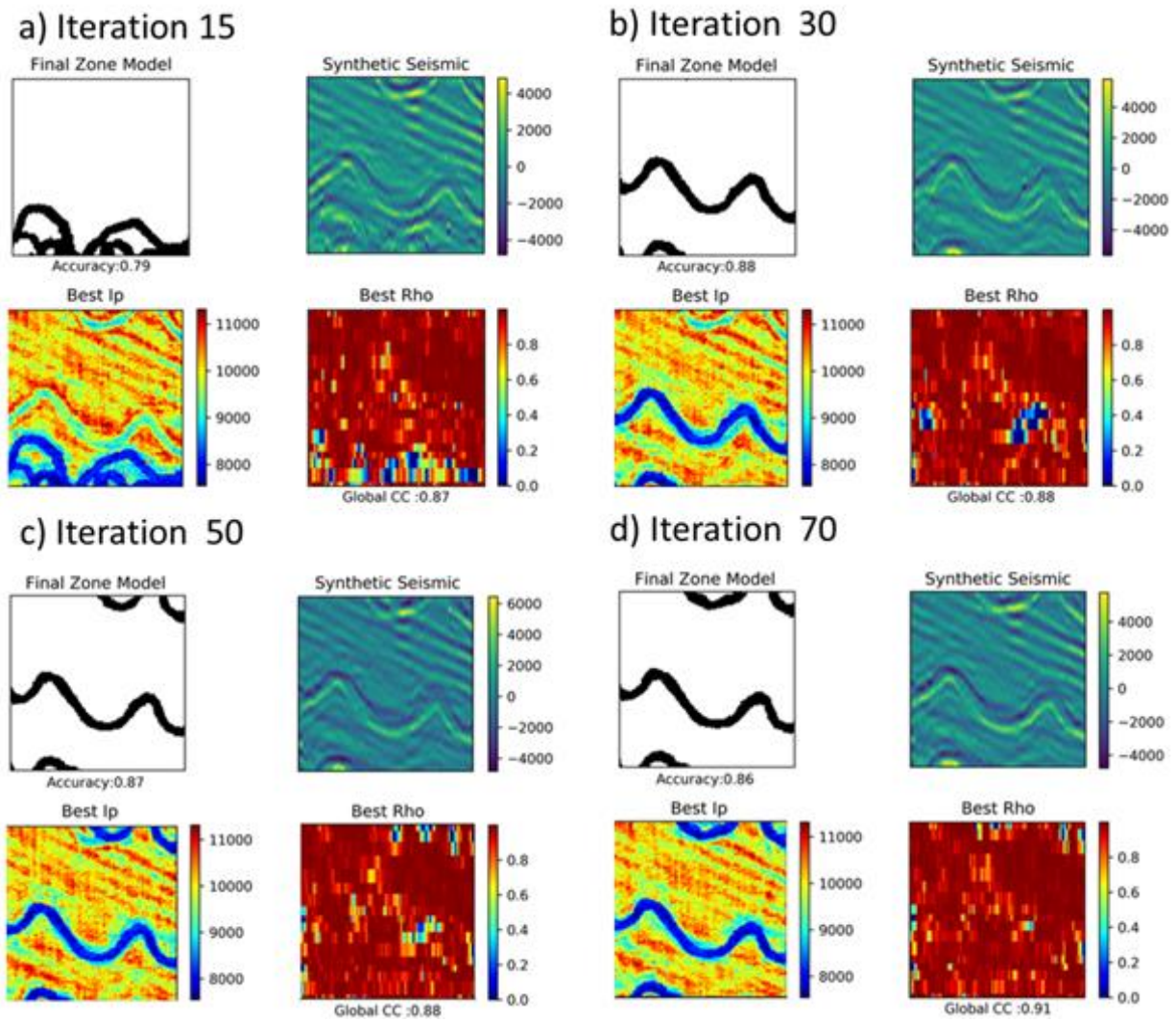

Figure 60. Examples of the best facies models of each iteration: a) iteration 15, b) iteration 30, c) iteration 50 and d) iteration 70.

In iteration 70, the considered best model of the current iteration surpass the threshold of 90% of correlation defined.

In Figure 61 it is possible to see the evolution of the best facies model from the first iteration until the last iteration. It is possible to see that the channels structures are reproduced in this model.

