**Paula Yamada Bürkle**

# Deep Physics-Driven Stochastic Seismic Inversion

**Tese de Doutorado**

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica, do Departamento de Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica.

Advisor : Profª Marley Maria Bernardes Rebuzzi Vellasco
Co-advisor: Prof. Leonardo Azevedo

Rio de Janeiro
September 2022

**Paula Yamada Bürkle**

# Deep Physics-Driven Stochastic Seismic Inversion

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica. Approved by the Examination Committee:

Rio de Janeiro, September 26th, 2022

**Paula Yamada Bürkle**

Graduated in Computer Science at the Universidade Federal Fluminense in 2003. Obtained her M.Sc. Degree in Computer Science from Universidade Federal Fluminense in 2006. Since 2006 has been working as a geoscience software developer for Petrobras.

Bibliographic data

Bürkle, Paula Yamada

Deep Physics-Driven Stochastic Seismic Inversion / Paula Yamada Bürkle; advisor: Marley Maria Bernardes Rebuzzi Vellasco; co-advisor: Leonardo Azevedo. – 2022.

112 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2022.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Aprendizado Profundo. 3. Inversão Sísmica. 4. Inversão Geoestatística. 5. Caracterização de Subsuperfície. I. Vellasco, Marley Maria Bernardes Rebuzzi. II. Azevedo, Leonardo. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

# Acknowledgments

First and foremost I would like to thank my advisors, Marley Vellasco and Leonardo Azevedo, for their time, effort, and guidance during this Ph.D. study.

Additionally, I wish to extend my appreciation to the examination committee members for all their insightful feedback and suggestions. I take this opportunity to express gratitude to all of the Department of Electrical Engineering faculty and staff members for their help and support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Many thanks to all of my friends and colleagues at TGEO. In particular, I would like to express deep feelings of gratitude to Luiz Eduardo Varella for his generous support and invaluable friendship at different stages of this research. I would also like to give special thanks to Andre Korenchendler for being there for me whenever I needed his help. I would like to extend special thanks to Anderson Pimentel, Erick Talarico, Matheus Sobreira, and Henrique Fraquelli for the insightful conversations we have had along the way.

I would like to thank Petrobras for the financial support that allowed me to conduct this research. Additionally, I would like to offer my special thanks to Silmara Campos, Luiz Alberto Barbosa de Lima, and Otaviano Pessoa for providing me the opportunity to engage in this Ph.D.. My appreciation also extends to Álvaro Arouca and Jonilton Pessoa for all their support during the last stage of this research.

I am deeply grateful to Braulio Silva for his moral support and assistance since the beginning of this journey.

Last but not least, I would like to express my gratitude to my family and friends who have been there for me emotionally as I worked toward my Ph.D.. This would not have been possible without their encouragement and support, specially the tremendous understanding of my husband and my son in the past few years.

# Abstract

Bürkle, Paula Yamada; Vellasco, Marley Maria Bernardes Rebuzzi (Advisor); Azevedo, Leonardo (Co-Advisor). **Deep Physics-Driven Stochastic Seismic Inversion**. Rio de Janeiro, 2022. 112p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Seismic inversion allows the prediction of subsurface properties from seismic reflection data and is a key step in reservoir modeling and characterization. Traditional seismic inversion methods usually come with a high computational cost or suffer from issues concerning the non-linearity and the strong non-uniqueness of the seismic inversion model. With the generalization of machine learning in geophysics, deep learning methods have been proposed as efficient seismic inversion methods. However, most of them lack a probabilistic approach to deal with the uncertainties inherent in the seismic inversion problems and/or rely on complete and representative training data, which are often scarcely available. To overcome these limitations, we introduce a novel seismic inversion method that explores the ability of deep convolutional neural networks to extract meaningful and complex representations from spatially structured data, combined with geostatistical stochastic simulation, to efficiently invert seismic reflection data directly for high-resolution subsurface models. Our method incorporates physics constraints, sparse direct measurements, while leveraging the use of imprecise but widely distributed indirect measurements as represented by the seismic data. The geostatistical realizations provide additional information with higher spatial resolution than the original seismic data. When used as input to our inversion system, they allow the generation of multiple possible outcomes for the uncertain model. Our approach is self-supervised, as it does not depend on ground truth input-output pairs. In summary, the proposed method is able to: (1) provide uncertainty assessment of the predictions, (2) model the complex non-linear relationship between observed data and model, (3) extend the seismic bandwidth at both low and high ends of the frequency parameters spectrum, and (4) lessen the need for large, annotated training data. The proposed methodology is first described in the acoustic domain to invert acoustic impedance models from full-stack seismic data. Next, it is generalized for the elastic domain to invert P-wave velocity, S-wave velocity and density models from pre-stack seismic data. Finally, we show that the proposed methodology can be further extended to perform petrophysical seismic inversion in a simultaneous workflow. The method was tested on a synthetic case and successfully applied to a real three-dimensional case from a Brazilian reservoir. The inverted models are compared to those obtained from a full iterative geostatistical seismic inversion. The proposed methodology allows retrieving similar models but has the advantage of generating alternative solutions in greater numbers, providing a larger exploration of the model parameter space in less time than the iterative geostatistical seismic inversion.

## Keywords

Deep Learning; Seismic Inversion; Geostatistical Inversion; Subsurface Characterization.

## Resumo

Bürkle, Paula Yamada; Vellasco, Marley Maria Bernardes Rebuzzi; Azevedo, Leonardo. **Inversão Sísmica Estocástica com Aprendizado Profundo Orientado à Física**. Rio de Janeiro, 2022. 112p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

A inversão sísmica é uma etapa essencial na modelagem e caracterização de reservatórios que permite a estimativa de propriedades da subsuperfície a partir dos dados de reflexão sísmica. Os métodos convencionais usualmente possuem um alto custo computacional ou apresentam problemas relativos à não-linearidade e à forte ambiguidade do modelo de inversão sísmica. Recentemente, com a generalização do aprendizado de máquina na geofísica, novos métodos de inversão sísmica surgiram baseados nas técnicas de aprendizado profundo. Entretanto, a aplicação prática desses métodos é limitada devido a ausência de uma abordagem probabilística capaz de lidar com as incertezas inerentes ao problema da inversão sísmica e/ou a necessidade de dados de treinamento completos e representativos. Para superar estas limitações, um novo método é proposto para inverter dados de reflexão sísmica diretamente para modelos da subsuperfície de alta resolução. O método proposto explora a capacidade das redes neurais convolucionais em extrair representações significativas e complexas de dados espacialmente estruturados, combinada à simulação estocástica geoestatística. Em abordagem auto-supervisionada, modelos físicos são incorporados no sistema de inversão com o objetivo de potencializar o uso de medições indiretas e imprecisas, mas amplamente distribuídas do método sísmico. As realizações geradas com simulação geoestatística fornecem informações adicionais com maior resolução espacial do que a originalmente encontrada nos dados sísmicos. Quando utilizadas como entrada do sistema de inversão, elas permitem a geração de múltiplos modelos alternativos da subsuperfície. Em resumo, o método proposto é capaz de: (1) quantificar as incertezas das previsões, (2) modelar a relação complexa e não-linear entre os dados observados e o modelo da subsuperfície, (3) estender a largura de banda sísmica nas extremidades baixa e alta do espectro de parâmetros de frequência, e (4) diminuir a necessidade de dados de treinamento anotados. A metodologia proposta é inicialmente descrita no domínio acústico para inverter modelos de impedância acústica a partir de dados sísmicos pós-empilhados. Em seguida, a metodologia é generalizada para o domínio elástico para inverter a partir de dados sísmicos pré-empilhados modelos de velocidade da onda P, de velocidade da onda S e de densidade. Em seguida, a metodologia proposta é estendida para a inversão sísmica petrofísica em um fluxo de trabalho simultâneo. O método foi validado em um caso sintético e aplicado com sucesso a um caso tridimensional de um reservatório brasileiro real. Os modelos invertidos são comparados àqueles obtidos a partir de uma inversão sísmica geoestatística iterativa. A metodologia proposta permite obter modelos similares, mas tem a vantagem de gerar soluções alternativas em maior número, permitindo explorar de forma mais efetiva o espaço de parâmetros do modelo quando comparada à inversão sísmica geoestatística iterativa.

## Palavras-chave

Aprendizado Profundo; Inversão Sísmica; Inversão Geoestatística; Caracterização de Subsuperfície.

# Table of contents

# 1
# Introduction

This chapter provides the context of this PhD thesis, the statement of the problem, the challenges, and the contributions of the research conducted.

## 1.1
## Context

Keeping up with the increasing amount of data coming from modern sensor technologies and advanced data acquisition systems is an ongoing challenge in the energy industry. Machine learning (ML) is one of the most promising technologies to deal with big data. Deep learning (DL) is among the fastest emerging technologies in ML. The rapid development in this area is supported by the availability of powerful hardware, large datasets and a wide range of ready-to-use ML solutions provided by open-source frameworks, such as PyTorch [1] and TensorFlow [2].

DL applies deep neural networks (DNNs), which consist in neural networks composed of a large number of hidden-layers that are stacked in a hierarchy of increasing complexity and abstraction. Convolutional Neural Networks (CNN) [3] are a specialized kind of DNNs that have considerably outperformed their predecessors in computer vision tasks [4, 5, 6, 7].

In the field of geophysics, CNNs have gained popularity due to the spatial and multidimensional nature of geophysical data. Examples of applications of CNNs in geophysics comprise identifying geological features from seismic attributes [8], seismic structure interpretation [9, 10, 11, 12], and seismic facies classification [13, 14].

Recently, DL has been applied to seismic reservoir characterization, overcoming the limitations of traditional methods [15]. In seismic reservoir characterization, one aims at predicting the spatial distribution of elastic and/or petrophysical rock properties from seismic reflection data [16, 17, 18, 19].

During the stages of exploration and production of hydrocarbon resources, reservoir models are required for the understanding of the geological context and its challenges. They are also critical inputs for improving and accelerating drilling decisions. Given that the average cost of drilling an offshore well is approximately US$1 million per day [20], optimal decision-making might result in saving hundreds of millions of dollars.

This process of building a reliable subsurface model involves several disciplines, the main ones being geology and geophysics [17, 18, 19]. Geological data differ from geophysical data in terms of resolution, coverage, and degrees of precision. The former contains sparse-coverage and fine-scale information, usually obtained from direct observation of rocks retrieved by drilling wells. Although more reliable, well data is expensive to acquire and spatially limited. On the other hand, by taking indirect measures from the subsurface, geophysical methods provide relatively low-cost and widely distributed information with large uncertainty.

Since large-scale exploratory drilling is expensive and impractical, seismic methods are used in reservoir characterization not only for identifying and mapping geological features that might contain hydrocarbons, but also for predicting the physical properties of the rocks within a reservoir [17, 18, 19, 21]. Integrating both geological and geophysical sources of information is important to preserve small-scale heterogeneity from well data and assimilate large-scale features from seismic data. This step of the geo-modeling workflow involves solving a challenging inverse problem - the seismic inversion.

## 1.2 Problem Definition

Seismic inversion is the process of reconstructing the subsurface model [1] that created the recorded seismic data. In this process, the original interface property, as represented by the seismic reflectivity data, is converted into an elastic or petrophysical rock property.

Seismic inversion methods ideally compute a complete three-dimensional (3D) subsurface model of a certain target area. The subsurface model is typically defined as a finite-dimensional space, usually referred to as the model parameters' space, where the continuous parameters in the Earth field pass to a discrete set of quantities to approximate the continuous space [22].

Most seismic-inversion methods involve forward seismic modeling to devise synthetic data by simulating seismic field experiments from a predicted Earth-model. In this process, the model parameters' space is mapped into the data space by a seismic forward operator, which may be a function with an analytic form, a matrix operator, or any computational algorithm.

The seismic inverse problem is commonly expressed as the resolution to the following mathematical problem [22]:

$$\mathbf{m} = f^{-1}(\mathbf{d}) + \eta, \tag{1-1}$$

where $\mathbf{m}$ is the set of the model parameters that represents the spatial distribution of the subsurface property, $\mathbf{d}$ is the set of recorded seismic data, $f$ is the seismic forward operator, and $\eta$ stands for the random noise.

The random noise is associated with inaccuracies of the seismic measurement method, such as the resolution and band-limited nature of the seismic data, measurement errors, and modeling uncertainties. Because of it, finding the solution for equation 1-1 is an ill-posed problem, which means that many combinations of the model parameters fit the observed data equally well.

## 1.3 Challenges

Seismic inversion methods can be broadly divided into deterministic and stochastic inversion [18]. Deterministic seismic inversion methods predict a single best-fit inverted model with limited capability to assess uncertainty about the model predictions. On the other hand, stochastic inversion methods

[1] In this thesis, we use the terms Earth model, subsurface model, and reservoir model interchangeably to mean a 3D numerical representation of subsurface properties that is obtained by seismic inversion workflows.

allow for assessing the uncertainty about the predictions by sampling from the model parameters space.

Different stochastic inversion methods exist depending on assumptions about the distribution of the model parameters and how the measurement errors are considered. Iterative geostatistical seismic inversion methods [19, 23, 24, 25] are examples of this family of methods which use stochastic sequential simulation and co-simulation as the model perturbation and update techniques [26]. While they allow predicting multiple subsurface models that fit equally well with the observed geophysical data, they are computationally expensive, mainly due to the model generation and update steps [27].

Solutions based on DL for the seismic inversion problem most benefit from the capacity of DNNs in mapping complex and nonlinear relationships between data and model domains. Since DL techniques have shown to perform well in problems involving massive amounts of labeled data, early DL-based approaches to seismic inversion problems are fully data-driven, thus relying entirely on the completeness of the training data.

However, because of the high cost of acquiring direct measures of the subsurface, many geophysical problems have a huge amount of unlabeled data and a limited quantity of annotated data, which is usually obtained from real elastic and petrophysical data from borehole logging (i.e., well-log data). In this context, the lack of sufficient and representative samples of the target subsurface model is one of the biggest challenges in the generalization of DL in seismic inversion.

The scarcity of labeled data has been addressed using transfer learning, data augmentation techniques, and generating synthetic data with geostatistical modeling tools [28, 29, 30, 31, 32]. Such strategies aim at reducing the dependency on large training data and mitigating overfitting issues.

Concurrently, deterministic physical models can be applied to guide the training of a machine learning model, a paradigm known as theory-guided or physics-driven data science [33]. When physics-driven learning is applied to inverse problems, the known physics laws that explain the system under investigation (i.e., the propagation of a seismic wavefield) are used to leverage unlabeled data by constraining the relationship between the subsurface model and observed seismic data [34, 35, 36, 34, 37, 38, 39, 38, 40].

An additional challenge, still largely open to research, is how to quantify uncertainty in neural networks' predictions [41, 42]. This is a crucial concern in the predictions of subsurface models, where direct observations are rare and indirect investigation methods are inherently uncertain [40]. Monte Carlo dropout method [43] has been proposed to capture the network uncertainty about the subsurface model predictions [34]. Alternatively, deep generative networks have been applied to learn the underlying set of subsurface models for uncertainty propagation [28, 44, 45, 46, 47, 48].

## 1.4 Contributions

We propose a physics-driven DL-based framework to invert multiple equally probable elastic and petrophysical subsurface models from seismic reflection data. Our approach mainly differs from others in that it is stochastic and fully self-supervised. Existing DL-based approaches are deterministic and/or rely, totally or partially, on labeled training data to train a machine learning model, whether by using an approximate or exact process to generate input-output pairs, or by adopting a semi-supervised learning strategy.

To compensate for the incompleteness of the seismic data, we borrow prior information about the spatial distribution of the subsurface properties from an ensemble of subsurface models generated with geostatistical simulation [26]. These models are conditioned locally to well data and to a prior belief on the spatial distribution pattern (i.e., a variogram model). The geostatistical simulation methods provide the high- and low-frequency information not captured in the observed seismic data.

In the proposed inversion framework, a deep neural network works as a surrogate to the inverse of the seismic forward model. The geostatistical models are fed into the neural network alongside the observed seismic data. The computational costs of our method come mostly from training, which happens only once up front. After training, the inference processing time is virtually zero, thus making the overall computing costs a fraction of that needed for iterative methods. As a result, the proposed inversion framework allows exploring the model parameters' space with low computational effort.

The physics-driven learning is adopted to overcome the lack of sufficiently large annotated training data. As with other physics-driven approaches, the seismic forward model is incorporated into the graph of the neural network to provide the physics-based updates to the network parameters. Additionally, we propose a physics-based adaptive loss function to automatically control whether the network assimilates more information from the seismic data or from the geostatistical models.

Each geostatistical model that is used as input to the trained network will result in an alternate solution to the inverted model. The final set of inverted subsurface models can be further used to evaluate the uncertainty of the network predictions. This is an alternative probabilistic perspective to the previous studies.

We summarize the main contributions of this work as follows:

1. **Self-supervised learning** - We adopt the physics-driven learning to constrain the model-data relationship and leverage the use of unlabeled seismic data. Our training procedure does not depend on ground truth models. Instead, it relies on the diversity of unconditional (with respect to the seismic data) geostatistical models to explore the uncertainty space of the model parameters.

2. **Uncertainty quantification** - Alternatively to Monte Carlo dropout approaches, the predictions are given based on one single deterministic neural network, but previously simulated models are used to input stochasticity into the network in order to devise multiple estimates. From the set of estimates, one can extract the density probability function for each location and spatially quantify uncertainty.

3. **Efficiency** - Unlike conventional iterative geostatistical seismic inversion methods, the computational complexity of the proposed method does

not scale with the number of realizations. The training procedure is performed on a subset of the set of geostatistical simulated models for which the relative computational cost decreases with the number of final estimates. The network serves as a surrogate for the whole iterative optimization process. Once trained, the network can efficiently generate thousands of inverted models to hopefully devise more well-calibrated uncertainty quantification.

This leads to a novel physics-driven and self-supervised DL-based method for seismic inversion problems, accounting for uncertainty quantification. An iterative geostatistical seismic inversion method will be the benchmark for the proposed inversion framework.

## 1.5 Organization

The rest of this work is organized as follows:

In Chapter 2 we provide an overview of seismic modeling and acquisition methods, and review non-DL-based seismic inversion methods that are well-established in the energy industry.

Chapter 3 reviews key concepts of deep learning and presents related tasks to the seismic inversion problem in the field of computer vision. We close Chapter 3 with a review of the existing literature on DL-based methods applied to seismic inversion problems.

Chapter 4 presents the proposed seismic inversion framework. We start with a general overview, followed by a detailed description of the parameterization and architecture of the neural network used to invert acoustic impedance models from full-stack seismic data. Then, the architecture of the seismic forward modeling network is described. We follow by generalizing our methodology for the elastic domain to simultaneously invert P-wave velocity, S-wave velocity, and density models from pre- or partially stacked seismic data. Finally, we close Chapter 4 by showing that the proposed methodology can be further extended to directly invert seismic reflection data for petrophysical properties.

Chapter 5 comprises the application examples. We first illustrate and validate the proposed methodology using an one-dimensional synthetic example. Next, the effectiveness of our approach is demonstrated on a three-dimensional real case from a Brazilian post-salt area. We further compare our results in the real case with those obtained by the iterative geostatistical inversion according to quantitative and qualitative performance metrics, as well as relative to the computational cost.

Finally, concluding remarks and directions for future research are presented in Chapter 6.

Appendix A provides supplementary information on geostatistics, while complementary results for the real application case can be found in Appendix B.

# 2
# Background

This chapter covers the background of this research and reviews related literature used in this study. We start by presenting the theoretical basis of the seismic refraction method, a geophysical technique for subsurface investigation that forms the basis of the research problem. An overview of the seismic inversion solutions that are consolidated in the energy industry is presented in the last section of this chapter.

## 2.1
## Seismic Data Modeling and Acquisition

The seismic method consists in sending acoustic waves into the Earth and then recording the strength of the reflected waves and the time it takes for each wave to reflect back. By studying the propagation of seismic waves into the rock layers beneath the subsurface, seismic methods provide detailed images of structures, stratigraphy, and rock physical properties.

Seismic surveys can be conducted onshore and offshore, the latter being the most common in petroleum exploration. Figure 2.1 shows a graphical depiction of how an offshore seismic survey works. In offshore surveys, a survey ship tows an array of sensors (i.e., receptors), referred to as hydrophones, located just beneath the water surface. Compressed air guns (i.e., the source) towed behind the ship periodically release a high-pressure pulse of air underwater. The distance between the source and the receptor is referred to as the offset. The sound waves propagate through the water column and into the subsurface until they reach a layer with different elastic properties from which they may be reflected. The interface between two layers with contrasting elastic properties is referred to as a seismic reflector. After passing through the first layers' discontinuity, the transmitted waves continue traveling down to a second or third prominent change in subsurface rock properties.