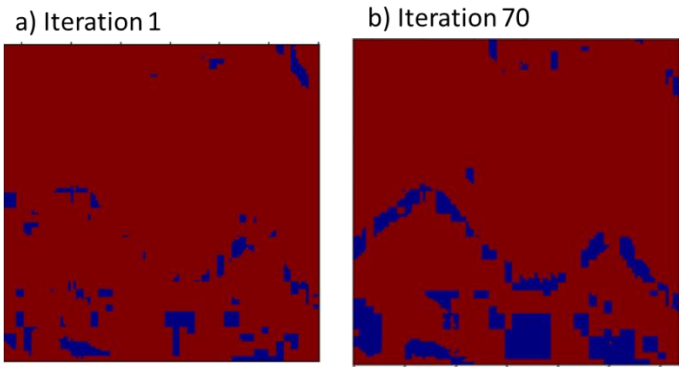

Figure 61. Examples of the conditioning data model: a) iteration 1 and b) iteration 70.

4.4 Final considerations and discussion from the obtained results

Regarding the creation of subsurface models, it has been shown that the generative adversarial network is a reliable method for generating subsurface models. It has been seen that the models generated through this method not only reproduce the spatial pattern of the training dataset, but also reproduce the statistics that are intrinsic to the dataset.

One of the main advantages of using the GAN as a parameterization method is that it was able to produce conditioned data with different kinds of conditioning data and accuracy, as seen in section 4.2. Another advantage is that, contrary to the conventional geostatistical methods, it is not necessary to define the parameterization of the dataset, the network learns the data distribution by itself and is able to handle the conditioning data in different scenarios. Another mention that is important to do is that, as seen in section 4.2, the means of the realizations outside the conditioning regions are around the mean value of the training dataset and that the variance outside this areas also increases, which is a characteristic that is also observed in stochastic simulations.

From a subsurface modelling point of view, besides the ability to reproduce spatial patterns such as channels, we do need to effectively incorporate sparse well information as these data is the only source of direct measurements about the subsurface properties of interest and represent locations with lower uncertainty about our spatial predictions. However, the use of GAN to produce subsurface models has another advantage, these examples show that depending on the reliability of the existing experimental data (e.g. data might be contaminated with a great percentage of noise) we can increase or decrease the amount of its reproduction in the simulated models. This particular feature is not taken into account in traditional geostatistical modelling as experimental data is most often used as hard-data without considering uncertainty.

Regarding the use in GANs in seismic inversion, it is a method still in development. The results presented here have shown that the method worked as expected in a simple case. From the seismic data, it was

possible to invert to a facies and acoustic impedance models in which the channel structure is clearly recognized.

However, some issues have to be addressed. The first fact is that the best facies model used as conditioning data, may not present reliable facies values. As we had a facies reference model, it was possible to see that the conditioning model (Figure 61b) assigns channels values as the best corresponding facies values that are not corroborating with the reference facies model (Figure 58a). In theory, this model was just used to provide an auxiliary loss, since at the first try, for the loss function it was used only the mismatch at the well location and the information of correlation between seismics. It happens that, when there are no channels, only non-channels, the structure of the original seismic is reproduced in the synthetic seismic without the interference of the channels, therefore, creating a false high global correlation between the models as can be seen in Figure 62. And the best facies model was developed in attempt to introduce an auxiliary loss.

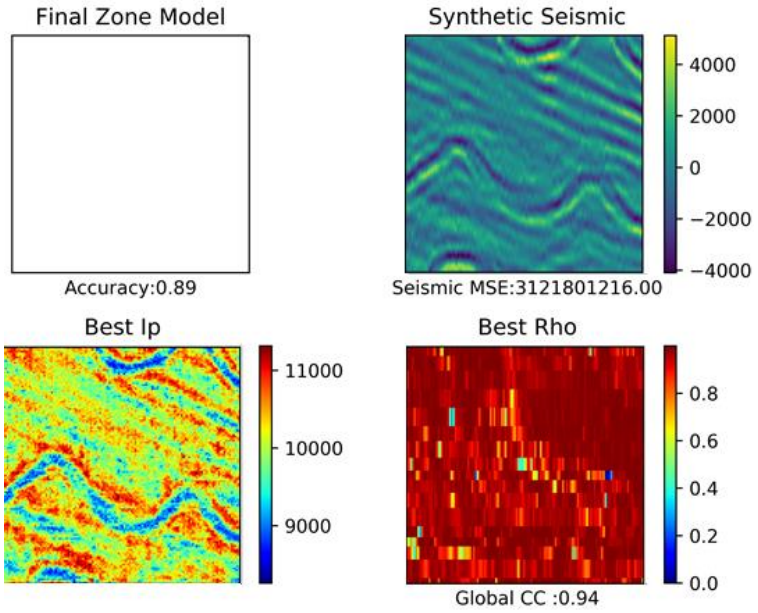

Figure 62. Example demonstrating that a realization containing no channel facies values provides high correlation between synthetic and original seismic. Therefore, it is not possible to just rely in the correlation coefficient for the optimization.