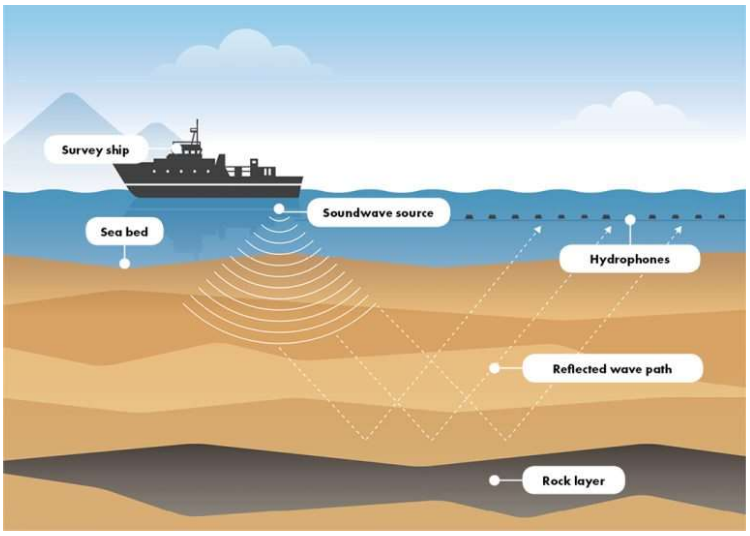


Figure 2.1: Graphical depiction of an offshore seismic survey. Source: [49]

Further processing is conducted to transform recorded seismic reflection data into various forms of images of reflecting boundaries in the subsurface. Figure 2.2 shows an example of an offshore seismic reflection profile. During seismic processing, pre-stack seismic data comprising traces from different shot records with a common reflection point are stacked to form a single trace (Figure 2.3). The resulting product of this process is usually referred to as post-stack (or full-stack) seismic data, while the original recorded seismic reflection data is referred to as pre-stack seismic data.

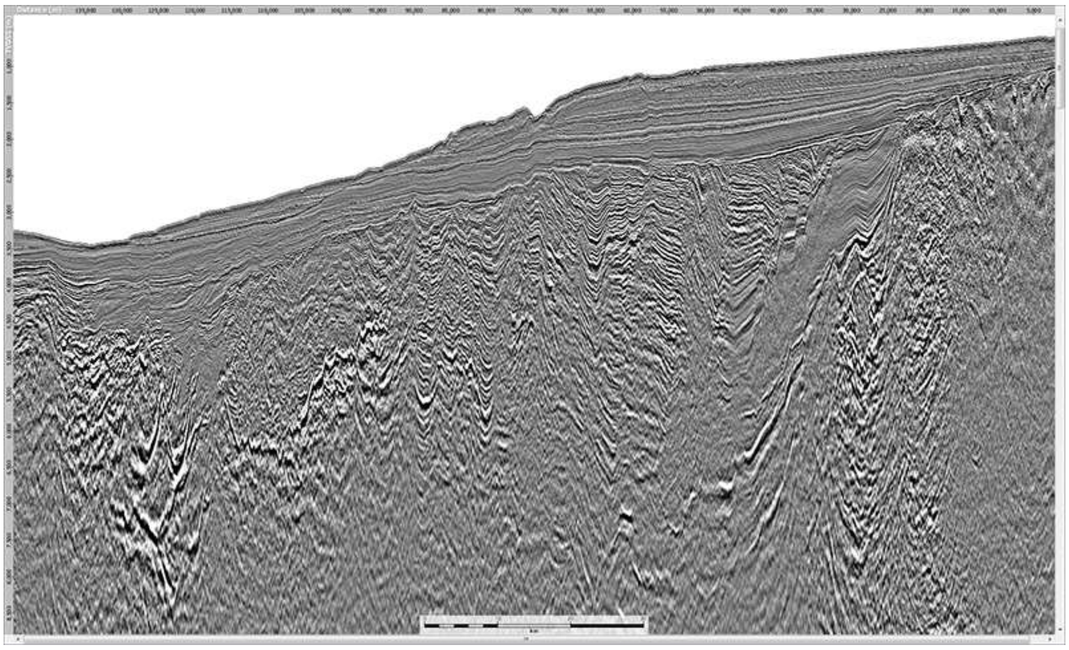


Figure 2.2: Example of an offshore seismic reflection profile with 10 x vertical exaggeration. Source: [50]

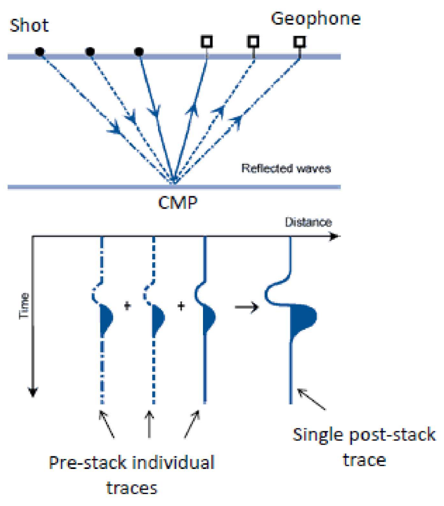


Figure 2.3: Simple illustration of how multiple traces with a common mid point (CMP) combine to form a single stacked trace.

As the seismic waves propagate, they cause an elastic deformation of the medium by applying a longitudinal stress (compression or dilation) and shear stress (parallel to the surface). The medium returns to its original state when these forces are removed. Essentially there are two types of seismic waves captured by seismic recordings that geophysicists are interested in: compressional waves (primary or P-waves) and shear waves (secondary or S-waves) (Figure 2.4). P-waves are pressure waves that can move through solid rock and fluids. They cause vibrations that are parallel to the direction of wave propagation. S-waves are shear waves that can only move through solid rock. They are slower than P-waves and cause vibrations that are orthogonal to the direction of wave propagation.

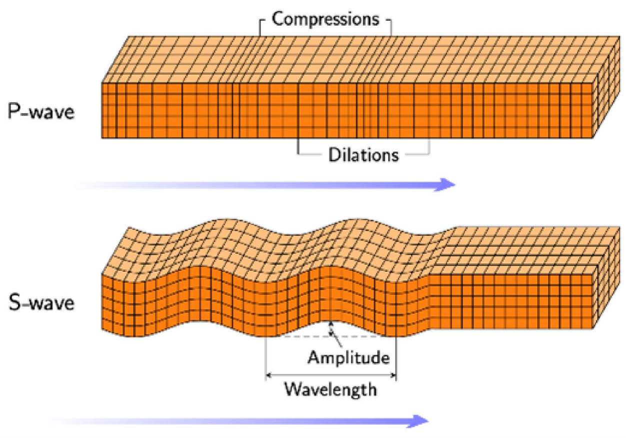


Figure 2.4: Two types of seismic waves geophysicists are interested in: P-Waves and S-Waves.

The hydrophones receive and record the reflected seismic waves at different times. Each recorded time corresponds to the time taken for a seismic wave to travel from the source down to a reflector and back to it, which is referred to as two-way travel time (TWT). The time depends on the velocity at which the seismic waves travel through the subsurface. The velocity of the P- and S-waves, $V_P$ and $V_S$, respectively, are defined by the elastic properties of the rocks that compose the medium:

$$V_P = \sqrt{\frac{\lambda + 2\mu}{\rho}}, \tag{2-1}$$

$$V_S = \sqrt{\frac{\mu}{\rho}}, \tag{2-2}$$

where $\rho$ is the density and $\mu$ and $\lambda$ are Lamé constants, which describe the stress-strain relation within a solid medium.

The acoustic impedance (AI[1]) or P-wave impedance of a rock layer is equal to the product of density ($\rho$) and the P-wave velocity:

$$\mathrm{AI} = \rho \cdot V_P \tag{2-3}$$

Similarly, the shear impedance (SI) or S-wave impedance is the product of bulk density and S-wave velocity:

$$\mathrm{SI} = \rho \cdot V_S \tag{2-4}$$

While seismic reflection data are related to relative changes in reservoir properties, elastic properties, such as acoustic and shear impedance, are directly related to rock properties of interest, such as fluid saturation, porosity, and lithology [17]. For example, if porosity increases the acoustic impedance generally decreases.

The poisson's ratio ($\sigma$) is another important elastic parameter as it allows identifying fluid type in the pores of a reservoir. It is derived from velocity and density, following:

$$\sigma = \frac{V_P^2 - 2V_S^2}{2(V_P^2 - V_S^2)} \tag{2-5}$$

Post-stack seismic data contains information about P-impedance, while pre-stack seismic data contains additional information about the rock physical properties such as P-impedance, S-impedance and density [17].

When the P-wave ray is normally incident on the layers' interface, only other P-waves are reflected and transmitted. For non-normally incident waves, a pair of reflected and transmitted S-waves is also generated (Figure 2.5).

[1]The abbreviation used to describe acoustic impedance is AI. This abbreviation is familiar in the geophysical context, but it can be mistaken for artificial intelligence by the general reader.

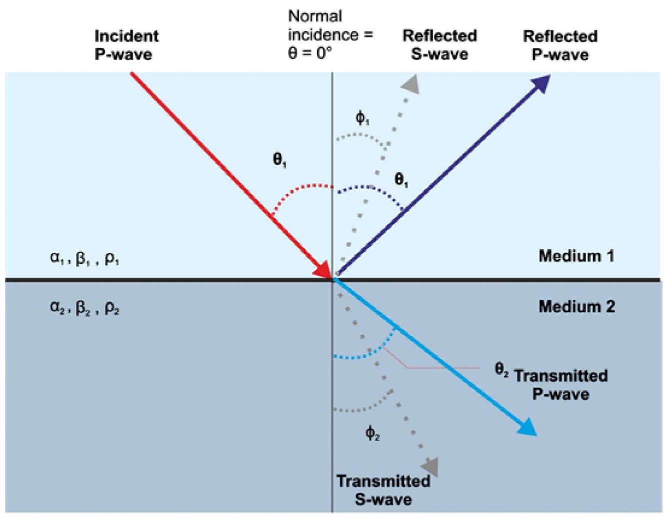


Figure 2.5: Reflection and transmission of an incident P-wave propagating through an interface between two layers with different elastic properties. Source: [51].

The Zoeppritz equations [52] are a set of equations that fully describe the relationship between angles, reflected and transmitted coefficients of compressional and shear waves at an interface:

$$\begin{pmatrix} \cos\theta_1 & \cos\theta_2 & -\sin\phi_1 & \sin\phi_2 \\ \sin\theta_1 & -\sin\theta_2 & \cos\phi_1 & \cos\phi_2 \\ \cos 2\phi_1 & -\frac{AI_2}{AI_1}\cos 2\theta_2 & -\frac{V_{S1}}{V_{P1}}\sin 2\phi_1 & -\frac{SI_2}{AI_1}\sin 2\phi_2 \\ \sin 2\theta_1 & \frac{\rho_2 V_{S2}^2 V_{P1}}{\rho_1 V_{S1}^2 V_{P2}}\sin 2\theta_2 & \frac{V_{P1}}{V_{S1}}\cos 2\phi_1 & -\frac{\rho_2 V_{S2} V_{P1}}{\rho_1 V_{S1}^2}\cos 2\phi_2 \end{pmatrix} \begin{pmatrix} R_{PP} \\ T_{PP} \\ R_{PS} \\ T_{PS} \end{pmatrix} = \begin{pmatrix} \cos\theta_1 \\ -\sin\theta_1 \\ -\cos 2\phi_1 \\ -\sin 2\theta_1 \end{pmatrix} \quad (2\text{-}6)$$

where $AI_1, V_{P2}, V_{S2}$ and $AI_1, V_{P2}, V_{S2}$ represent the acoustic impedance and the P- and S- waves velocities of first and second medium, respectively (see Figure 2.5). $R_{PP}$ and $R_{PS}$ denote reflected coefficients of compressional and shear waves, respectively, while $T_{PP}$ and $T_{PS}$ denote transmitted coefficients of compressional and shear waves, respectively.

These equations give the exact theoretical amplitude of an incident P-wave and the amplitude of reflected and refracted P- and S-waves as a function of the angle of incidence. For a normally incident wave, the reflection coefficients ($R_{PP}$) are given by the ratio of the amplitudes of the reflected and incident waves, defined as the difference of acoustic impedance (AI) over the sum:

$$R_{P0} = \frac{\mathrm{AI}_2 - \mathrm{AI}_1}{\mathrm{AI}_2 + \mathrm{AI}_1} \quad (2\text{-}7)$$

Although the Zoeppritz equations form a system of four equations that can be solved for the four unknowns, their application to seismic data has proven to be difficult [53, 51]. Approximations of the Zoeppritz equations provide simpler models of how the reflection amplitudes vary with the rock properties involved (e.g., density and velocity). One of the most successful approximations to the Zoeppritz equations is the 3-term Shuey's [53], which assumes Poisson's ratio to be the elastic property most directly related to the angular dependence of the reflection coefficient:

$$R_{PP}(\theta) \approx R_{P0} + G\sin^2\theta + \frac{1}{2}F(\tan^2\theta - \sin^2\theta), \tag{2-8}$$

$$R_{P0} = \frac{1}{2}\Big(\frac{\Delta V_P}{V_P} + \frac{\Delta\rho}{\rho}\Big), \tag{2-9}$$

$$G = \frac{1}{2}\frac{\Delta V_P}{V_P} - 2\frac{V_S^2}{V_P^2}\Big(\frac{\Delta\rho}{\rho} + 2\frac{\Delta V_S}{V_S}\Big), \tag{2-10}$$

$$F = \frac{1}{2}\frac{\Delta V_P}{V_P}, \tag{2-11}$$

$$\Delta V_\rho = \rho_2 - \rho_1, \tag{2-12}$$

$$\Delta V_S = V_{S2} - V_{S1}, \tag{2-13}$$

$$\Delta V_P = V_{P2} - V_{P1}, \tag{2-14}$$

$$\tag{2-15}$$

where $\theta$ is the average of the incident and transmitted P-wave angles.

The first term of equation 2-8 ($R_{P0}$) gives the reflection coefficient for a normally incident wave, while the second term describes $R_{PP}(\theta)$ at intermediate angles, and the last term explains $R_{PP}(\theta)$ to the critical angle (i.e., at large angles/far offsets).

In order to compare the field observation data with modeled data, one has to perform seismic forward modeling. The main seismic parameters used to generate a synthetic seismic trace are density and P- and S-wave velocities, and the source pulse (i.e., wavelet). In this process, at first, synthetic seismic traces are computed based on these seismic parameters by using the Zoeppritz's equations or one of their approximations. A conversion to time is necessary as the reflection coefficients are a function of depth. Then the seismic traces are formed by a process called "convolution" of a series of reflection coefficients with the predefined wavelet (Figure 2.6).

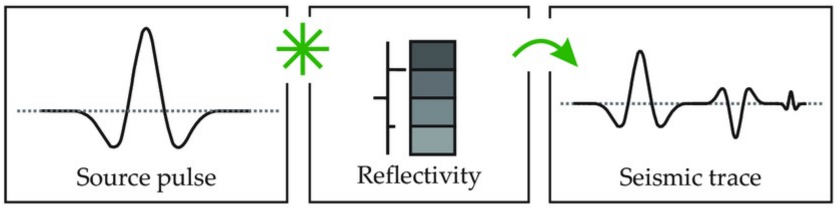


Figure 2.6: Convolutional model in the formation of a seismic trace from a given source pulse and reflection coefficients. Source: [54].

The synthetic seismic trace, $S(t)$, is obtained through the integration of all of the reflection events:

$$S(t) = w(t) * R(t), \tag{2-16}$$

where $w(t)$ is the assumed source pulse function, and $R(t)$ denotes the reflectivity function.

## 2.2 Seismic Inversion Methods

The inversion of seismic data is challenging due to its under-determined nature and the non-linearity of the forward model. The existing approaches can be broadly divided into two main types of problems: deterministic and probabilistic (or stochastic).

Traditional seismic inversion methods, deterministic and stochastic, minimize a cost function that consists of a data misfit term, which measures the residual between the theoretical predictions of the forward problem and the observed seismic [22]. The model-based seismic inversion approaches apply successive updates to an initial guess of the subsurface model to minimize the data misfit. This inverse problem can be deterministically solved by estimating the parameters of the model that give the minimum data misfit:

$$\arg\min_{\hat{\mathbf{m}}} \|f(\hat{\mathbf{m}}) - \mathbf{d}\|_2^2, \tag{2-17}$$

where $\hat{\mathbf{m}}$ represents the parameters of the model to be estimated, $\|.\|_2^2$ denotes the squared L2-norm, $\mathbf{d}$ is the set of recorded seismic data, and $f$ is the seismic forward model.

Given an input, deterministic methods will always produce the same and a single output that represents the best-fit subsurface model or the most likely to occur. These algorithms are usually based on full-stack seismic data and rely on simplifications and approximations to the subsurface, for example, assuming linearized reflection coefficients [55, 22, 56, 57]. Global optimization techniques, such as simulated annealing [58, 59] and genetic algorithms [60], can be used to solve nonlinear inversion problems in a deterministic form.

Full-waveform inversion (FWI) [22, 61, 62] is a popular seismic inversion method based on the wave equation, which reflects the physics nature of the seismic wavefield propagation. As with any model-based approaches, FWI starts from an initial model and solves for seismic inversion by minimizing an objective function that represents the residuals between predicted and observed seismic traces. In spite of the potential of FWI to provide more accurate high-resolution reservoir models from seismic data, the method is highly computationally demanding with no guarantees of global convergence.

Deterministic methods suffer from the strong non-uniqueness of the seismic inverse problem. However, predicting multiple equally probable subsurface models and assessing uncertainties is of critical importance to mitigating risks and optimal decision-making [18]. As a means to deal with the non-uniqueness of inverse problems, stochastic methods attempt to find all of the acceptable solutions or express the solution as a probability distribution function over the model parameters space.

The Bayesian linear inversion provides a probabilistic framework for quantifying the uncertainty by posing the seismic inversion problem as a Bayesian inference problem [57, 56, 22, 63]. These methods consider a discrete linear inverse problem with Gaussian distributed prior and data errors to analytically devise a complete posterior distribution that is also Gaussian. This posterior distribution can be further used to estimate the most likely solution or any population parameter of interest (e.g., mean and variance). In cases where the solution to the posterior distribution cannot be analytically calculated, the posterior probability density can be assessed by using a Markov Chain Monte Carlo (MCMC) sampling method [22]. In some cases, a Gaussian-Mixture model framework can be applied to overcome the limitations imposed by the Gaussian assumption [63].

The linearization of the forward model, the low vertical resolution of the inverted models, and the Gaussian assumption between all parameters and the observed data are limiting factors that require consideration in the practical application of the Bayesian linear inversion methods. Iterative procedures based on stochastic sequential simulation are able to overcome these limitations imposed by the Bayesian framework [19]. We next present an example of this class of stochastic sequential simulation methods, which is used to benchmark this research.

### 2.2.1
### Iterative Geostatistical Seismic Inversion

Geostatistical seismic inversion methodologies are iterative optimization methods where the perturbation of the model parameter space is performed with stochastic sequential simulation and co-simulation [19]. These seismic inversion methods use geostatistical simulation tools as they provide a framework to represent reservoir heterogeneity and to model complex relationships between reservoir properties [17].

Soares et al. [64] introduced the global stochastic inversion methodology, an iterative approach that uses the principles of crossover from genetic algorithms as the global optimization technique. The global geostatistical acoustic inversion (GSI) [64, 65] allows the inversion of post-stack seismic reflection data for acoustic impedance (AI) models.

The general outline of this family of geostatistical inversion algorithms is synthesized in Figure 2.7. In GSI, we start by generating a set of $M$ acoustic impedance models from the set of well-log data and imposing a spatial continuity model (i.e., a variogram model) with direct sequential simulation (DSS) [66].

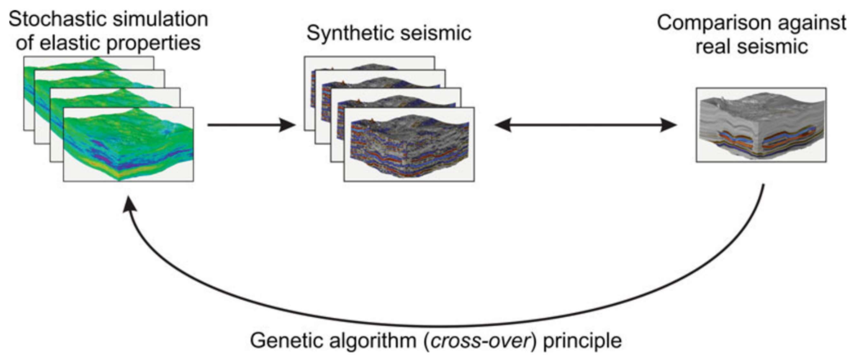

Stochastic simulation of elastic properties
Synthetic seismic
Comparison against real seismic
Genetic algorithm (*cross-over*) principle

Figure 2.7: General outline of the global geostatistical seismic inversion (GSI) algorithms. Source: [19].

In DSS, the simulation grid (or inversion grid) is visited along a random path that visits all the grid cells. At each location, the kriging estimate, and kriging variance are computed based on direct observations and previously simulated grid cells. The kriging estimate and variance are then used to draw a value from an auxiliary probability distribution function built from the global distribution function of AI as estimated from the well-log data. The simulated value is then assigned to the inversion grid and considered as conditioning data for the next simulation location along the random path. As the random path changes each time the simulation runs, the conditioning data at each location along the simulation random path changes, and therefore results in alternative models (i.e., geostatistical realizations).

From the set of $M$ AI realizations we compute the normal incidence reflection coefficients, which are then convolved with a wavelet to generate $M$ synthetic full-stack seismic volumes. Each synthetic seismic trace is compared with the corresponding observed seismic trace following:

$$S^{j,r} = \frac{2\sum_{k=-m}^{m}(\mathbf{d}_{t+k} \cdot \mathbf{d}_{t+k}^{j,r})}{\sum_{k=-m}^{m}(\mathbf{d}_{t+k})^2 + \sum_{k=-m}^{m}(\mathbf{d}_{t+k}^{j,r})^2}, \tag{2-18}$$

where $S$ is the local trace-by-trace similarity coefficient, $j$ is the iteration number, $r$ is the realization number, $m$ is the number of samples of a moving window, and $\mathbf{d}_t$ and $\mathbf{d}_t^{j,r}$ are the observed and synthetic seismic traces at sample $t$.

$S$ works as Pearson's correlation coefficient (ranges from -1 to 1) but is sensitive to both the waveform and the amplitude content of the seismic traces. However, due to restrictions related to geostatistical co-simulation used in the subsequent iteration, negative values of S are truncated at zero.

The AI traces that generate the highest S are selected and stored in auxiliary volumes along with the corresponding S and used in the co-simulation of a new set of AI models in the next iteration. The iterative inversion finishes when the global S, computed between synthetic and recorded full-stack volumes, is above a given threshold or a pre-defined number of iterations is reached. The GSI can be summarized with the following algorithm [64, 19]:

**Algorithm 1:** Global Geostatistical Acoustic Inversion (GSI)

```
Input: Variogram model, AI well-log data, post-stack seismic
       reflection data
Output: Set of estimated AI volumes
1 Generate a set of $M$ realizations of AI from the existing AI well-log
    data and imposing a variogram model that describes the expected
    spatial continuity pattern of AI in the subsurface.
2 Forward model each $M$ AI model and compute $M$ synthetic seismic
    volumes.
3 Compare, on a trace-by-trace basis, the $M$ synthetic and real seismic
    volumes following equation 2-18.
4 For each location within the inversion grid, select the simulated AI
    traces that ensure the maximum $S$. Store the selected AI traces and
    the collocated $S$ values in two auxiliary volumes.
5 Use the stored auxiliary volumes (step 4) as secondary variables in the
    co-simulation of a new set of $M$ AI models. Locations associated with
    a high $S$ will exhibit a similar spatial pattern as observed in the
    auxiliary volume, while locations collocated with a low $S$ will have
    little influence from the auxiliary volume.
6 Return to step 2 and iterate until the global $S$ between real and
    synthetic volumes is above a predefined threshold.
```

# 3
# Deep Learning

This chapter starts with an introductory overview of deep learning, including the feedforward networks, the techniques for optimization and regularization of these models, and convolutional neural networks - the machine learning models on which the proposed inversion framework is built. The concepts presented here were mainly adapted from [5], which the reader can refer to a broad range of topics in deep learning.

Then, we draw a parallel between the seismic inversion problem and image reconstruction tasks in computer vision, such as super-resolution and image denoising. We revisit key concepts regarding such tasks and describe practical building blocks and criteria for addressing them, including the fully convolutional neural networks and different strategies commonly applied to train these machine learning models.

Finally, we review the existing literature on the use of deep learning methods in geophysical inverse problems.

## 3.1
## Basic Concepts

The history of deep learning dates back to the 1950s, when Walter Pitts and Warren McCulloch, inspired by biological neural networks, created the first mathematical model of an artificial neuron [67]. The first neural network, the Perceptron [68], was originally developed by Frank Rosenblatt in 1957. Three decades later, Yann LeCunn et al. published a seminal paper [3] describing a “modern” neural network architecture for document recognition, a predecessor of current deep learning models.

### 3.1.1
### Feedforward Neural Networks

Feedforward neural networks are the most common type of deep learning models. They are used to approximate some function $f^*$, and can be expressed as a mapping:

$$\hat{\mathbf{y}} = f_\theta(\mathbf{x}), \tag{3-1}$$

where $\hat{\mathbf{y}}$ is the output (e.g., a category), $f$ is a function, $\mathbf{x}$ is the input, and $\theta$ are the network parameters that result in the best function approximation [5].

As the name suggest, the information flows in the forward direction through the function $f$, starting from the input $\mathbf{x}$ towards the output layer, where it finally produces an output $\hat{\mathbf{y}}$.

The function $f$ is a composition of functions in a chain. Each function typically computes an affine transformation controlled by parameters $\theta$, followed by a fixed nonlinear operation. For example, given an input $\mathbf{x}$, we can express a network composed of three functions as:

$$
\begin{aligned}
f_{\theta_1}^{(1)}(\mathbf{x}) &= \mathbf{h}_1 = g_1(\mathbf{x}^T\mathbf{w}_1 + \mathbf{b}_1),\\
f_{\theta_2}^{(2)}(\mathbf{h}_1) &= \mathbf{h}_2 = g_2(\mathbf{h}_1^T\mathbf{w}_2 + \mathbf{b}_2),\\
f_{\theta_3}^{(3)}(\mathbf{h}_2) &= \hat{\mathbf{y}} = g_3(\mathbf{h}_2^T\mathbf{w}_3 + \mathbf{b}_3),
\end{aligned}
\tag{3-2}
$$

where $g_i$ is a nonlinear operation, namely the activation function, $\mathbf{w}_i$ is a linear transformation, $\mathbf{b}_i$ is the biases, with $\theta_i$ consisting of $\mathbf{w}_i$ and $\mathbf{b}_i$, and $\hat{\mathbf{y}}$ is the output.

In neural networks, each function is represented by a layer. Following the example above, the input $\mathbf{x}$ is referred to as the input layer, whereas $f_{\theta_1}^{(1)}$ and $f_{\theta_2}^{(2)}$ are known as hidden layers, and $f_{\theta_3}^{(3)}$ or $\hat{\mathbf{y}}$ represents the output layer. To count the number of layers in a neural network, only the layers with connections between them are considered, omitting the first (i.e., the input layer). The number of layers gives the depth of the model, whereas the dimensionality of the hidden layers determines the width of the model.

### 3.1.2 Deep Neural Networks

Deep neural networks (DNNs) are neural networks composed of several hidden layers that amounts to modeling high-level abstractions from data. This is done in a hierarchical and incremental manner by stacking low level concepts on top of each other. Low level concepts are identified in the first layers of the network and enriched with the number of layers. The capacity of these models to automatically extract features reduces the need for feature engineering, the dominant approach until the advent of deep learning.

Although it was originally introduced over 20 years ago, it is only recently, with the availability of large data and growing availability of massive computational resources that DL has emerged as a powerful technique in the ML research domain. In the last two decades, DL-based solutions have considerably improved the state-of-the-art performance across a variety of application domains such as computer vision, natural language processing and bio-informatics [4, 7].

### 3.1.3 Levels of Supervision for Training Machine Learning Algorithms

Machine learning algorithms can be broadly classified as supervised and unsupervised. In supervised learning, the training data has a collection of inputs and expected outputs. Given sufficiently large datasets of labeled training examples, a deep neural network can represent functions of increasing complexity [5].

The goal of the supervised training is to learn to predict $\mathbf{y}$ from $\mathbf{x}$, usually by estimating $p(Y|X)$, where $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ is the set of input feature vectors and $Y = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}\}$ its observed targets. In supervised training, several examples of an input vector $\mathbf{x}$ are used as inputs to the neural network and the outputs are compared against the associated value or vector

$\mathbf{y}$. Then, the parameters of the machine learning model are adjusted to obtain an improved estimate for the target outputs.

Conversely, no labels are provided within the unsupervised type of machine learning. These techniques aim at extracting patterns and hidden structures from unlabeled data, which will be further used to draw inferences. By observing examples of an input vector $\mathbf{x}$, the unsupervised learning algorithm attempts to learn the probability distribution $p(X)$ implicitly or explicitly.

Semi-supervised methods lie between unsupervised and supervised learning and aggregate the techniques of both approaches. These methods rely on semi-labeled datasets that usually contain a small amount of labeled data and a large amount of unlabeled data.

Self-supervised learning is a type of unsupervised learning as it uses unlabeled data and learns on itself without supervision. However, instead of finding high-level patterns to analyze and cluster unlabeled datasets, self-supervised learning attempts to solve classification and regression tasks that are traditionally targeted by supervised learning.

### 3.1.4
### Cost Function

The cost function determines how inaccurate the model is for a given dataset and is considered an important aspect of the design of a deep neural network. The most commonly used cost functions are based on the maximum likelihood estimation (MLE) principle. The MLE algorithm attempts to maximize the probability of observing data given a specific probability distribution known as the likelihood function.

The likelihood function is defined as the joint density of the observed data as a function of the model's parameters. Consider $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ a set of $N$ data instances drawn independently from the same data-generating distribution denoted by $p_{\text{data}}(\mathbf{X})$. Let $p_{\text{model}}(X, \theta)$ be our parametric model with some learnable parameters denoted by $\theta$. The likelihood function, $L(\theta)$, is then given by:

$$\begin{aligned} L(\theta) &= p_{\text{model}}(X, \theta), && \text{(3-3)} \\ &= \prod_{i=1}^{N} p_{\text{model}}(\mathbf{x}^{(\mathbf{i})}, \theta) && \text{(3-4)} \end{aligned}$$

Usually, for analytical and numerical simplifications, the log likelihood function, $l(\theta)$, is used instead of the likelihood function. The maximum likelihood estimator for $\theta$ is thus defined as:

$$\theta_{\text{MLE}} = \arg\max_{\theta} l(\theta), \tag{3-5}$$

$$= \arg\max_{\theta} \sum_{i=1}^{N} \log p_{\text{model}}(\mathbf{x^{(i)}}, \theta) \tag{3-6}$$

$$\tag{3-7}$$

Recall that in supervised learning, the machine learning model learns to predict $\mathbf{y}$ from $\mathbf{x}$, usually by estimating $p(Y|X)$, where $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ is the set of input feature vectors and $Y = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}\}$ its observed targets. The MLE can be thought of as the cross-entropy between the training data and the model's predictions [5]. In this case,the MLE algorithm can be generalized to estimate the conditional probability, given by:

$$l(\theta) = \sum_{i=1}^{n} \log p_{\text{model}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}; \theta), \tag{3-8}$$

$$\theta_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{\text{model}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}; \theta) \tag{3-9}$$

As we demonstrate next, the Mean Squared Error (MSE), one of the most commonly used cost functions for regression, is derived from the MLE principle. For simplification purposes, we chose the linear regression example. In linear regression, the goal is to build a system that can take a vector $\mathbf{x}$ and predict the values of a scalar $y$. The data is described by a linear model $y = \mathbf{w}^T\mathbf{x} + \epsilon$, where $\mathbf{w}$ is a vector of parameters and $\epsilon \sim \mathcal{N}(\epsilon_i; 0, \sigma_\epsilon^2)$.

Consider that $\sigma_\epsilon^2$ is known and that the data points are drawn independently from the same fixed distribution. Given that $\sigma_\epsilon^2$ is Gaussian (i.e., normally distributed), the log likelihood of our model is also Gaussian and can be expressed by:

$$l(\mathbf{w}) = p(Y|X; \mathbf{w}), \tag{3-10}$$

$$= \sum_{i=1}^{N} \log \mathcal{N}(\epsilon_i; 0, \sigma_\epsilon^2), \tag{3-11}$$

$$= -\frac{N}{2} \log 2\pi\sigma_\epsilon^2 - \sum_{i=1}^{N} \frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma_\epsilon^2}, \tag{3-12}$$

where $N$ is the number of data points. Eliminating the terms that do not depend on $\mathbf{w}$ gives:

$$l(\mathbf{w}) \approx -\sum_{i=1}^{N} (y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2, \tag{3-13}$$

and the linear regression problem can be expressed following:

$$\mathbf{w}_{\mathrm{MLE}} = \arg\max_{\mathbf{w}} l(\mathbf{w}), \tag{3-14}$$

$$= \arg\max_{\mathbf{w}} -\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2, \tag{3-15}$$

$$= \arg\min_{\mathbf{w}} \sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2, \tag{3-16}$$

$$= \arg\min_{\mathbf{w}} \mathrm{MSE}(\mathbf{w}), \tag{3-17}$$

which proves that maximizing the log likelihood amounts to minimizing the mean squared error.

### 3.1.5 Gradient-Based Learning

The non-linearity of the neural networks causes the loss functions to become non-convex in the way that these models are usually trained by using iterative, gradient-based optimizers. Gradient descent, a first-order optimization algorithm, is the most commonly used approach. Starting from a random point, it takes steps in the opposite direction to the gradient of the cost function with respect to the network parameters. This process is repeated until a stopping criterion is satisfied, which is typically when a predefined number of iterations is reached. This optimization process is expressed as:

$$\theta = \theta - \alpha\nabla\theta,$$

where $\theta$ are the network parameters, $\alpha$ is the learning rate and $\nabla\theta$ are the gradients of the loss function with respect to the network parameters. This update step is performed for every iteration.

Standard gradient descent computes the gradients for the entire dataset to perform one update afterwards. In the case of convex error surfaces, it guarantees the convergence to the global minimum (or to a local minimum in the case of non-convex error surfaces). However, it can be extremely slow depending on a large dataset. Conversely, stochastic gradient descent (SGD) attempts to approximate the gradient by performing a parameter update for one randomly picked data sample on every iteration, which drastically reduces the training time. These high-frequency updates with high variance cause repeated fluctuations in the cost function, and the algorithm might end up overshooting close to the target actual minima.

To address these challenges, mini batch gradient descent tries to find a balance between the stability of standard gradient descent and the speed of SGD. The parameters update is performed considering a batch with a fixed size, usually referred to as a "mini batch". At each iteration, the algorithm trains on a different mini batch. Because of the way computer memory is laid out and accessed, mini batch sizes are usually a power of 2, typically ranging from 64 up to 1024, depending on the data size. As a result, the algorithm is less noisy than SGD and requires a smaller number of iterations to converge for large datasets.

Gradient descent with momentum is a method used to accelerate the training process carried out by SGD approaches. It is based on the exponential moving average of the previously computed gradients and the current one. The exponential moving average is a moving average that assigns a greater weight to the most recent values. The momentum term avoids the slow down caused by the oscillations of the cost function and prevents the network from getting stuck in local minimums. The rule update for gradient descent with momentum is expressed as:

$$\begin{aligned} \nu_t &= \beta\nu_{t-1} + (1-\beta)\nabla\theta, \\ \theta &= \theta - \alpha\nu_t, \end{aligned} \tag{3-18}$$

where $\beta$ is a hyperparameter ($\beta \in (0,1]$) that controls the exponential decay rate. Large values of $\beta$ averages over more older observations. The most common value for $\beta$ is 0.9 when it averages over the last ten iterations' gradients.

Adaptive moment estimation (Adam) [69] is an optimization algorithm that combines the advantages of two other extensions of SGD: adaptive gradient algorithm (AdaGrad) [70] and root mean square propagation (RMSProp) [71]. It was designed to accelerate the optimization process and to improve the performance of the algorithm. This is achieved by calculating an adaptive learning rate per-parameter being optimized from estimates of first and second moments of the gradients. The Adam's update rule is given by:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1)\nabla\theta, \\ \nu_t &= \beta_2 \cdot \nu_{t-1} + (1-\beta_2)\nabla\theta^2, \\ \hat{m}_t &= m_t/(1-\beta_1^t) \\ \hat{\nu}_t &= \nu_t/(1-\beta_2^t) \\ \theta &= \theta - \alpha \cdot \hat{m}_t/(\sqrt{\hat{\nu}_t} + \epsilon), \end{aligned} \tag{3-19}$$

where $m_t$ and $\nu_t$ denote the exponential moving averages of the gradient and of the squared gradient, respectively, $\beta_1$ and $\beta_2$ are hyperparameters ($\beta_1, \beta_2 \in (0,1]$) that control the exponential decay rates of these moving averages, and $\epsilon$ is a small positive constant. Recommended values for the exponential decay rates are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Since $m_t$ and $\nu_t$ are initialized as 0, they gain a tendency to be biased towards 0. This problem is fixed by computing the bias-corrected estimates $\hat{m}_t$ and $\hat{\nu}_t$.

Intuitively, the ratio $\hat{m}_t/\sqrt{\hat{\nu}_t}$ represents the spread (or simply uncertainty) in the gradient's history. The greater the uncertainty, the smaller the individual learning rate for each particular weight. Another advantage of Adam algorithm is that the magnitudes of parameter updates are invariant to rescaling of the gradient. This is because rescaling the gradients with a factor $c$ will scale $\hat{m}_t$ with a factor $c$ and $\hat{\nu}_t$ with a factor $c^2$, which cancel out. The authors show empirically that Adam is robust and well-suited to a wide range of non-convex optimization problems. It has become popular as one of the most effective and robust optimization algorithms in the field of deep learning.

## 3.1.6 Backpropagation

Gradient-based learning algorithms require the computation of the gradient of the cost function with respect to the parameters of the model. The backpropagation algorithm [72] is a method to compute the gradient of the cost (i.e., error) associated with an output $\hat{\mathbf{y}}$, that was produced by forward propagation of an input $\mathbf{x}$ through a feedforward neural network.

As the name backpropagation suggests, the calculation of the gradient proceeds backwards through the network. It is based on computational graphs, with each node corresponding to a variable. The method is highly efficient as it computes the chain rule of calculus instead of calculating the gradient of each node separately. The partial computations of the gradient from one node are reused in the computation of the gradient for the previous node.

For example, to compute the gradient of some scalar $z$ with respect to one of its ancestors $\mathbf{x}$ in the graph, it begins by observing that the gradient with respect to $z$ is given by $\frac{dz}{dz} = 1$. Then it computes the gradient with respect to each parent of $z$ in the graph by multiplying the current gradient by the Jacobian of the operation that produced $z$. This is continued in the backward direction until it reaches $x$. Gradients arriving from different paths in one node are simply summed up.

The backpropagation method enables every weight of the neural network to be individually updated to gradually reduce the error over many iterations. It was one of the first methods to demonstrate that neural networks could learn good internal representations and is widely used to train deep neural networks because of its efficiency.

## 3.1.7 Activation Functions

The activation function is an essential part of a neural network, which defines how the weighted sum of the inputs is transformed into the output of a particular node, introducing nonlinearity to the model. Two well-known activation functions are the hyperbolic tangent and sigmoid activation functions. Both have the nice properties of activation functions: nonlinearity, continuously differentiable, and having a fixed output range.

The curve of the hyperbolic tangent (tanh) function looks like a S-shape. It takes a real value as input and outputs values between -1 and 1. The tanh function is expressed as: $g(x) = (e^x - e^{-x})/(e^x + e^{-x})$, where $e$ is a mathematical constant also known as the Euler's number. The larger the input (i.e., more positive), the closer the output will be to 1, whereas the smaller the input (i.e., more negative), the closer the output will be to -1.

The sigmoid activation function, also called the logistic function, has the same S-shape curve as the tanh function. It takes a real value as input and outputs another value between 0 and 1. The sigmoid function is denoted by: $g(x) = 1/(1 + e^x)$. Similar to the tanh function, large values are squashed into 1, while small values are transformed to the value 0.

Activation functions such as the tanh and sigmoid functions that saturate (i.e., become very flat) for both very negative and very positive inputs came out to be a major reason behind the vanishing gradients, an issue where the network is unable to back-propagate useful gradient information to the first layers. The vanishing gradients occurs because the backpropagation of the gradients of the error is based on the chain rule, which is basically multiplication operations. Repeated multiplication with small values renders gradients very close to zero in earlier layers of the network.

The ReLU is the default activation function recommended for use in modern neural networks [5]. It is defined as the positive part of its argument: $g(\mathbf{x}) = \max(0, \mathbf{x})$. The ReLU gradient is either 0 or 1 for the entire range. This effectively addresses the vanishing gradient problem as the gradient does not become increasingly small as it back-propagates through the network. Moreover, ReLU is less computationally expensive than sigmoid and tanh since it involves simpler mathematical operations.

Some drawbacks with ReLU are that it is non-zero-centered and is not differentiable at zero, which limits its use to the hidden layers. Additionally, ReLU units can be vulnerable during training and die, which means that they output the same value and remain inactive no matter the input supplied. Once it ends up in this state, the unit is unlikely to recover, as the function gradient at zero is also zero.

Leaky ReLU is an improved version of the ReLU activation function. Instead of a flat slope, it has a small slope for negative values. The mathematical formula for Leaky ReLU is $g(x) = \max(0.05x, x)$. It allows a small positive gradient when the unit is not active and is the most common method to solve the "dying ReLU" problem.

The ReLU and Leaky ReLU activation functions only partly solve the vanishing gradients as there is no guarantee on the values of the model parameters and on the representations of input data throughout the network.