5 Conclusions and Future Work

The application examples shown here shows the ability of generative adversarial networks to produce reliable subsurface geological models in both discrete and continuous domains, reproducing the spatial patterns and main statistics of the training dataset. Regarding the discrete case, it is important to point to the fact that the continuity of the channel was reproduced, which is an important factor for reservoir modeling. However, it is possible some artifacts like pixel noise and endpoint of channels.

Regarding the use of GAN as a model reconstruction technique, the results produced here with generative adversarial networks show that resulting models are able to incorporate different sources of information with different levels of detail and accuracy without the need to fully explicit the geological parameterization as in the case of geostatistical simulation. Training the network with sets of geostatistical simulations allows to incorporate simultaneously the spatial continuity patterns of the training dataset with local and sparse conditioning data without compromising their performance.

Regarding the use of GANs in a seismic inversion procedure, it has been seen that the idea is possible to be applied, although more complex than the previous case. The challenge was to incorporate the different information from mismatch of seismic and facies values for the optimization of the facies model. The approach using an adapted GSI algorithm succeeded. The use of a best facies model, originated from the seismic inversion, to condition the facies models of the next iterations was the key factor. This optimization of the facies framework was enabled by the GAN framework, which generates a batch of models and optimizes them altogether, this mechanism enabled the update of the best facies model as the GAN was optimizing the facies model, producing facies models that presented lower values for the loss function.

One of the points of using the GAN in seismic inversion method is that it is a method that works with a global perturbation, which means that, as it is, this method cannot be applied to limited regions with low correlation coefficients.

Analyzing the actual limitations of the proposed methods: a) GAN as a model reconstruction technique and b) GAN in a seismic inversion workflow some further studies are proposed:

- Develop deeper neural network able to reproduce the channels continuity without endpoints;
- Increase the generalization capability of the network as an attempt to increase the possibility of 100% match for the conditioning data;
- Develop more the method of GAN in the seismic inversion workflow looking for alternatives to create a more reliable model of conditioning data;
- Apply the GAN seismic inversion in a more complex case study;
- Develop a GAN seismic inversion method able to work within regions of low correlations (local perturbation instead of global perturbation).

This page was intentionally left blank.

6 References

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2015). *Learning Actvation Functions*. (2013), 1–9.

Arjovsky, M., & Bottou, L. (2017). *Towards Principled Methods for Training Generative Adversarial Networks*. 1–17. https://doi.org/10.1128/JVI.06350-11

Arjovsky, M., Chintala, S., & Bottou, L. (2017). *Wasserstein GAN*. https://doi.org/10.2507/daaam.scibook.2010.27

Azevedo, L., & Soares, A. (2017). Integration of Geophysical Data for Reservoir Modeling and Characterization. In *Geostatistical Methods for Reservoir Geophysics* (pp. 51–89). https://doi.org/10.1007/978-3-319-53201-1

Bengio, Y., Glorot, X., & Bordes, A. (2011). *Deep Sparse Rectifier Neural Networks*. *15*, 315–323.

Caetano, H. (2012). Integration of Seismic Information in Reservoir Models: Global Stochastic Inversion. *New Technologies in the Oil and Gas Industry*, 119–150. https://doi.org/10.5772/52308

Caterini, A., & Chang, D. E. (2016a). *A Geometric Framework for Convolutional Neural Networks*. Retrieved from http://arxiv.org/abs/1608.04374

Caterini, A., & Chang, D. E. (2016b). *A Novel Representation of Neural Networks*. Retrieved from http://arxiv.org/abs/1610.01549

Chan, S., & Elsheikh, A. H. (2017). *Parametrization and Generation of Geological Models with Generative Adversarial Networks*. Retrieved from http://arxiv.org/abs/1708.01810

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A. A., & Ieee, M. (2017). *Generative Adversarial Networks : An Overview*. (April), 1–14.

Goodfellow, I. (2016a). Generative Adversarial Networks (GANs). *NIPS Tutorial*, 1–86.