### 3.1.8
### Regularization

Generalization is an important problem in machine learning, which is concerned with the performance of a trained model on unseen data. When training deep neural networks with a large number of parameters and limiting training data, overfitting becomes particularly critical. Goodfellow et al. [5] defined regularization as: *"any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error"*.

Many methods have been developed for reducing the generalization error. Some of them, such as the weight decay method, add extra constraints on a machine learning model. In the weight decay approach, a penalty term is added to the cost function to bias the learning model towards solutions that have smaller weights. The smaller the weights are, the smaller the change in the network behavior if we change a few random inputs. This forces the network to put weight on fewer features that are seen across the training set, improving its generalization ability.

The most common form of weight decay regularization is the sum of the squared weights, also known as the L2-norm. For example, to perform linear regression with L2-norm regularization, we add the squared magnitude of the weights to the cost function $\mathcal{L}$:

$$\mathcal{L}(\mathbf{w}) = \text{MSE}(\mathbf{w}) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

where $||.||_2^2$ is the squared L2-norm, $\mathbf{w}$ is the set of parameters and $\lambda$ is the regularization constant, a non-negative hyperparameter that controls the trade-off between weights being small and fitting the training data. The half and the square of the L2-norm are taken for mathematical convenience.

The L1-norm is another form of weight decay regularization. It adds the absolute value of the magnitude of the weights as a penalty term to the loss function:

$$\mathcal{L}(\mathbf{w}) = \text{MSE}(\mathbf{w}) + \lambda||\mathbf{w}||_1$$

where $||.||_1$ is the L1-norm and $\lambda$ is a hyperparameter that controls the strength of the regularization, as with L2-norm regularization.

While L1-norm regularization tends to confine the weights near to zero, resulting in solutions that are sparer, L2-norm regularization tends to shrink the weights evenly. The sparsity induced by L1-norm regularization works as an automatic feature selection mechanism, as it can eliminate unimportant variables associated with weights that become zero. On the other hand, L2-norm regularization can be helpful when the output variable is the function of the whole set of input variables. In addition, L2-norm regularization leads to more accurate customized models as it disperses the error terms in all the weights.

Dataset augmentation is another regularization technique that artificially increases the size and diversity of the training set by applying transformations to existing training examples without invalidating the distribution of the data itself. Creating modified data can be straightforward, as for a large-scale problem in computer vision concerned with object recognition in images. New and different synthetic examples can be created by simple operations such as rotating, translating, or scaling an image.

While data augmentation techniques have proven to be effective for computer vision applications, in other cases, whether those invariances do not apply or if the training set distribution is unknown, it might be difficult to generate new fake data. In deep learning, physics-informed data augmentation techniques have been successfully applied to generate additional training data that are physically meaningful [73, 74, 74, 75].

Adding noise in the hidden units of a neural network can also be seen *"as a form of dataset augmentation at multiple levels of abstraction"* [5]. This strategy can also prevent overfitting associated with the co-adaptation of feature detectors. Dropout [76] (from dropping out units) is a regularization technique that adds binary noise to the hidden units, randomly setting some of them to zero. It works by temporarily deleting neurons from the network along with all their incoming and outgoing connections, with a fixed probability $q = 1 - p$. Applying dropout prevents any neuron from relying excessively on the output of any other neuron, forcing it to create useful features on its own. At test time, the weights are multiplied by $p$, which ensures that the expected output is the same as the actual output for any hidden unit.

In addition, dropout provides a computationally inexpensive and effective approximation of an averaging method, which involves training and evaluating a bagged ensemble of many neural networks. During training, for each presentation of each training example, each unit is randomly removed from the network. This is equivalent to sampling and training a subnetwork from it for each presentation of each training case. With dropout, a neural network with $n$ units, can be seen as an ensemble of $2^n$ possible subnetworks.

Early stopping is an effective and the most commonly used technique for reducing generalization errors. The method involves monitoring the train and validation set errors as the number of training iterations over the dataset (or epochs) goes by. During training, the algorithm learns and the error on the train set naturally decreases, as does the validation set error. The training procedure is interrupted when the validation set error begins to rise. Beyond this point, the model starts to overfit the training data. In practice, the algorithm stores a copy of the model parameters every time the error on the validation set goes down and terminates when no improvement is observed for some pre-specified number of iterations. The model parameters that resulted in the best validation error are then used instead of the latest parameters.

## 3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [3] are a class of DL models that revolutionized computer vision [77], achieving outstanding results in processing image tasks [4, 6, 7]. Unlike conventional neural networks, CNNs make the explicit assumption that inputs have a spatial-temporal structure of multidimensional arrays, such as images, videos, audio signals, and volumetric data (i.e., a three-dimensional grid of pixels). These types of data have two basic properties in common: first, they have one or more axes for which order is important; second, there is one axis, namely the channel axis, that is dedicated to accessing different perspectives of the data.

Convolution is a specialized kind of linear operation that replaces the matrix multiplication performed in conventional neural networks and preserves this notion of ordering. A neural network is considered a convolutional neural network if it uses convolution in at least one of its layers.

The convolution operator is denoted by the symbol $*$ and is expressed as the integral of the product of two functions $x(t)$ and $w(t)$. To define a convolution operation, each function is first expressed in terms of an auxiliary variable $a$. The function is then reversed as $w(a) = w(-a)$. In this context, a convolution operation $s(t)$ can be described as the area under the function $x(a)$ weighted by the function $w(-a)$, and shifted by amount $t$:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\,da, \tag{3-20}$$

For discrete functions, the convolution operation is given by:

$$s(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a) \tag{3-21}$$

In a convolutional layer, the functions $x$, $w$, and $s$ are multidimensional vectors ($\mathbf{x}$, $\mathbf{w}$, $\mathbf{s}$) and the summation is implemented over a finite number of vector elements. The vector $\mathbf{x}$ is referred to as the input image or the input feature map, $\mathbf{w}$ is known as the kernel (or filter), and the output $\mathbf{s}$ is referred to as the output feature map.

For simplicity, many neural network libraries implement the convolution as a cross-correlation function, which differs from convolution only in that the kernel is not reversed and shifted. The kernel slides over the whole input feature map. When training a CNN, the results will be the same, the only difference is that the resulting weights are flipped from each other.

The output for each location $(i, j)$ is calculated as the sum of the products between each element of the kernel and the input element that it overlaps (i.e., element-wise multiplication). In this new formulation, the convolutional operation between an image $\mathbf{x}$ and a two-dimensional kernel $\mathbf{k}$ of size $m \times n$ is defined as:

$$\mathbf{s}(i,j) = (\mathbf{x} * \mathbf{k})(i,j) = \sum_{m}\sum_{n} \mathbf{x}(i+m, j+n)\mathbf{k}(m,n) \tag{3-22}$$

The kernel can be a 1-dimensional (1D), a 2-dimensional (2D) or a 3-dimensional (3D) array. The number of dimensions defines the number of directions in which the kernel will move over the input data performing an element-wise multiplication. A 1D kernel moves in one direction and is mostly used on time-series data. The most common type is the 2D kernel to process 2D image data. A 3D kernel slides in three dimensions as opposed to two dimensions with 2D kernels and is usually applied to 3D medical images and video-based data.

The area in the input space that a particular kernel is looking at is called the receptive field. Typically, more than one kernel is used to produce different output feature maps. Thus, this procedure is repeated to convolve the input feature map with each distinct kernel. If there is more than one input feature map, the output feature maps will be summed up element-wise to produce the final output feature map.

In some cases, either for computational efficiency or because one wishes to reduce the resolution of the output, it is desirable to move the kernel window more than one element at a time, skipping the intermediate locations. The number of elements traversed per slide is known as the stride.

The interaction of the filter with the border of the image causes a reduction in the size of the input feature map. Many successive convolution operations in a deep CNN will eventually reduce the size of the feature maps to zero. Extra pixels are added around the perimeter of the input image to preserve the spatial dimensions of the input feature map post-convolution. These extra pixel values are typically set to zero. The number of pixels added to an input image is referred to as padding.

A typical convolution operator is controlled by three parameters: kernel size, stride and padding. The size of the image post-convolution ($n_{\text{out}}$) is a function of these parameters given an input image:

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1, \tag{3-23}$$

where $n_{\text{in}}$ is the size of the input image along a given axis, $p$ is the padding, $k$ is the kernel size, and $s$ is the stride. Figure 3.1 illustrates an example of a 2D convolution operation performed on an image.

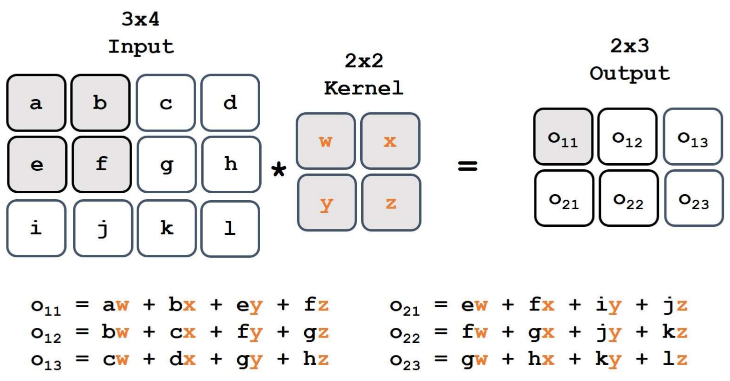

3x4
Input
a
b
c
d
e
f
g
h
i
j
k
l
2x2
Kernel
w
x
y
z
2x3
Output
$o_{11}$
$o_{12}$
$o_{13}$
$o_{21}$
$o_{22}$
$o_{23}$
$o_{11} = aw + bx + ey + fz$
$o_{12} = bw + cx + fy + gz$
$o_{13} = cw + dx + gy + hz$
$o_{21} = ew + fx + iy + jz$
$o_{22} = fw + gx + jy + kz$
$o_{23} = gw + hx + ky + lz$

Figure 3.1: Example of a convolution operation performed on a tensor of size $3 \times 4$ using a kernel of size $2 \times 2$, stride of 1 and zero padding. The output is a tensor of size $2 \times 3$.

Influenced by the feature hierarchy of the human visual system, CNNs are based on three fundamental ideas: sparse interactions, parameter sharing, and equivariance [5]. Unlike traditional fully connected (FC) neural networks, which allocate one weight per unit in the previous layer for each unit in the following layer, in a CNN, the interactions are sparse. This is done by making the kernel smaller than the input, which enables the network to capture local and small features, such as an edge, a corner or a part of an object. As long as these features are successively fed into the next layer, they become progressively and hierarchically more complex. As a result, when compared to dense matrix multiplication from FC neural networks, the convolution operation is more efficient in terms of memory requirements and statistical efficiency (i.e., requires fewer training examples to learn).

In a convolution operation, the same parameter value is applied to more than one input unit. The consequence of the parameter sharing is the equivariance to translation (or translation invariance). A function $f(x)$ is said to be equivariant to a function $g(x)$ if $f(g(x)) = g(f(x))$. This means that the result of a convolution applied to a translated input, will be the same as if we applied a translation to the convolution of this input. This property is important to detect features (e.g., edges and eyes) regardless of its exact location in an image.

The convolution operation is typically followed by a non-linear transformation and, sequentially, by a sub-sampling operation. The sub-sampling operation, often referred to as pooling, is performed to create a low-resolution version of the input by replacing values within sub-regions in the input image with a summary statistic calculated over them. Pooling helps to increase noise-resilience and translation invariance. Moreover, it reduces the computational cost by reducing the number of learnable parameters.

Max-pooling [78] is a common type of pooling, which takes the maximum value within each patch of the input feature map to create a new element in the output matrix. Figure 3.2 illustrates an example of a max-pooling operation.

There are three stages that compose a typical convolutional layer: an affine transformation performed by the convolution operator, a non-linear transformation, and the pooling function (Figure 3.3). A standard CNN is

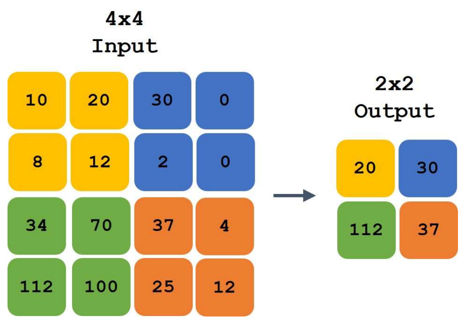


Figure 3.2: Example of a max-pooling operation performed on a tensor of size $4 \times 4$. The output is a tensor of size $2 \times 2$.

composed of repeatedly stacking these convolutional layers, optionally followed by a fully connected (FC) layer, depending on the required task. The FC layer is usually responsible for mapping the extracted features into the final output (e.g., the scores for each category in a classification task).

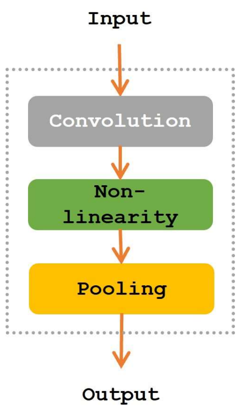


Figure 3.3: The components of a typical convolutional layer.

## 3.2.1 Very Deep Convolutional Neural Networks

Depth has been identified as a key factor in the success of DL models in dealing with complex concepts representations, which are frequent in most computer vision applications. However, the assumption that adding more layers increases the network performance holds true up to a certain point [79]. Training very deep neural networks can be a challenging task as they are more likely to suffer from overfitting, vanishing gradients, loss in information flow, long training time, and other issues that might emerge as the number of layers of the network increases [80, 81, 82, 83, 84, 79]. In recent years, different approaches have been proposed to overcome these problems and enable deeper neural networks.

Batch Normalization (BN) [85] is a standard practice technique to improve the performance and speed up the convergence in the training of deep neural networks. The idea behind BN is to avoid the erratically changing input values of the layers as the model parameters are updated, a phenomenon referred to by the authors as internal covariate shift (ICS). This is accomplished by including additional layers that control the first two moments of the layers' input distributions. These extra parameters are learnt along with the original model parameters.

When training with BN, the mean and standard deviation of each input variable to a unit are computed with respect to each mini-batch. These statistics are then used to normalize the input to each unit to have zero mean and unit variance. By stabilizing the distribution of inputs, it ensures that the expectation of a unit on the distribution of its inputs will not change during the weights updates, which speeds up the learning process. Moreover, BN prevents the activation function from amplifying into larger and sub-optimal output values and getting stuck in the saturated regimes of non-linearities. Ultimately, similar to dropout, by adding some noise to the values within a mini-batch, BN helps with the generalization capability of the model.

Concurrently, much research has focused on the definition of new architectures of neural networks to reduce complexity and further improve accuracy in DL. Residual Neural Networks (ResNets) [79] were motivated by experiments that demonstrated that the accuracy gets saturated and degrades rapidly as the network depth increases. The authors conjecture that, similar to the gradients in a backward pass, the information on a forward flow can also vanish and "wash out" as it passes through many successive layers, which results in a higher training error. Residual (or identity) blocks attempt to ease the training of deep neural networks by reformulating the layers as learning residual functions.

The ResNet architecture is simpler and more efficient when compared to its predecessors. ResNets are formed by stacking several residual blocks together. Residual blocks are implemented by means of skip connections (aka identity connections), where the inputs are connected back to their outputs with the goal of creating identity mappings. If we denote the underlying mapping as $H(x)$, the residual mapping is then expressed as $F(x) = H(x) - x$, and the original mapping becomes $F(x) + x$. A typical residual block is shown in Figure 3.4.

By enabling the identity mapping, the residual blocks allow the network to be considerably deeper as these extra layers do not affect its performance. A higher layer will perform as well as a lower layer, and better whenever it learns new information from data. As the authors stated: "*a deeper model should have a training error no greater than its shallower counterpart.*" Additionally, the short connections allow the propagation of information directly from the shallow to the deeper layers, which helps with the vanishing gradient challenge and loss in information flow in training very deep networks.

## 3.3
## Deep Learning Methods in Inverse Problems

Many computer vision tasks are concerned with artificially generating new examples that are similar to those in the training data. This group of generative tasks is often referred to as synthesis and sampling [5]. In some

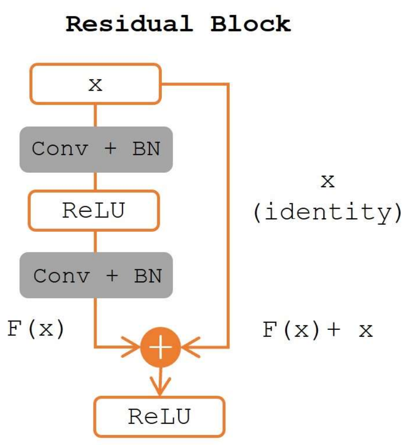


Figure 3.4: A residual block has two convolutional layers followed by a batch normalization layer and a ReLU activation function. The short connection skips these two convolution operations and adds the input directly before the last ReLU activation function.

cases, the generator process can be conditioned to some specified constraints from users, such as a class label, an image, or a text description (e.g., synthesizing images from a text description). Most of these tasks are inherently ill-posed, when there is no single correct output and a large amount of variation in the output is desirable. Image synthesis and sampling methods have been extensively explored for a wide variety of applications. Among them, CNN-based solutions have been empirically shown to be useful and have become the dominant approach.

When the input and output data are images, the problem is known as image-to-image translation [86], in which the algorithm attempts to learn a mapping or "translation" between two images (typically taken from different domains) so that the output image has some particular desired content of the input image (e.g., structure, style, or texture). Isola et al. [86] defined image-to-image translation as "*the task of translating one possible representation of a scene into another, given sufficient training data.*" Examples of applications from computer vision include synthesizing real-world images from a sketch [86, 87, 88] and adapting satellite images into map routes [86, 89].

One challenging type of image-to-image problem is semantic image segmentation, which involves, as the name suggests, segmenting parts of an image that belong to the same class by assigning a predefined class label for each pixel in the input image. This task has application for a wide variety of domains, ranging from autonomous cars [90] to medical imaging analysis [91, 92, 93].

Some image-to-image translation tasks take a noisy version of an image to produce an image that represents the "best guess" of the original image. These inverse problems in imaging, commonly referred to as image recovery or image reconstruction [94, 95, 96], are concerned with reconstructing an unknown signal/image $\mathbf{y}^* \in \mathbb{R}^N$ from a set of measurements $\mathbf{x} \in \mathbb{R}^P$, which are related via a forward model $f$, in some cases known, typically non-invertible, of the form:

$$\mathbf{x} = f(\mathbf{y}) + \eta \tag{3-24}$$

where $\eta \in \mathbb{R}^P$ represents the noise vector.

Additional information representing a prior belief about the sought solution is typically applied to overcome the ill-posed conditions of inverse problems [22]. In this case, the goal is to find a vectorized image $\mathbf{y}^*$ that match the observations $\mathbf{x}$, which is likely given the prior knowledge.

Examples of imaging tasks that fit under this framework are image inpainting, image denoising and super-resolution (SR). SR has become an important class of image processing techniques in computer vision. SR focuses on the task of recovering high-resolution images from low-resolution images. This problem is inherently ill-posed because high frequency information is lacking, which is usually solved by exploiting prior information.

The seismic inversion problem, the object of analysis in this study, fits under the latter group of inverse problems, more specifically, the ones in which an analytical model of the laws underlying the measurement process is known. Along with other application domains, such as medical imaging, deep learning techniques are currently transforming inverse problems' methods in geophysical imaging. Next, we outline CNN-based models that are commonly used to address inverse problems in imaging and some related problems.

### 3.3.1 Fully Convolutional Networks

Because image reconstruction involves predicting an output value for every pixel in the input image (i.e., pixels-to-pixels), this task is commonly referred to as dense prediction. Long et al. [91] first introduced the framework of an architecture without fully connected layers, namely, the fully convolutional network (FCN). Because FCNs only perform convolution operations, they can process arbitrary-sized inputs and be trained end-to-end for individual pixel prediction.

A naive approach is to directly learn a mapping from the input image to its corresponding output image through consecutive convolutional layers while preserving the original full spatial resolution. However, this would be computationally costly as the number of channels of almost any CNN rapidly increases with depth. Instead, a standard FCN has an encoder-decoder architecture, a special case of feedforward neural networks.

Encoder-decoder networks include a middle bottleneck layer (i.e., a lower dimensional hidden layer) to force the compression of the input into a lower dimensional representation of the latent space, reducing the number of parameters of the network and consequently its computational cost.

The encoder-decoder architecture is symmetric and composed of two parts: a contracting path (i.e., encoder) and an expansive path (i.e., decoder). The encoder part ($e$) parameterizes a high-dimensional input image ($\mathbf{x}$) into a low-dimensional latent space representation ($\mathbf{h}$): $\mathbf{h} = e(\mathbf{x})$. One can roughly think of an encoder as a classification network that maps the input image into an embedded representation of this image, in this case a class (e.g., apple or orange). Then, in the decoder part ($d$), $\mathbf{h}$ is used as input to the decoder part to reconstruct a new high-dimensional output image: $\mathbf{y} = d(\mathbf{h})$. Figure 3.5 illustrates in a didactic manner a representation of an encoder-decoder architecture.

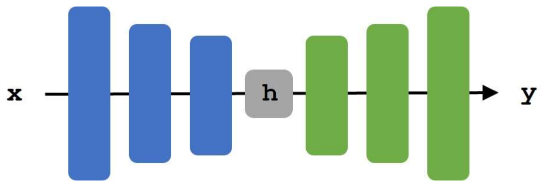


Figure 3.5: A schematic representation of the encoder-decoder architecture given an input $\mathbf{x}$ and an output $\mathbf{y}$. Each box represents a multi-channel feature map. The encoder is the first half of the diagram (in blue), the latent code representation is in the middle (in gray) and the decoder is the second half (in green). Based on [86].

A known benefit of encoder-decoder networks concerns the reception field size. Recall that the receptive field in CNNs is the region of the input space that affects a particular unit of the network. Given a constant filter size, down-sampling broadens the receptive field with respect to the input. This is important to capture information about larger objects since larger regions of the input space will affect a particular unit in the output layer.

A typical encoder is composed of a series of convolution blocks followed by a max-pooling operation to reduce (i.e., down-sample) the spatial dimensions of the input. Conversely, in the decoder part, the size of the feature maps can be increased (i.e., up-sampled) by a fast deterministic method, such as the nearest and bi-linear operators. The first simply copies the values from the nearest pixels, while the second applies a linear interpolation from the values of nearby pixels. Alternatively, transposed convolution (i.e., fractionally-strided convolution) [97] attempts to recreate the original input before the convolution operation. By replacing the deterministic simple scaling with a convolution operation, it enables the network to learn the inverse to the convolution function by itself.

Given an input tensor of size $h \times w$ and a kernel of size $k \times k$, a transposed convolution operates as follows. The kernel window slides $w$ times in each row and $h$ times in each column, resulting in a total of $h \times w$ intermediate results. The intermediate results are tensors initialized as zeros. Each intermediate tensor is computed by multiplying each element in the input tensor by the kernel so that the resulting tensor replaces a portion of each intermediate tensor. This portion corresponds to the position of the element in the input tensor used for the computation. The final output tensor is the sum of all the intermediate results. The output is a tensor of size $(h + k - 1) \times (w + k - 1)$.

In transposed convolution, the padding parameter specifies the number of border elements to be removed from the output. Stride is used to determine how long the kernel jumps for each intermediate result. Figure 3.6 shows an example of a transposed convolution on a tensor of size $2 \times 2$ using a kernel of size $2 \times 2$, stride of 1 and zero padding. The expansion of the condensed intermediate feature maps is usually followed by regular convolution operations.

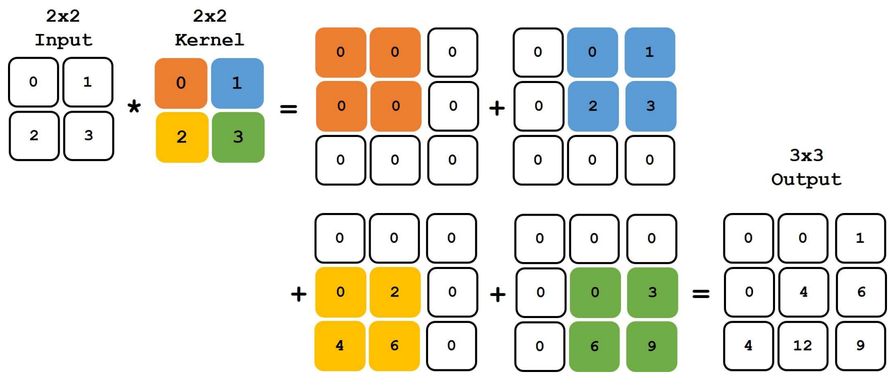


Figure 3.6: Example of a transposed convolution operation performed on a tensor of size $2 \times 2$ using a kernel of size $2 \times 2$, stride of 1 and zero padding. The output is a tensor of size $3 \times 3$.

The combination of the encoder and decoder modules promotes the reconstruction of images that look similar to the input ones. Auto-encoders are a common type of encoder-decoder architecture whose goal is to approximate the identity function. While compressing the input into an embedded representation, it aims at preserving as much information as possible to be able to reconstruct the output in the decoder part. This is achieved by minimizing the distance between the input and the reconstructed output on the training data. Auto-encoders are particularly useful for dimensionality reduction and for representation learning [98].

In [91], the authors proposed the adaptation of well succeeded image classification networks (e.g., AlexNet and GoogLeNet) to work as encoders, appending a decoder network with transposed convolutional layers to up-sample the latest feature maps into a dense segmentation map. This adaptation involves turning fully connected layers into convolutional layers. Skip connections from earlier layers in the network were added to up-sampling layers to help recover the full spatial resolution at the network output.

In image reconstruction problems, it is often beneficial to use skip connections to shuttle information directly across the network, since there is a great deal of low-level information shared between the input and output. Even though ResNet was originally proposed for image classification, it has been used in many image reconstruction applications [99, 100]. Networks with more complicated hierarchical skip connections, such as U-Net [92], are also commonly used.

U-Net [92] (named after its U shape) is a well-known encoder-decoder network whose architecture was primarily used for biomedical images segmentation. While up-sampling, it also concatenates (instead of summing as FCN) the higher resolution feature maps from the encoder part with the up-sampled feature maps at the same resolution. With these extra connections, features learnt at different stages of the encoder are projected onto the output pixel space. These skip connections are important to avoid spatial information from getting lost while being compressed in the encoding stage. A simplified representation of the U-Net architecture is shown in Figure 3.7.

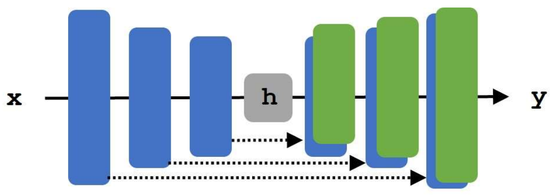


Figure 3.7: A simplified representation of the U-Net architecture [92] given an input $\mathbf{x}$ and an output $\mathbf{y}$. Each box represents a multi-channel feature map. The blue boxes from the encoder part of the network are copied and concatenated with up-sampled feature maps at same level, as represented by the green boxes. Based on [86].

### 3.3.2 Image Matching

Recall that in image reconstruction tasks, we are interested in finding a vectorized image $\mathbf{y}^* \in \mathbb{R}^N$ from measurements $\mathbf{x} \in \mathbb{R}^P$, of the form $\mathbf{x} = f(\mathbf{y}) + \eta$, where $\eta \in \mathbb{R}^P$ represents the noise vector and $f$ is the forward measurement operator.

DL-based methods to solve image inverse problems typically minimize a cost function composed of a data-fit term, which measures if the predicted image $\mathbf{y}^*$ is a good fit to the observations, and a regularizer, which measures the lack of conformity of $\mathbf{y}^*$ to a prior model, and promotes images with desirable properties. This optimization procedure usually involves image matching, i.e., the computation of a difference criterion between a reference image and the predicted image or the synthesized response obtained from the predicted image via the forward model.

A per-pixel classification or regression loss assumes an unstructured output space, where pixels are conditionally independent of each other. The most common types are either the mean squared error (MSE) or cross-entropy between the output and ground-truth images. Alternatively, the mean absolute error (MAE) is used instead of MSE to not over-penalize larger errors and produce sharper results.

Structure losses, on the other hand, penalize the joint distribution of the output. These metrics have a higher correlation with the human perception of visual quality when compared to unstructured losses. In this case, we assume that a pixel value is influenced by the values of nearby pixels in an image.

A popular choice is the structural similarity index (SSIM) [101], which is based on a weighted combination of three comparative measures: luminance ($l$), contrast ($c$) and structure ($s$). The SSIM metric is calculated over a window of an image. Given two windows $x$ and $y$ of common size, the SSIM metric is given by:

$$
\begin{aligned}
l(x,y) &= \frac{(2\mu_x\mu_y + c_1)}{(\mu_x^2 + \mu_y^2 + c_1)} \\
c(x,y) &= \frac{(2\sigma_{xy} + c_2)}{(\sigma_x^2 + \sigma_y^2 + c_2)} \\
s(x,y) &= \frac{(\sigma_{xy} + c_2/2)}{(\sigma_x\sigma_y + c_2/2)} \\
\mathrm{SSIM}(x,y) &= l(x,y) \times c(x,y) \times s(x,y) \\
\mathrm{SSIM}(x,y) &= \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}
\end{aligned}
\tag{3-25}
$$

where $\mu_x$ and $\mu_y$ denote the averages of $x$ and $y$, respectively, $\sigma_x$ and $\sigma_y$ are the variances of $x$ and $y$, respectively, $\sigma_{xy}$ represents the covariance of $x$ and $y$, and $c_1$ and $c_2$ are small constants.

In [102], the authors studied the effects of different types of loss functions applied to image restoration tasks (e.g., super-resolution), including MSE, MAE, SSIM, and MS-SSIM. The latter is an extension to SSIM performed at multiple scales through a multi-step down-sampling process. They proposed a novel loss function by combining MAE and MS-SSIM metrics. In the performed experiments, the proposed loss function attained the best quality, followed closely by the MAE loss used on its own.

In recent years, feature-wise losses have been explored to increase the perceptual visual quality of the networks' output [103, 104, 105, 106]. The feature-wise loss is based on differences between visual representations computed in the feature space of a trained CNN, such as the VGG network [107] trained on the ImageNet dataset [108]. By comparing distances in the feature space, the network is encouraged to produce images that have similar feature representations rather than generating outputs that match the ground truth images. For example, Gatys et. al. [103] proposed a loss function based on statistics computed over the extracted feature representations to match the style (i.e., texture) of a reference image, a problem in computer vision known as style transfer.

Rather than hand-crafting a loss function, the Generative Adversarial Network (GAN) [5] can learn a function that reflects the distance between the ground truth distribution and the distribution of the data generated by the network. Given a deep neural network of the generative type $G : \mathbf{z} \in \mathbb{R}^K \to \mathbf{y} \in \mathbb{R}^N, K \ll N$ that accurately models the prior information on the signal we are interested in, it is straightforward to decode an image $\mathbf{y}^* = G(\mathbf{z})$ that best fits the measures in some sense, for example, by minimizing $\|\mathbf{x} - f(G(\mathbf{z}))\|_2^2$.

The GAN framework is composed of two networks, namely generator ($G$) and discriminator ($D$). During training, the two networks act like players in a competitive game. The generator's goal is to fool the discriminator by producing fake samples that follow similar distribution to the actual data distribution. Meanwhile, the discriminator tries to distinguish real data from fake data produced by the generator. In standard GAN, the generator takes random noise ($\mathbf{z}$) as input and maps it into a fake image. The discriminator is a classifier that takes an image $\mathbf{y}$ as input and outputs if it is real or fake.

The objective of a standard GAN can be expressed as:

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_y[\log D(\mathbf{y})] + \mathbb{E}_z[\log(1 - D(G(\mathbf{z}))] \tag{3-26}$$

where G tries to minimize this objective against D tries to maximize it.

Conditional GAN (cGAN) is a variation of GANs in which the generator and discriminator are conditioned on auxiliary data. In this context, the generator learns a conditional mapping from an observed image $\mathbf{x}$ plus a random noise vector $\mathbf{z}$ to an output image $\mathbf{y}$. The objective of a cGAN can be expressed as:

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{x,y}[\log D(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{x},\mathbf{z}}[\log(1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z}))], \tag{3-27}$$

where G tries to minimize this objective against D tries to maximize it.

Conditional GANs are explored in [86] to solve image-to-image translation problems. The idea behind the paper is that a general-purpose solution to image-to-image translation problems can be built by replacing the hand-engineering of the problem-specific mapping function with the high-level loss function automatically learnt by GANs.

### 3.3.3 Deep Learning Models to Solve Geophysical Inverse Problems

Following its success in imaging inverse problems from computer vision, such as super-resolution and image denoising, deep learning has received considerable attention in the context of geophysical inverse problems. The central idea of these approaches is to train a deep neural network to work as a surrogate for the inverse mapping or, in other words, to train a network that takes in recorded seismic data (i.e., measurements) and reconstructs the corresponding parameters of the subsurface model.

In seismic inversion, the measurements are typically denoted by $\mathbf{d}$, whereas $\mathbf{m}$ is used to refer to the parameters of the subsurface model. In this formulation, the goal of the seismic inversion problem is to find the parameters of the subsurface model $\hat{\mathbf{m}} \in \mathbb{R}^N$ from measurements $\mathbf{d} \in \mathbb{R}^P$, of the form $\mathbf{d} = f(\mathbf{m}) + \eta$, where $\eta \in \mathbb{R}^P$ represents the noise vector and $f$ is the seismic forward model.

Related work in this study can be organized by broadly dividing the ML-based methods according to their level of supervision. Previously, seismic inversion problems were tackled using fully supervised approaches [109, 110, 111, 30, 36]. An example of annotated data in the seismic inversion problem is the acoustic impedance log at the well locations and the corresponding seismic data at the same locations. In this supervised approach, the network learns from the labeled dataset empirically by minimizing an objective function following:

$$\hat{\mathbf{m}} = f_\theta^{-1}(\mathbf{d}), \tag{3-28}$$

$$\mathcal{L}_{\text{M}} = ||\hat{\mathbf{m}} - \mathbf{m}||_2^2 \tag{3-29}$$

$$\arg\min_\theta \mathbb{E}_{\mathbf{m},\mathbf{d}}[\mathcal{L}_M], \tag{3-30}$$

where $\|.\|_2^2$ is the squared L2-norm, $(\mathbf{m}, \mathbf{d})$ is a sample from the training dataset, $f_\theta^{-1}$ is the inverse network and $\theta$ are the parameters to be estimated. The loss $\mathcal{L}_{\mathrm{M}}$ is referred to as model misfit, which measures the difference between the predicted subsurface model $\hat{\mathbf{m}}$ and the ground truth subsurface model $\mathbf{m}$.

The fully supervised framework may suffer from generalization issues due to the limited availability of ground truth models corresponding to measured data, which hinders them from practical application in exploration geophysics. As an alternative, including domain knowledge makes it possible to enhance well-log or training data based on geological processes.

In [28, 11, 30, 31, 32, 112, 29], a geological modeling algorithm (e.g., geostatistical simulation and process-mimicking methods) is used to synthetically generate multiple realistic subsurface models that represent the expected target spatial field, from which the forward operator is simulated to devise pair-wise instances for training a machine learning model.

Once trained on the synthetic examples, the network is used to make predictions on a real dataset outside the spatial domain of the training data. However, the synthetic geological models that compose the training dataset must be exhaustive and representative of the system under study, which is often difficult to obtain. If either the measurement process changes (e.g., source wavelet) or the target geological setting is different from the training data (e.g., subsurface models with different spatial correlations or facies proportions), a new network needs to be trained. This process can be very laborious and subject to errors.

In unsupervised and semi-supervised learning, vast amounts of unlabeled seismic data are applied in both the input and output layers of the network as a proxy for the ground truth. In the physics-driven learning framework, geophysical constraints are incorporated into the network architecture to guide the training process. Karpane et al. [33] refer to this paradigm that attempts to integrate scientific knowledge and data science as theory-guided data science.

The physics-driven framework is usually composed of two neural networks: one to learn an inverse mapping $f_\theta^{-1} : \mathbf{d} \in \mathbb{R}^P \to \mathbf{m} \in \mathbb{R}^N$, and another one to perform the forward simulation $f_\omega : \mathbf{m} \in \mathbb{R}^N \to \mathbf{d} \in \mathbb{R}^P$. The computational graph of the forward simulation keeps track of dependencies of the arithmetical operations, stores intermediate results and computes the gradient of a loss function based on the mismatch between observed and simulated seismic data:

$$\begin{gathered}\hat{\mathbf{m}} = f_\theta^{-1}(\mathbf{d}),\\ \mathcal{L}_{\mathrm{D}} = \|f_\omega(\hat{\mathbf{m}}) - \mathbf{d}\|_2^2\\ \arg\min_\theta \mathbb{E}_{\mathbf{d}}[\mathcal{L}_{\mathrm{D}}],\end{gathered}$$

where $\mathbf{d}$ is an unlabeled sample from the training dataset, $f_\omega$ is the forward simulation network, $f_\theta^{-1}$ is the inverse network and $\theta$ are the parameters of $f_\theta^{-1}$ to be estimated.

In general, the forward simulation network $f_\omega$ has non-learnable parameters. The data misfit $\mathcal{L}_{\mathrm{D}}$ measures the difference between observed measures and synthetic responses obtained via $f_\omega$ applied to a predicted subsurface model.

Since the data loss is computed in the band-limited seismic data domain, the most common approach is the semi-supervised learning strategy, where the network is simultaneously guided by the misfits between observed measures and synthetic responses (i.e., data misfit) and between the predicted and the expected subsurface model parameters (i.e., model misfit). In this case, a weighted sum of the data misfit (i.e., the unsupervised term) and the model misfit (i.e., the supervised term) is used to compose the final cost function:

$$\mathcal{L} = \lambda_d \mathcal{L}_d + \lambda_m \mathcal{L}_m,$$

where $\lambda_d$ and $\lambda_m$ are hyper-parameters dictating scaling and relative importance of the two misfit terms. The data loss stabilizes the convergence of the training process and mitigates the overfitting problem. The model loss, on the other hand, provides essential high- and low-frequency information for the reconstruction of the subsurface model.

A number of physics-guided DL-based models have recently been proposed to solve geophysical inversion problems, such as velocity inversion [37, 38, 113, 34], impedance and amplitude-variation-with-offset (AVO) inversion [36, 35, 39], geosteering inversion [38] and porosity estimation [40].

A set of experiments were carried out in [34] to evaluate and compare fully data-driven, fully physics-driven and semi-supervised DL-based approaches to solve the seismic full wave inversion (FWI) problem. The trade-off between data misfit and model misfit is analyzed through a set of experiments based on a 2D synthetic velocity dataset. The experiments show that the physics-based objective function significantly improves the accuracy of the estimated velocity models.

In [28, 38, 114] a semi-supervised approach is devised by formulating the seismic inversion problem as a domain-transfer task. In these studies, the cycleGAN framework [115] is applied to map between the seismic amplitude domain and velocity models. The inversion system using the cycleGAN framework consists of two networks: one to learn a surrogate function of forward modeling $f_\omega : \mathbf{m} \in \mathbb{R}^N \rightarrow \mathbf{d} \in \mathbb{R}^P$, and the other one to learn an inverse mapping $f_\theta^{-1} : \mathbf{d} \in \mathbb{R}^P \rightarrow \mathbf{m} \in \mathbb{R}^N$. The cycle-consistent loss ensures that subsurface models obtained by transforming from the seismic amplitude to the velocity domain and back to the amplitude representation are consistent (i.e., $f_\omega$ and $f_\theta^{-1}$ are reverse to each other), which can be calculated both on labeled and unlabeled data.

Alternatively, a standard GAN [5] can be used to learn a generative prior from samples of the expected geological patterns, as represented, for example, by synthetically created subsurface models. The generative model defines a low-dimensional representation of the original high-dimensional space of the model parameters $G : \mathbf{z} \in \mathbb{R}^K \rightarrow \hat{\mathbf{m}} \in \mathbb{R}^N, K \ll N$. Once the generative model $G$ is trained, the estimate of a measured seismic data is obtained by searching in the latent space for a inverted subsurface model that best explains the observed seismic data [46, 47, 48]. This is achieved by solving the following optimization problem:

$$\hat{\mathbf{m}} = G(\hat{\mathbf{z}}), \tag{3-31}$$

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \mathbb{E}_{\mathbf{z}}[\mathcal{L}_{\mathrm{z}}], \tag{3-32}$$

$$\mathcal{L}_z = \|f_\omega(G(\mathbf{z})) - \mathbf{d}\|_2^2, \tag{3-33}$$

where the subsurface model estimate is given by $\hat{\mathbf{m}}$.

The parameterization of geological models by deep generative models can be particularly useful to address the non-uniqueness of inverse problems, as they can generate stochastic realizations of the underlying subsurface model that match the measured data. However, deep generative models are unstable and difficult to train. GANs have been notorious for their mode collapse, when only a few modes of data are generated, and the difficulty of reaching Nash equilibrium during training [116].

Conventional supervised (or semi-supervised) deep neural networks define a deterministic function from the input to the output from a given training dataset. However, the uncertainties inherent to seismic inversion problems make it crucial to quantify the uncertainty associated with the network estimates.

In [34], the authors propose the use of the Monte Carlo dropout method [43] to approximate a deep ensemble Bayesian model and quantitatively analyze the performance of a DNN in inverting velocity models from seismic reflection data. From the set of predicted velocity models, they compute the mean and the standard deviation statistics. With these statistics at hand, it is possible to identify high uncertainty areas related to illumination issues and susceptibility to missing or incomplete data.

A variety of CNN architectures have been applied in the seismic inversion methods referenced in this thesis, including FCN, U-Net and residual based feedforward architectures. In [36], a recurrent DL network is applied to capture the temporal dynamics of seismic traces in a physics-driven framework for elastic inversion. With few exceptions, most of these works apply 1D convolutions to process 2D and 3D datasets in a per-trace manner.