Goodfellow, I. (2016b). *NIPS 2016 Tutorial: Generative Adversarial Networks*. https://doi.org/10.1007/978-3-319-10590-1_53

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). *Generative Adversarial Networks*. *155*(4), 270–275. https://doi.org/10.1016/j.vetimm.2013.08.005

Haykin, S. (1999). *Neural networks: a comprehensive foundation* (Second). Prentice Hall.

Keskar, N. S., & Socher, R. (2017). *Improving Generalization Performance by Switching from Adam to SGD*. Retrieved from http://arxiv.org/abs/1712.07628

Kingma, D. P., & Ba, J. L. (2015). ADAM: A method for Stochastic Optimization. *Proceedings of ICLR 2015*. https://doi.org/10.1063/1.4902458

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86. https://doi.org/10.1214/aoms/1177729694

LeCun, Y., Bottou, L., Bengion, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*, 2278–2324. https://doi.org/10.1109/5.726791

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, *37*(1), 145–151. https://doi.org/10.1109/18.61115

Luke, S. (2013). *Essentials of metaheuristics. Lulu.*

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *ICML '13*, *28*, 6. Retrieved from http://www.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf

Mijwel, M. (2018). *Artificial Neural Networks Advantages and Disadvantages*.

Mosser, L., Dubrule, O., & Blunt, M. J. (2018a). *Conditioning of three-dimensional generative adversarial networks for pore and reservoir-scale models*. 1–5. Retrieved from http://arxiv.org/abs/1802.05622

Mosser, L., Dubrule, O., & Blunt, M. J. (2018b). *Stochastic SeismicWaveform Inversion using Generative Adversarial Networks as a Geological Prior*.

Müller, A. (1997). Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, *29*(2), 429–443. https://doi.org/10.2307/1428011

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *ICML*.

Nunes, R., Soares, A., Azevedo, L., & Pereira, P. (2017). Geostatistical Seismic Inversion with Direct Sequential Simulation and Co-simulation with Multi-local Distribution Functions. *Mathematical Geosciences*, *49*(5), 583–601. https://doi.org/10.1007/s11004-016-9651-0

Paneiro, G., Durão, F. O., Costa e Silva, M., & Falcão Neves, P. (2018). Artificial neural network model for ground vibration amplitudes prediction due to light railway traffic in urban areas. *Neural Computing and Applications*, *29*(11), 1045–1057. https://doi.org/10.1007/s00521-016-2625-9

Patro, S. G. K., & Sahu, K. K. (2015). *Normalization: A Preprocessing Stage*. Retrieved from http://arxiv.org/abs/1503.06462

Radford, A., Metz, L., & Chintala, S. (2015). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 1–16. https://doi.org/10.1051/0004-6361/201527329

Russell, B. H. (1988). 2. Part 2 - The Convolutional Model. In *Introduction to Seismic Inversion Methods* (pp. 2-1-2–19). https://doi.org/10.1190/1.9781560802303.ch2

Soares, Amílcar. (2001). Direct Sequential Simulation and Cosimulation. *Mathematical Geology*, *33*(8), 911–926.

Soares, Amilcar, Guerreiro, L., Caetano, H., Maciel, C., Real, A., Silva, F., … Al Shemsi, M. (2007). *Global Seismic Inversion - A New Approach to Integrate Seismic Informationin the Stochastic Models*. 1–4. https://doi.org/10.2523/111305-ms

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). *Striving for Simplicity: The All Convolutional Net*. 1–14.

Strebelle, S. (2002). Conditional Simulation of Complex Geological Structures Using Multiple-Point Statistics. *Mathematical Geology*, *34*(1), 1–21. https://doi.org/1014009426274

Tahmasebi, P. (2018). Multiple Point Statistics: A Review. In B. S. Daya Sagar, Q. Cheng, & F. Agterberg (Eds.), *Handbook of Mathematical Geosciences: Fifty Years of IAMG* (pp. 613–643). https://doi.org/10.1007/978-3-319-78999-6

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. https://doi.org/10.1137/1.9780898717921

Tieleman, T., & Hinton, G. (2012). *Lecture 6e - rmsprop: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning.

Villani, C. (2009). *Optimal Transport*. https://doi.org/10.1007/978-3-540-71050-9

Widrow, B., & Hoff, M. E. (1960). Adaptive Switching Circuits. *1960 IRE WESCON*, 96–104. New York: IRE.

Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., & Do, M. N. (2017). Semantic image inpainting with deep generative models. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 6882–6890. https://doi.org/10.1109/CVPR.2017.728

Yilmaz, Ö. (2001). *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data*.

https://doi.org/10.1190/1.9781560801580