Despite many recent advances, there are still key challenges regarding the unavailability of labeled data (real or synthetically generated), and the uncertainty assessment of the predictions. As will be shown in the next chapter, we deal with these challenges by coupling geostatistical simulation to assess uncertainty about the subsurface predictions and to exploit the model parameters' space in a physics-driven and self-supervised manner.

# 4
# Deep Physics-Driven Stochastic Seismic Inversion

We propose a new self-supervised DL-based method to invert multiple fine-scale 3D subsurface models that fit equally well to the observed seismic reflection data. This methodology shares the principles of the Global Geostatistical Seismic Inversion [19].

We start by describing the proposed methodology in the acoustic domain to invert acoustic impedance models from full-stack seismic data. Next, we generalize our methodology for the elastic domain to invert P-wave velocity, S-wave velocity, and density models from pre-stack seismic data. We close by showing that the proposed methodology can simultaneously integrate pre-stack seismic data with petrophysical data to invert directly for reservoir properties, such as porosity.

## 4.1
## Acoustic Inversion

Acoustic inversion aims at predicting the parameters of the acoustic impedance (AI) model, $\hat{\mathbf{m}} \in \mathbb{Y} \subseteq \mathbb{R}^N$, from observed seismic data, $\mathbf{d} \in \mathbb{X} \subseteq \mathbb{R}^P$, of the form $\mathbf{d} = f(\mathbf{m}) + \eta$, where $\eta \in \mathbb{R}^P$ represents the noise vector and $f$ is the seismic forward model. Usually $P \ll N$ (i.e., $\mathbf{m}$ is a much denser grid), so we start by performing an up-sampling operation in $\mathbf{d}$ to match the dimensions of $\mathbf{m}$.

The architecture of the proposed inversion system consists of two components: $f_\theta^{-1}$, a CNN of the type encoder-decoder parameterized by weights and biases in a multidimensional space ($\theta \in \Theta \subseteq \mathbb{R}^K$), and $f_\omega$, a physics-based CNN with non-learnable weights. The network $f_\theta^{-1}$ is any encoder-decoder backbone. It is used as a surrogate to the inverse of the seismic forward model, i.e., $f_\theta^{-1} : \mathbb{X} \rightarrow \mathbb{Y}$. The network $f_\omega$ is used to implement the seismic forward model $f$, i.e., $f_\omega : \mathbb{Y} \rightarrow \mathbb{X}$.

The goal of $f_\theta^{-1}$ is to synthesize AI models that match observed data from a prior distribution on the model parameters, as represented by geologically consistent AI models generated with geostatistical simulation.

We introduce two workflows, one for training and one for inference, as explained next.

### 4.1.1
### Training the Inverse Network

We start by generating a set of $M$ AI models ($\mathcal{M} : \{\mathbf{m}'_j \in \mathbb{Y} \subseteq \mathbb{R}^N\}, j = 1, \cdots, M$) from the set of AI well-log data and imposing a variogram model with direct sequential simulation (DSS) [66].

These geostatistical models are used as input to $f_\theta^{-1}$, additionally to the observed seismic data, to account for both low and high frequency content that cannot be predicted from the limited bandwidth seismic data. Also, this set of geostatistical models works as a random noise in a high-dimensional latent space that introduces stochasticity to ensure the diversity between the network outcomes.

During training, $f_\theta^{-1}$ learns a non-linear mapping, such that:

$$\hat{\mathbf{m}}_j = f_\theta^{-1}(\mathbf{d}, \mathbf{m}'_j) \approx \mathbf{m}, \tag{4-1}$$

where $\hat{\mathbf{m}}_j$ is the estimated AI model, $\mathbf{d}$ is the observed seismic data, $\mathbf{m}'_j$ is a geostatistical model drawn from the set of previously simulated models ($\mathbf{m}'_j \sim p_{\mathcal{M}}$), and $\mathbf{m}$ represents the unknown ground truth AI model.

In such a setting, $f_\theta^{-1}$ assumes a role similar to that of the generator in a conditional GAN framework, in which the generative model learns a conditional mapping from the observed seismic data ($\mathbf{d}$) plus a random noise ($\mathbf{m}'_j \sim p_{\mathcal{M}}$) into an inverted AI model ($\hat{\mathbf{m}}_j$).

As in other physics-based learning approaches, the physical forward network ($f_\omega$) is applied to devise the gradients with respect to the misfit between the observed seismic data and the synthesized seismic data computed from the network predictions:

$$\mathcal{L}_{\text{D}} = \|(f_\omega(\hat{\mathbf{m}}_j) - \mathbf{d})\|_2^2, \tag{4-2}$$

A second objective function, namely the model reconstruction loss, is defined to encourage the network to be sensitive to the geostatistical model and to preserve its statistical properties (i.e., mean, variance, and spatial distribution pattern). The model reconstruction loss is defined as:

$$\mathcal{L}_M = \|\mathbf{\Gamma} \odot (\hat{\mathbf{m}}_j - \mathbf{m}'_j)\|_2^2, \tag{4-3}$$

where $\mathbf{\Gamma}$ is a matrix representing the confidence in the geostatistical model. The symbol $\odot$ is used to denote the element-wise multiplication between two multidimensional vectors.

The matrix $\mathbf{\Gamma}$ is utilized to assign confidence levels to the geostatistical models. The confidence level is lower for regions where the synthesized seismic responses from the geostatistical model present a poor match with the observed seismic data. For these regions, the update of the network parameters will assimilate more information from the seismic data rather than from the geostatistical model.

The matrix $\mathbf{\Gamma}$ is computed for each geostatistical model used for training ($\mathbf{m}'_j \sim p_{\mathcal{M}}$) by performing the following steps. First, by applying the physical forward network, we synthesize seismic data from the geostatistical model. Then, we compute the matrix $\mathbf{\Gamma}$ based on the residuals between the retrieved synthetic seismic data and the observed seismic data, as follows:

$$\mathbf{z} = \mathbf{d} - f_\omega(\mathbf{m}'_j), \tag{4-4}$$

$$\mathbf{\Gamma} = \Lambda(|\mathbf{z}|), \tag{4-5}$$

where $\Lambda$ is a function that maps the values of $\mathbf{z} \in \mathbb{Z} \subseteq \mathbb{R}^N$ into a vector $\lambda \in \mathbb{L} \subseteq \mathbb{R}^N, 0 \leq \lambda \leq 1$.

The function $\Lambda$ describes the process of decreasing the relative importance of the geostatistical model as the values of its corresponding seismic data misfit increase. It can assume many forms, such as the linear form (i.e., constant rate decay) and be in the shape of an exponential function (i.e., exponential rate decay).

The choice of $\Lambda$ is subjective and based on the quality of the information available. For example, in areas with a considerable number of wells and where the subsurface geology is well understood, one can increase the confidence level related to the prior information. On the other hand, for high-quality seismic data and relatively unexplored regions where few wells are available, the confidence level associated with the seismic data should be increased. For this reason, the choice of $\Lambda$ is adjusted depending on the application examples.

In this work, we propose a function $\Lambda$ based on the complementary Gauss error function. The complementary Gauss error function, often denoted by **erfc**, is a special type of sigmoid function, defined as:

$$\Lambda(z) = \operatorname{erfc}(z), \tag{4-6}$$

$$\operatorname{erfc}(z) = 1 - \operatorname{erf}(z), \tag{4-7}$$

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt \tag{4-8}$$

In statistics, for a random variable $Z$ that is normally distributed with mean zero and standard deviation $1/\sqrt{2}$, the Gauss error function, for $z \geq 0$, is interpreted as the probability that $Z$ falls in the range $[-z, z]$. If we assume the seismic data misfit normally distributed with mean zero, i.e., $Z \sim \mathcal{N}(Z; 0, \sigma^2)$, $|Z|$ follows a half-normal distribution parameterized by a scale parameter $\sigma > 0$, such that:

$$\mathbb{E}(|Z|) = \sigma\sqrt{\frac{2}{\pi}}, \tag{4-9}$$

$$\operatorname{Var}(|Z|) = \sigma^2\left(1 - \frac{2}{\pi}\right), \tag{4-10}$$

$$F(|Z|, \sigma) = \operatorname{erf}\left(\frac{|Z|}{\sigma^2\sqrt{2}}\right), \tag{4-11}$$

$$Q(q, \sigma) = \sigma\sqrt{2}\, \operatorname{erf}^{-1}(q), 0 \leq q \leq 1, \tag{4-12}$$

Where $\operatorname{erf}^{-1}(z)$ is the inverse error function, $F$ denotes the cumulative distribution function (CDF) and $Q$ the quantile function of the random variable $|Z|$.

By the choice of $\sigma$, one can control whether the results will assimilate more information from the seismic data or from the geostatistical models. The larger the scale parameter the more spread out the distribution of $|Z|$ and, consequently, the smaller the decay factor. Figure 4.1 shows an example of a half-normal distribution with $\sigma = 0.13$, where $\Lambda(z) = \operatorname{erfc}(z) \geq 0.7$ for $|z| \leq 0.05$.

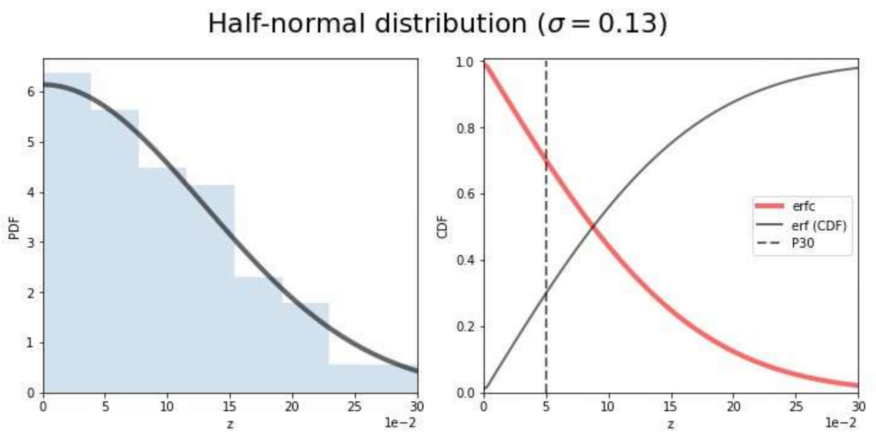


Figure 4.1: On the left, the probability density function and, on the right, the cumulative density function (CDF) of a half-normal distribution with $\sigma = 0.13$, where erfc($z$) $\geq 0.7$ for $z \leq 0.05$. The $\Lambda(z)$ function is defined by the complementary Gauss error function (erfc), defined as $1 -$ erf (or CDF).

One intuitive way to define the scale parameter is to choose a maximum acceptable error value $z^{\max}$, such that $\sigma$ is given as:

$$\sigma = \frac{z^{\max}}{\sqrt{2}\ \mathrm{erf}^{-1}(q)}, \tag{4-13}$$

Where $q$ is the lower bound limit of the values in the confidence matrix $\Gamma$ for which the data misfit is less than the maximum acceptable error ($|z| \leq z^{\max}$).

The parameters of the inverse network are updated to simultaneously minimize the data loss and the model reconstruction loss, such that:

$$\arg\min_{\theta} \mathbb{E}_{\mathbf{d},\mathbf{m}'_j \sim p_{\mathcal{M}}}[\mathcal{L}], \tag{4-14}$$

$$\mathcal{L} = \mathcal{L}_M + \mathcal{L}_D, \tag{4-15}$$

While the geostatistical models match exactly the AI well-log data after the up-scaling due to the use of geostatistical simulation, the proposed method does not impose any local constraint, which might lead to a mismatch between measured and predicted AI values. This might be an advantage as well, as we are accounting for any uncertainty or measurement errors that might be present in the log data.

However, one can force the network to be locally conditioned at the well locations (i.e., reproduce exactly the well-logs) by adding a new term to the objective function in equation 4-15:

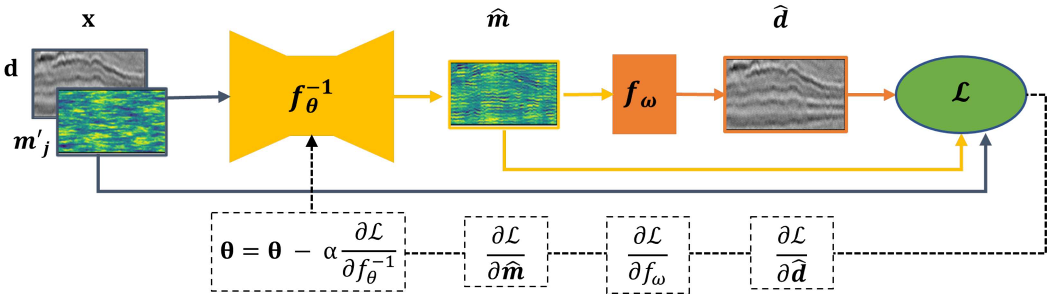
$$\mathcal{L}_W = \|\mathbf{M} \odot (\mathbf{m}'_j - \hat{\mathbf{m}}_j)\|_2^2, \tag{4-16}$$

$$\mathcal{L} = \mathcal{L}_M + \mathcal{L}_D + \mathcal{L}_W, \tag{4-17}$$

where $\mathbf{M}$ is a matrix with elements 1 for the well-log data locations and 0 for the rest (i.e., a mask).

The inverse network is trained on a randomly chosen subset from the set of geostatistical models to reduce the training time. This approximation is valid as long as all of the models generated with geostatistical simulation share the same spatial variability and well data information. The size of the training subset is data dependent and can be empirically defined by taking into account the generalization error. If the geological context contains different statistical and spatial characteristics, then one should train a $f_\theta^{-1}$ per set of realizations (i.e., per prior distribution).

A schematic representation of the training workflow is shown in Figure 4.2. The workflow consists of two main components: the inverse network ($f_\theta^{-1}$) with learnable parameters, and the physical forward network ($f_\omega$) with non-learnable parameters. The inverse network takes full-stack seismic data and a geostatistical AI model as inputs, and outputs the best estimate of the inverted AI model. Next, the forward network is used to synthesize seismograms from the inverted AI model. The error is computed between the synthesized and the input seismic data (Equation 4-2), and between the predicted and the input AI model (Equation 4-3). The parameters of the inverse network are adjusted by combining both losses as in Equation 4-15 using a gradient-descent optimization.



$\mathbf{x}$
$\mathbf{d}$
$\boldsymbol{m'}_j$
$f_\theta^{-1}$
$\hat{\boldsymbol{m}}$
$f_\omega$
$\hat{\boldsymbol{d}}$
$\mathcal{L}$
$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \frac{\partial \mathcal{L}}{\partial f_\theta^{-1}}$
$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{m}}}$
$\frac{\partial \mathcal{L}}{\partial f_\omega}$
$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{d}}}$

Figure 4.2: Schematic representation of the training workflow of the proposed acoustic inversion.

### 4.1.2
### Prediction of the AI Models

In the prediction stage, the trained inverse network is applied to invert one high-frequency AI model, $\hat{\mathbf{m}}_j$, for each pair $(\mathbf{d}, \mathbf{m}'_j)$ composed of the observed seismic data and the $j^{\text{th}}$ model in the set of geostatistical AI models $(\mathcal{M} : \{\mathbf{m}'_j \in \mathbb{Y} \subseteq \mathbb{R}^N\}, j = 1, \cdots, M)$.

The values estimated at each grid cell define a posterior distribution from which we can calculate the statistics (e.g., mean, variance, and percentiles) in order to quantify the uncertainty about the network predictions. As the cost of training is fixed and independent of the number of estimates, this method is especially appropriate for inverting many thousands of models to better explore the model's uncertainty space.

### 4.1.3
### Implementation

A pseudo-3D strategy, in which we assume 2D multi-slice representation of the 3D input and output data, is adopted. A 2D-CNN architecture was used for both inverse and forward networks. Recall that, in a 2D convolutional layer, a 2D convolutional kernel slides over 2D input data along two spatial dimensions.

The 3D-CNN architecture involves a much larger number of parameters than that of its 2D counterpart. Although there have been significant advances in developing resource-efficient 3D-CNN architectures, the current well-known resource-efficient CNN architectures are built with 2D convolutional kernels.

The inversion grid is composed of a set of hexahedral cells regularly aligned along layers and conformed to the reservoir stratigraphy. This structure, usually referred to as a stratigraphic grid, represents the skeleton of the reservoir framework. Due to the large number of model parameters to invert, usually of the order of magnitude of $10^8$ [17], the computation cost is a constraining factor that requires consideration in the practical use of 3D-CNN architectures for seismic inversion problems.

Typically, the seismic grid is laterally denser (i.e., with a higher resolution) than the stratigraphic grid, and samples have to be averaged laterally into one cell of the stratigraphic grid. Vertically, the seismic samples are sparser (i.e., with a lower resolution) and an operation to increase the seismic sample interval is needed.

The vertical slices are extracted from the input volumetric data in a given direction, such as the inline and crossline directions. However, depending on the complexity of the geological background, a multiple-direction strategy can be applied.

#### 4.1.3.1
#### Architecture of the Inverse Network

The network $f_\theta^{-1}$ maps an input vector, $\mathbf{x} \in \mathbb{X}, \mathbb{Y} \subseteq \mathbb{R}^{2\times H\times W}$, into an output vector, $\hat{\mathbf{m}} \in \mathbb{Y} \subseteq R^{1\times H\times W}$, where $H$ and $W$ correspond to the height and width of the 2D vertical slices extracted from input volumetric data, i.e., observed seismic data and geostatistical model. The first dimension of the vector represents the number of channels in the input and output data. The input vector has two channels: the first one is dedicated to the seismic data, while the latter is used for the geostatistical model.

The data samples are normalized to be between zero and one, by using the expression:

$$x' = (x - x_{min})/(x_{max} - x_{min}), \tag{4-18}$$

where $x_{min}$ and $x_{max}$ are the global minimum and maximum values of each input channel, respectively, considering the prior distribution (i.e., set of geostatistical models).

The network $f_\theta^{-1}$ follows an encoder–decoder architecture based on the residual learning framework [79]. The reason is that we want to benefit from residual blocks to avoid vanishing gradients and degradation of accuracy in training $f_\theta^{-1}$. Furthermore, the use of identity connections is often beneficial for image reconstruction tasks, in which the output image should share structure with the input image.

A convolutional layer (Conv) or a transposed convolutional layer (Deconv) subsequent to a batch normalization layer (BN) and a rectified linear unit (ReLU) activation layer is a popular choice because of its fast convergence. We use those as building blocks to implement the encoder-decoder architecture for the inverse network.

The features are extracted and encoded by a series of Conv-BN-ReLU blocks. All convolutional layers use a $3 \times 3$ kernel size. Then they are passed through the bottleneck, implemented by a residual layer, followed by another series of Deconv-BN-ReLU blocks that perform the decoding operation. Finally, the output is adjusted with a convolutional layer with a sigmoid activation function to the desired dimensions and to bound the values into the range of $[0, 1]$.

The residual blocks follow the architectural guidelines given by [79]. Each residual block has a $3 \times 3$ convolutional layer subsequent to a batch normalization layer and a rectified linear unit (ReLU) activation function. A second $3 \times 3$ convolution layer plus a batch normalization layer is stacked on top of the previous block. The skip connection bypasses both these layers and adds directly before the ReLU activation function. These residual blocks are repeated five times to compose the residual layer.

Table 4.1.3.1 summarizes the architecture of the inverse network given input data of size $H = 128 \times W = 550$.

| **Layer** | **Filter size, Stride** | **Output (C x H x W)** |
|---|---|---|
| Input | - | 2 x 128 x 550 |
| Conv, BN, ReLU | 3 x 3, 1 | 32 x 128 x 550 |
| Conv, BN, ReLU | 3 x 3, 2 | 64 x 64 x 275 |
| Conv, BN, ReLU | 3 x 3, 2 | 128 x 32 x 138 |
| 5 x Residual Blocks | | |
| Conv, BN, ReLU, Conv, BN | 3 x 3, 1 | 128 x 32 x 138 |
| Deconv, BN, ReLU | 3 x 3, 1 | 64 x 64 x 275 |
| Deconv, BN, ReLU | 3 x 3, 1 | 32 x 128 x 550 |
| Conv, BN, Sigmoid | 3 x 3, 1 | 1 x 128 x 550 |

Total # parameters: 1,664,547

### 4.1.3.2 Architecture of the Physical Forward Network

By assuming a normally incident wave, we restrict ourselves to primary waves and post-stack data. In this case, the physical forward network $f_\omega$ performs the simulation of synthetic seismic data based on the simple convolution-based forward modeling [117].

The network $f_\omega$ receives as input a vector of the form $\mathbf{m} \in \mathbb{Y} \subseteq \mathbb{R}^{1 \times H \times W}$ in the AI domain and outputs a vector $\mathbf{d} \in \mathbb{X} \subseteq \mathbb{R}^{1 \times H \times W}$ in the amplitude domain.

The reflection coefficient log is obtained as a function of the contrast of the acoustic impedance values, from which the seismic trace is calculated by the convolution of the source wavelet. The forward operator is implemented as a convolutional model of the form:

$$\mathbf{d} = \omega * \mathbf{r}, \tag{4-19a}$$

$$\mathbf{r}[i,j] = R(\mathbf{m}[i,j]) \tag{4-19b}$$

$$R(\mathbf{m}[i,j]) = \frac{\mathbf{m}[i+\Delta i,j] - \mathbf{m}[i,j]}{\mathbf{m}[i+\Delta i,j] + \mathbf{m}[i,j]}, \tag{4-19c}$$

where $\omega$ is the source wavelet and $\mathbf{r}$ represents the reflection coefficients calculated from the acoustic impedance model through the function $R$, for $i = 1, \cdots, H$ and $j = 1, \cdots, W$.

Equation 4-19a can be modeled as a two-layer CNN, including an input layer ($\mathbf{m}$) and output layer ($\mathbf{d}$). In the input layer, the reflectivity coefficients are obtained from the input data by performing a series of differentiable operations as follows. First, the input AI model is scaled back to the AI global minimum and maximum values. Then, the re-scaled AI vector is used to calculate the normal incidence reflectivity coefficients (Equation 2-7). In the convolutional layer, $\omega$ is convolved with the reflectivity coefficients along the vertical profile (i.e., the height dimension). Finally, the retrieved synthetic seismic data is scaled back to the interval $[0, 1]$ in the output layer of the network.

### 4.1.4
### Workflow

Figure 4.3 shows the main outline of the **Deep Physics-Driven Stochastic Acoustic Inversion**. A set of $M$ acoustic impedance models is simulated from the set of AI well-log data and imposing a variogram model with direct sequential simulation (DSS). The training procedure is performed on a subset of the set of geostatistical models, along with the observed seismic data. Each $j^{th}$ prediction in the set of inverted AI models is obtained by taking the $j^{th}$ geostatistical AI model, along with the observed seismic data, as input to inverse network.

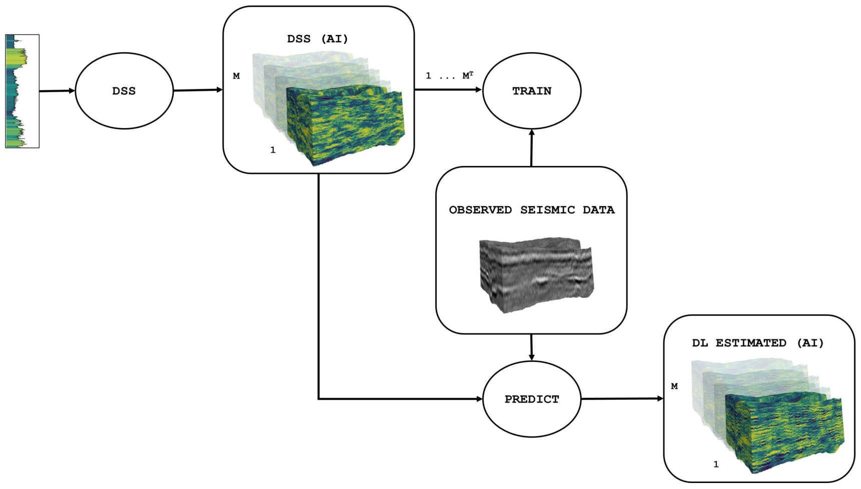


Figure 4.3: Main outline of the **Deep Physics-Driven Stochastic Acoustic Inversion**.

### 4.1.5
### Pseudo-Algorithm

The proposed Deep Physics-Driven Stochastic Acoustic Inversion method can be summarized by the following pseudo-algorithm:

**Algorithm 2:** Deep Physics-Driven Stochastic Acoustic Inversion

```
Input: Variogram model $\gamma$, AI well-log data $\mathbf{w}$, post-stack seismic
       reflection data $\mathbf{d}$
Output: Set of inverted AI models $\hat{\mathcal{M}} : \{\hat{\mathbf{m}}_j\}, j = 1, \cdots, M$
1 Generate the set of AI models, $\mathcal{M} : \{\mathbf{m}_j\}, j = 1, \cdots, M$, with
   geostatistical simulation using $\gamma$ and $\mathbf{w}$ as input data.
2 Concatenate $\mathbf{d}$ with each model in $\mathcal{M}$ to create
   $\mathcal{X} : \{(\mathbf{m}_j, \mathbf{d})\}, j = 1, \cdots, M$.
3 Randomly select a subset $\mathcal{X}^T \in \mathcal{X}$ for training.
4 Train $f_\theta^{-1}$ on $\mathcal{X}^T$ using the mini batch gradient descent algorithm.
5 Apply the trained $f_\theta^{-1}$ using each pair in $\mathcal{X}$ as input to obtain
   $\hat{\mathcal{M}} : \{\hat{\mathbf{m}}_j\}, j = 1, \cdots, M$.
6 Return the set of inverted AI models $\hat{\mathcal{M}}$ computed in the previous step.
```

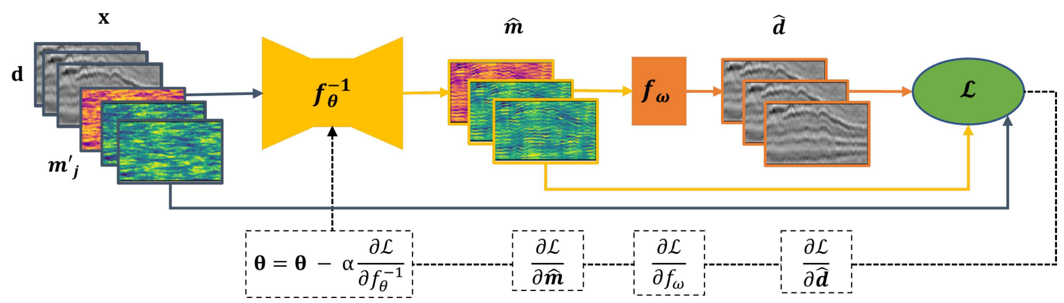
## 4.2
## Amplitude-Versus-Angle Inversion

Amplitude variation with the angle of incidence (AVA) analysis is a technique of reservoir characterization which involves the use of pre-stack seismic data to infer density, compressional wave (P-wave) and shear wave (S-wave) velocity models. In this section, we show that the proposed methodology to invert acoustic impedance from the normal angle of incidence can be generalized for the pre-stack domain.

The main outline of the **Deep Physics-Driven Stochastic AVA Inversion** is similar to the algorithm proposed for acoustic inversion. When the observed seismic data is sorted by angle gathers (i.e., multi-angle), the model parameters may represent subsurface elastic properties like P- and S-wave velocities ($V_P$ and $V_S$) and density ($\rho$). Each triplet of inverted $\rho$, $V_P$ and $V_S$ models is used to compute its corresponding elastic response (i.e., angle dependent synthetic reflection seismic data) by convolving the source wavelet with the reflection coefficients computed based on the Zoeppritz equations (Equation 2-6).

The geostatistical simulation process comprises the simulation of elastic models following a cascading approach. We start by simulating a set of $M$ elastic models of density and P- and S-wave velocities using direct sequential simulation (DSS) and co-simulation (Co-DSS). As usual, a spatial continuity pattern is imposed in the sequential simulations for each elastic property. Density is simulated with DSS using the existing well-log data as experimental data. Then, $V_P$ is co-simulated using the retrieved Density model as auxiliary variable. Finally, $V_S$ is co-simulated using the previous simulated $V_P$ model as auxiliary variable.

The workflows for training and prediction follow the same guidelines as the proposed acoustic version of this algorithm. The training workflow (Figure 4.4) consists of two main parts: the inverse network ($f_\theta^{-1}$) with learnable parameters, and the forward model ($f_\omega$) with non-learnable parameters. The inverse network takes as inputs partial angle stacks, alongside the geostatistical elastic models, and outputs the best estimate of the corresponding elastic models. Then, the forward network is used to calculate synthetic partial angle stacks from the estimated elastic models. The error is computed between the synthesized and the input seismic data (4-2), and between the predicted and the input elastic models (4-3). The parameters of the inverse network are adjusted by combining both losses as in equation 4-15 using a gradient-descent optimization.



$\mathbf{x}$
$\mathbf{d}$
$\boldsymbol{m'}_j$
$\boldsymbol{f_\theta^{-1}}$
$\widehat{\boldsymbol{m}}$
$\boldsymbol{f_\omega}$
$\widehat{\boldsymbol{d}}$
$\boldsymbol{\mathcal{L}}$
$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \frac{\partial \mathcal{L}}{\partial f_\theta^{-1}}$
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{m}}}$
$\frac{\partial \mathcal{L}}{\partial f_\omega}$
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{d}}}$

Figure 4.4: Schematic representation of the training workflow of the proposed AVA inversion.

The architectures of the networks for the inverse and forward mappings are adapted to work with the multiple channels that comprise the elastic models, one channel for each elastic property, and one channel for each partially stacked angle. With the exception of the input and output layers, the AVA inverse network follows the same architecture as the acoustic inverse network.

In this new formulation, the inverse network maps an input vector, $\mathbf{x} \in \mathbb{X}, \mathbb{Y} \subseteq \mathbb{R}^{(K+3)\times H\times W}$, into an output vector, $\hat{\mathbf{m}} \in \mathbb{Y} \subseteq \mathbb{R}^{3\times H\times W}$, where

$K$ is the number of partially stacked angles of the seismic data, and $H$ and $W$ correspond to the height and width of the 2D vertical slices extracted from input volumetric data.

The 3-term Shuey's approximation (Equation 2-8) [53] is used to calculate the synthetic seismic gather with offset angles from the inverted elastic models. The physical forward network receives as input a vector of the form $\mathbf{m} \in \mathbb{Y} \subseteq \mathbb{R}^{3\times H\times W}$ in the elastic domain and outputs a vector $\mathbf{d} \in \mathbb{X} \subseteq \mathbb{R}^{K\times H\times W}$ in the amplitude domain. The convolutional layer of the physical forward network operates with a kernel composed of $K$ channels, one for each partially stacked angle.

A schematic representation of the training workflow of the **Deep Physics-Driven Stochastic AVA Inversion** is shown in Figure 4.5. A set of $M$ triplets of elastic models is simulated from the set of density, P-wave and S-wave velocities well-log data and imposing a variogram model with direct sequential simulation (DSS). The training procedure is performed on a subset of the set of geostatistical models, alongside the observed seismic data. Each $j^{th}$ prediction in the set of inverted models is obtained by taking as input to inverse network the $j^{th}$ triplet of geostatistical elastic models, alongside the observed seismic data.

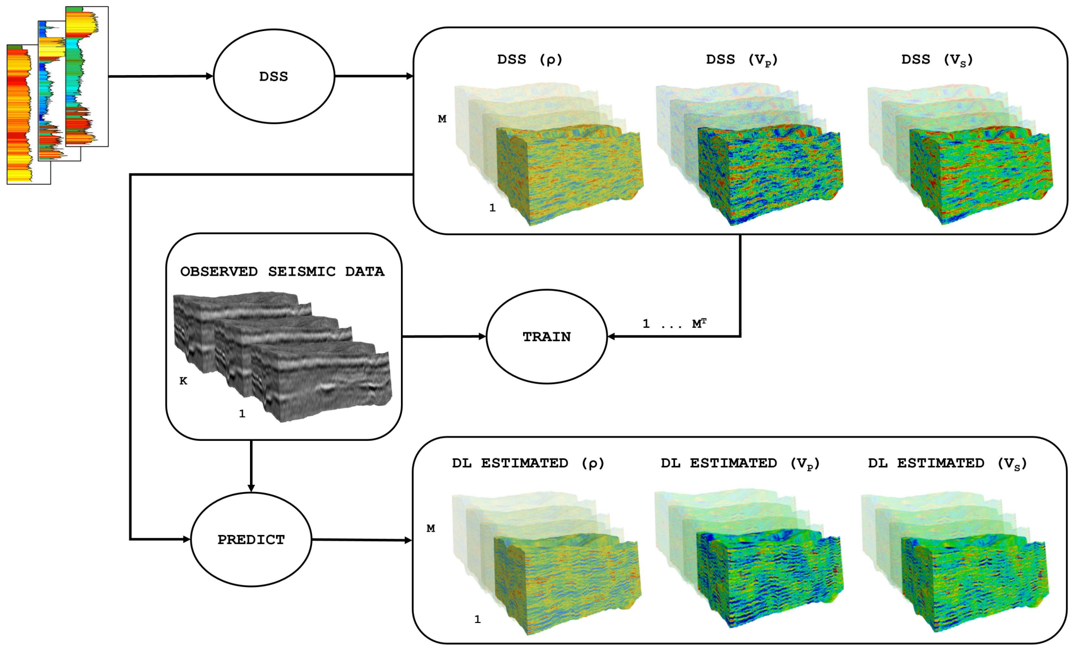


Figure 4.5: Main outline of the **Deep Physics-Driven Stochastic AVA Inversion**.

## 4.3 Petrophysical Inversion

While the Zoeppritz equations and the seismic convolutional model link the elastic properties to the seismic reflection data, the elastic and rock property domains are related through a rock physics model [118]. A rock physics model is a set of mathematical equations describing the elastic response of a rock given a set of petrophysical properties, such as fluid saturation, porosity, and volume of minerals.

The goal of petrophysical inversion is the joint prediction of both elastic and rock properties directly from the observed seismic reflection data. Unlike the traditional sequential workflows, where elastic inversion is followed by a rock physics inversion, the direct petrophysical inversion workflows guarantee consistency between the elastic and reservoir properties.

This problem can be mathematically summarized as follows:

$$\mathbf{v} = p(\mathbf{m}) \tag{4-20}$$

$$\mathbf{d} = f(\mathbf{v}) + \eta \tag{4-21}$$

where $\mathbf{v} \in \mathbb{Y} \subseteq \mathbb{R}^N$, is the elastic model, $\mathbf{m} \in \mathbb{P} \subseteq \mathbb{R}^N$ is the rock property model, $\mathbf{d} \in X \subseteq \mathbb{R}^P$ is the observed seismic data, $p$ is the rock physics model, $f$ is the seismic forward model and $\eta \in \mathbb{R}^P$ represents the noise vector.

In this study, we propose a DL-based workflow to directly estimate reservoir porosity models from post-stack seismic reflection data (Figure 4.6). A set of $M$ porosity models is simulated from the set of porosity well-log data and imposing a variogram model with direct sequential simulation (DSS). The training procedure is performed on a subset of the set of geostatistical models, alongside the observed seismic data. Each $j^{th}$ prediction in the set of inverted porosity model is obtained by taking the $j^{th}$ geostatistical porosity model, alongside the observed seismic data, as input to inverse network.

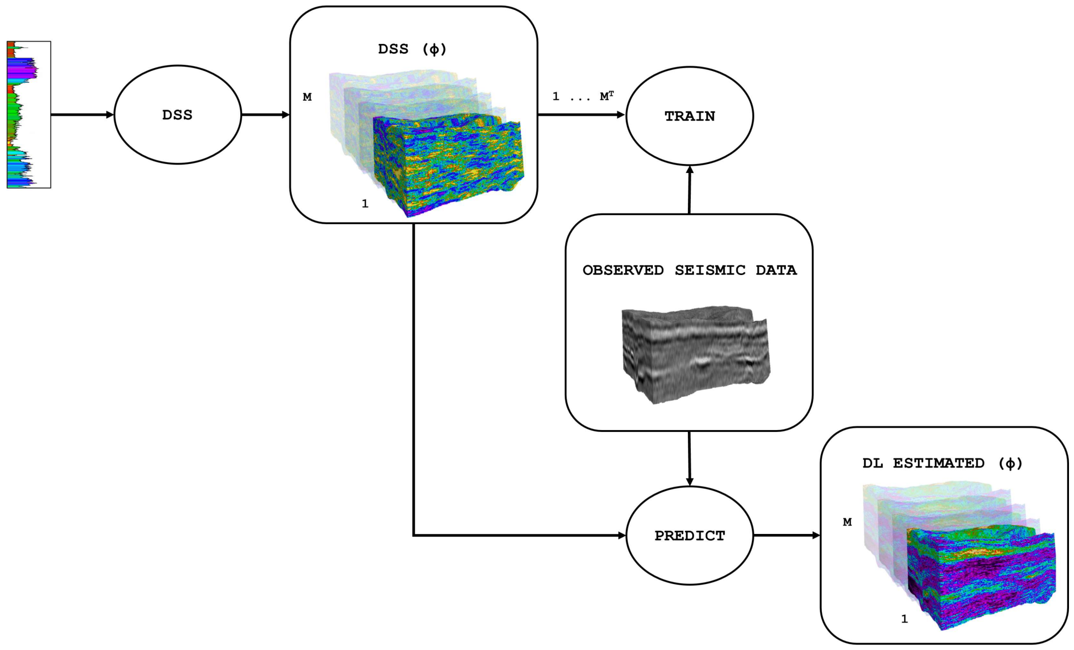


Figure 4.6: Main outline of the **Physics-Driven Stochastic Petrophysical Inversion**.

The space of the model parameters lies in the petrophysical domain ($\mathbb{P}$), as represented by the porosity models. In the proposed petrophysical inversion, the rock physics model ($p$) for the simulation of the rocks' elastic responses is a simplification of the full physics, defined as a linear relation of the form:

$$\mathbf{m} = a \times \mathbf{v} + b$$

where $a$ and $b$ are parameters ($a, b \in \mathbb{R}$) empirically defined from well-log data.

The inputs to the inverse network are the observed seismic data and the geostatistical porosity models generated with direct sequential simulation (DSS) [66]. The networks' architectures of the inverse and forward operators are the same as the networks used in the proposed acoustic inversion method.

The inverse network maps an input vector, $\mathbf{x} \in \mathbb{X}, \mathbb{P} \subseteq \mathbb{R}^{2\times H\times W}$, into an output vector, $\hat{\mathbf{m}} \in \mathbb{P} \subseteq \mathbb{R}^{1\times H\times W}$, where $H$ and $W$ correspond to the height and width of the 2D vertical slices extracted from the input volumetric data.

We start by generating $M$ porosity models ($\mathcal{M} : \{\mathbf{m}'_j\}, j = 1, \cdots, M$) from the set of porosity well-log data and imposing a variogram model with direct sequential simulation (DSS) [66].

The rock physics model is incorporated into our DL-based inversion system through the implementation of a series of differentiable operations, namely $p_\psi : \mathbb{P} \to \mathbb{Y}$, with $\psi$ consisting of $a$ and $b$. The data loss misfit is reformulated to take the petrophysical and seismic forward models into account:

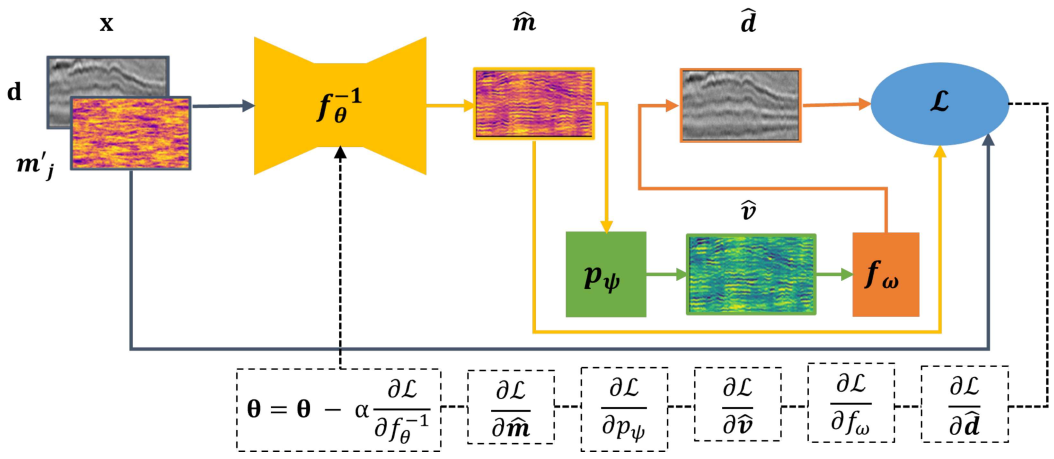
$$\hat{\mathbf{m}}_j = f_\theta^{-1}(\mathbf{d}, \mathbf{m}'_j) \tag{4-22}$$

$$\hat{\mathbf{v}}_j = p_\psi(\hat{\mathbf{m}}_j) \tag{4-23}$$

$$\mathcal{L}_{\mathrm{D}} = \|(f_\omega(\hat{\mathbf{v}}_j) - \mathbf{d})\|_2^2 \tag{4-24}$$

where $\|.\|_2^2$ is the squared L2-norm, $\mathbf{d}$ is the observed seismic data and $\mathbf{m}'_j$ is a geostatistical porosity model.

The training workflow (Figure 4.7) follows the same idea as the proposed acoustic inversion. The inverse network takes full-stack seismic data and a geostatistical porosity model as inputs, and outputs the best estimate of the inverted porosity model. We start by estimating the inverted porosity model ($\hat{\mathbf{m}}$) using the inverse network $f_\theta^{-1}$. Then, we obtain the corresponding AI model ($\hat{\mathbf{v}}$) from the inverted porosity model by applying the rock physics network ($p_\psi$). In the next step, the physical forward network ($f_\omega$) is applied to the retrieved AI model ($\hat{\mathbf{v}}$) to obtain the synthetic seismic data ($\hat{\mathbf{d}}$). The parameters of the inverse network are updated to simultaneously reduce the model reconstruction loss (Equation 4-3) and the seismic data loss (Equation 4-24).



$\mathbf{x}$
$\mathbf{d}$
$\boldsymbol{m}'_j$
$f_\theta^{-1}$
$\widehat{\boldsymbol{m}}$
$\widehat{\boldsymbol{d}}$
$\mathcal{L}$
$\widehat{\boldsymbol{v}}$
$p_\psi$
$f_\omega$
$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \frac{\partial \mathcal{L}}{\partial f_\theta^{-1}}$
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{m}}}$
$\frac{\partial \mathcal{L}}{\partial p_\psi}$
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{v}}}$
$\frac{\partial \mathcal{L}}{\partial f_\omega}$
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{d}}}$

Figure 4.7: Schematic representation of the training workflow of the proposed petrophysical inversion.

# 5
# Application Examples

The proposed acoustic inversion is first applied to a 1D synthetic example to illustrate and demonstrate the robustness of the method on synthetic data. Then, the proposed acoustic, AVA and petrophysical inversions are carried out in a 3D real application example. Both datasets are related to the same Albian post-salt carbonate sequence of Bacia de Santos in the Brazilian offshore.

We ran the training and prediction workflows on eight NVIDIA Tesla V100 GPUs with 32GB of memory. The implementation of the method was based on PyTorch [1], a popular open-source machine learning library for Python.

As in other applications of deep learning, the results shown herein were obtained after running several experiments to tune the hyper-parameters of the network. Adaptive Moment Estimation (Adam) optimizer [69] was used with learning rate of 0.001, $\beta_1$ 0.9 and $\beta_2$ 0.999.

We want to make sure we match the observed seismic data while preserving the statistical properties of the geostatistical models. We evaluate whether the results reproduce the prior variability and spatial continuity by comparing the histograms and variograms of inputs and outputs. The absolute value of the model reconstruction loss along with the absolute value of the seismic data residuals are evaluated as well. In addition, two quantitative metrics, the structural similarity index measure (SSIM) [101] and the correlation coefficient (CC) between inputs and outputs are adopted to provide a general idea of the prediction accuracy.

The results obtained with the proposed deep learning method will be indicated as **DL**, while the geostatistical models generated with direct sequential simulation and co-simulation will be referred to as **DSS**.

## 5.1
## 1D Synthetic Case Application

We start with a one-dimensional synthetic example built from a real AI well-log to illustrate and validate the proposed methodology. The synthetic seismogram was obtained from the true log using the wavelet extracted from the real seismic. The real AI well-log represents the true but unknown data-generating distribution. The vertical variogram model was fitted to the experimental variogram computed from the true log.

The inversion grid is defined by 128 vertical samples. We carried out the **Deep Physics-Driven Acoustic Inversion** on this synthetic dataset, considering an input data size of $H = 128 \times W = 1$. The total number of parameters of the inverse network is approximately 550 thousand. An ensemble of geostatistical AI logs composed of 1000 realizations was generated with direct sequential simulation [66] using six samples of the true log as experimental data.

From the ensemble of geostatistical AI logs, we selected a subset of 32 examples to illustrate the input and output data to our DL-based inverse system (Figure 5.1). In Figure 5.1, the first track **(a)** shows the ensemble of geostatistical AI logs (gray dashed lines) along with the experimental data (black dots) and the true AI log (black line). In track **(b)**, the synthetic seismograms retrieved from the geostatistical AI logs are shown (gray dashed lines) along with the seismic trace obtained from the true AI log (black line). As expected, the synthetic seismograms retrieved from the geostatistical AI logs do not match the target seismic trace.

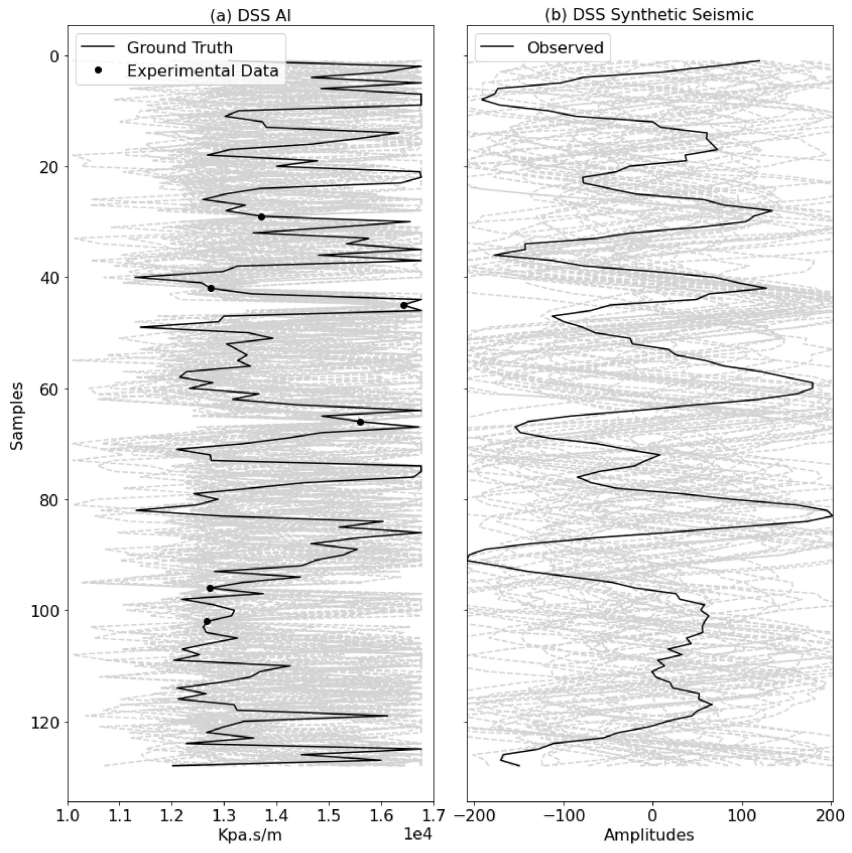


Figure 5.1: **(a)** Thirty-two geostatistical AI logs (dashed gray lines), along with the experimental data (black dots) and the real AI well-log (black line). **(b)** The synthetic seismograms (dashed gray lines) retrieved from the geostatistical AI logs and the target seismic trace (black line).

We first performed several experiments to evaluate the ability of the network to generalize on unseen data. Deciding the sizes for data set division in train and test sets is very dependent on the data available. As with other network hyper-parameters, this is done through trial-and-error experimentation.

We randomly divide the ensemble of geostatistical AI logs into non-overlapping train and test subsets of different lengths, within the range [1%, 100%] with 10% increment. Figure 5.2 shows the absolute mean error (MAE) between the input and output models, and between the observed and synthetically simulated seismic data on the train and test sets monitored at every epoch, for 500 epochs, and split ratios of 1%, 10%, and 50%.

The inverse network does not fit the test dataset with a 1% split ratio. A better fit of the test subset is obtained with a 10% split ratio. We find that with 50% of the dataset (i.e., 500 geostatistical AI logs) in the train subset, there is an overlap between the train and test curves, which indicates a good fit to the test subset. Once trained, $f_\theta^{-1}$ can be applied to the full ensemble of geostatistical AI logs or to any other unseen AI log within the same distribution as that on the train set.

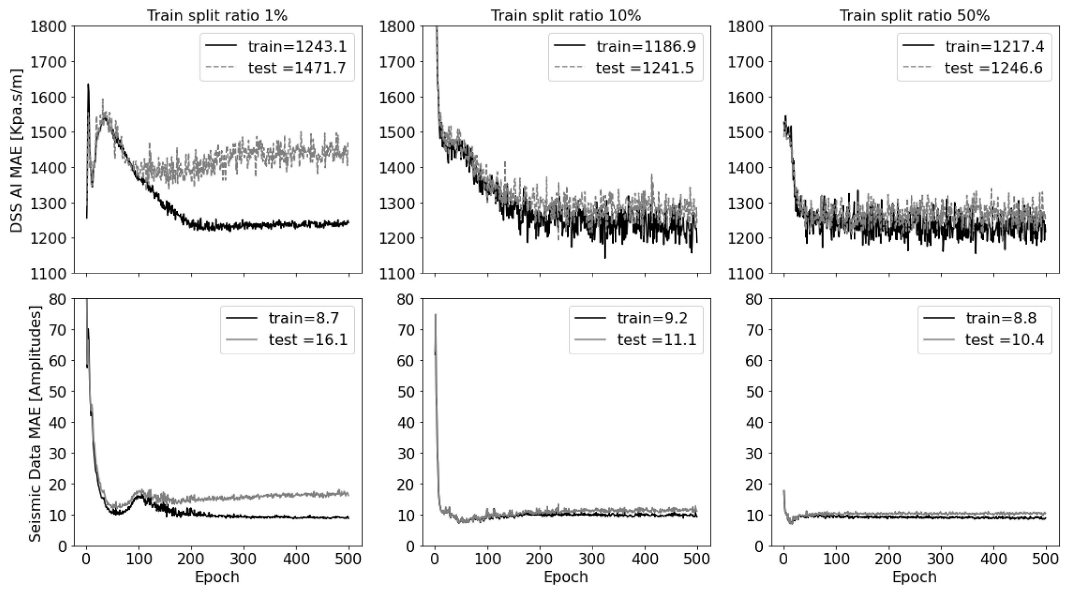


Figure 5.2: Progress of the mean absolute error (MAE) between the input and output models (first row), and between the observed and synthetically simulated seismic data (second row) on the train and test sets at every epoch, for 500 epochs, for the split ratios of 1%, 10%, and 50%. The split ratios correspond to the percentage of the ensemble of geostatistical AI logs used to train the neural network. The black curves are the train losses, and the test losses are shown in gray dashed lines.

We further investigate the influence of the confidence matrix ($\boldsymbol{\Gamma}$) on the performance of the network $f_\theta^{-1}$. In this experiment, the progress of the losses on the train set was monitored considering different values for the scale parameter of the half-normal distribution used as reference to define the confidence in the geostatistical models. Figure 5.3 shows different types of functions to compute the matrix $\Gamma$, which are used in this experiment. We also tested another scenario with equally distributed weights for seismic data and geostatistical models. For all the experiments, we trained the inverse network with a batch of size 32 for 300 epochs and 50% of the ensemble of geostatistical AI logs in the train subset.

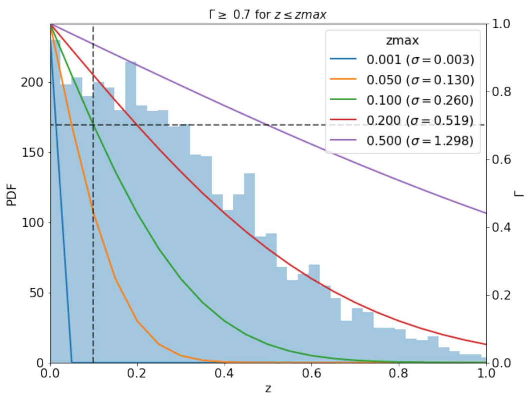


Figure 5.3: Different types of functions to compute the matrix $\Gamma$. Each line represents the erfc function considering the scale parameter $(\sigma)$ computed based on a given maximum acceptable error value $(z^{\max})$ between the observed seismic and the simulated seismic from the geostatistical model. The histogram of the $z$ values computed over the ensemble of geostatistical AI logs is shown in the background.

Figure 5.4 summarizes the set of experiments performed. The absolute mean error (MAE) between the input and output models (first row), and between the observed and synthetically simulated seismic data on the train set are shown at every epoch, for 300 epochs. The graph shows how the matrix $\Gamma$ controls whether the results assimilate more information from the seismic data or from the geostatistical models. In extreme cases, with $z^{\max} = 0.001$ $(\sigma = 0.003)$ and $\Gamma = 1$, the inverse network is only able to match either the seismic data or the DSS model.

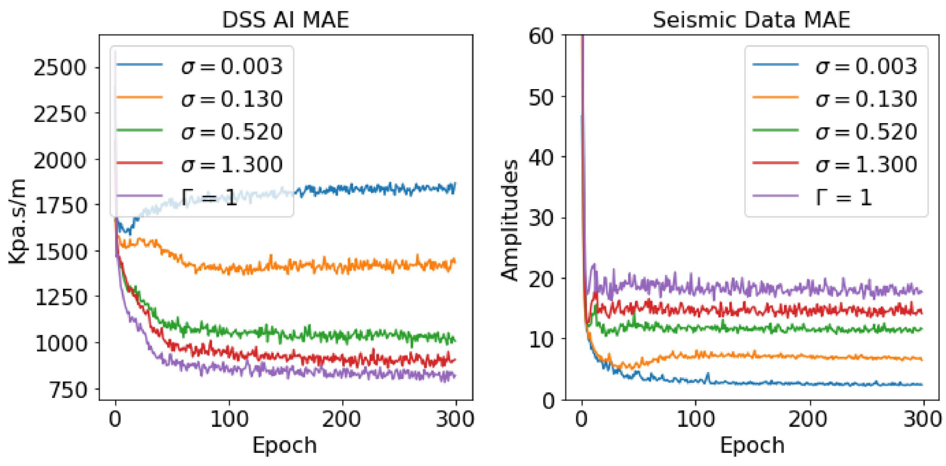


Figure 5.4: Progress of the absolute mean value errors of the AI model and of the seismic data on the train set at every epoch, for 300 epochs. Each curve represents a different scale parameter $(\sigma)$ used to define the confidence level in the geostatistical models.

Figures 5.5-a, 5.6-a and 5.7-a present the inverted AI logs when the observed seismic trace and the geostatistical AI logs in Figure 5.1-a are used as inputs to the trained $f_\theta^{-1}$. The synthetic seismograms retrieved from the inverted AI logs and the observed seismic trace are shown in Figures 5.5-b, 5.6-b and 5.7-b.

Figure 5.5 shows the results when the confidence in the geostatistical model is low. In this case, there is no variability among the network outcomes. The network ignores the geostatistical AI log and produces similar outputs that represent the best fit to the observed seismic data. The score metrics related to the seismic data are extremely high, while the metrics related to the reconstruction of the geostatistical AI log are very poor.

On the other hand, when the confidence in the geostatistical model is higher, the network learns to preserve information from the geostatistical AI log (Figures 5.6 and 5.7). However, this may come at the cost of increased mismatch between observed and synthetic seismograms (Figure 5.7).

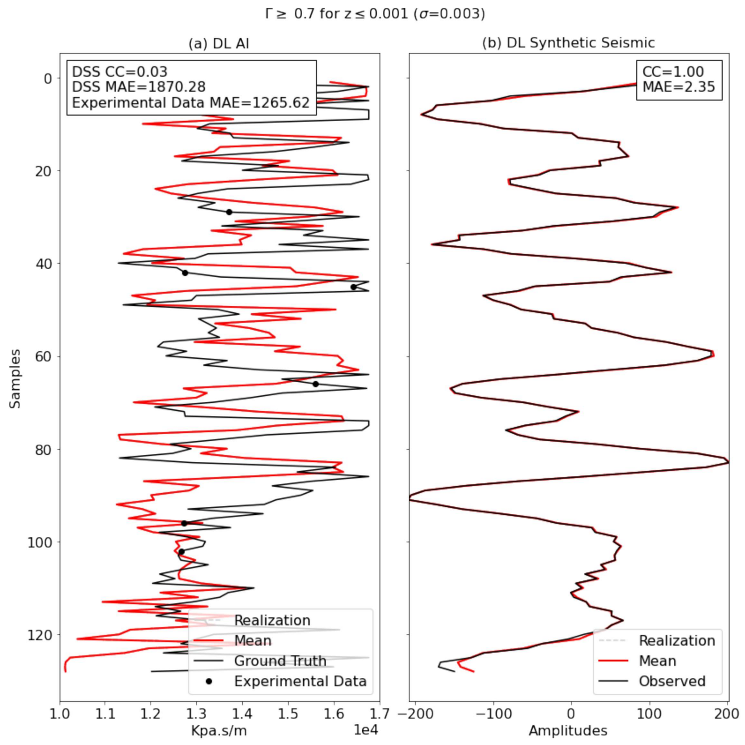


Figure 5.5: **(a)** Thirty-two predicted AI logs (dashed gray lines), along with the experimental data (black dots), the real AI well-log (black line), and the mean curve calculated over the set of predicted AI logs (red line). In this case, all the predicted AI logs coincide with each other. **(b)** The synthetic seismograms retrieved from the predicted AI logs (dashed gray lines) the target seismic trace (black line), and the mean curve calculated over the ensemble of synthetic seismograms.

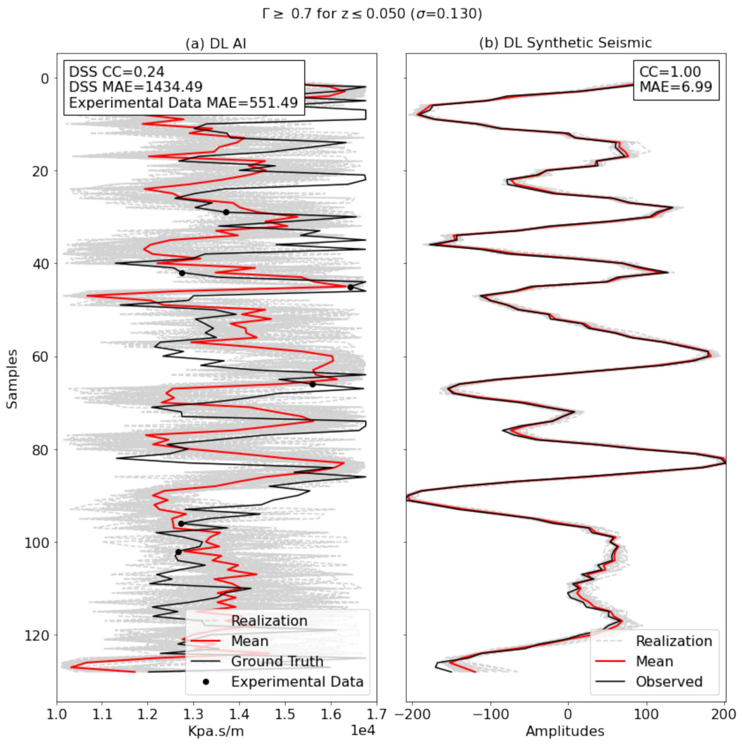


Figure 5.6: **(a)** Thirty-two predicted AI logs (dashed gray lines), along with the experimental data (black dots), the real AI well-log (black line), and the mean curve calculated over the set of predicted AI logs (red line). **(b)** The synthetic seismograms retrieved from the predicted AI logs (dashed gray lines), the target seismic trace (black line), and the mean curve calculated over the ensemble of synthetic seismograms.

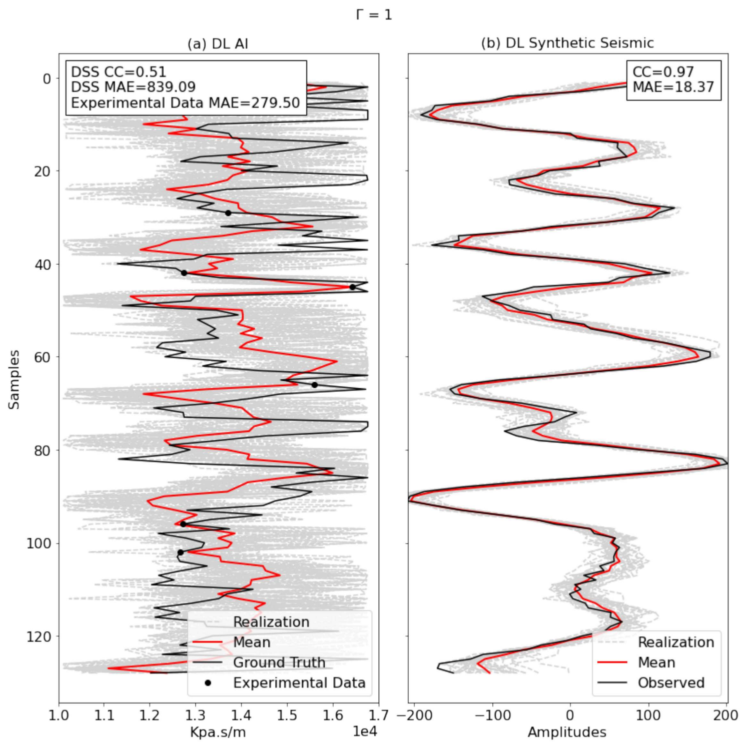


Figure 5.7: **(a)** Thirty-two predicted AI logs (dashed gray lines), along with the experimental data (black dots), the real AI well-log (black line), and the mean curve calculated over the set of predicted AI logs (red line). **(b)** The synthetic seismograms retrieved from the predicted AI logs (dashed gray lines) the target seismic trace (black line), and the mean curve calculated over the ensemble of synthetic seismograms.

## 5.2
## 3D Real Case Application

We applied the proposed methodology to a real case study of a Brazilian carbonate reservoir. The area is located approximately 175 km offshore, in southern Santos Basin. The seismic volume has an area of approximately 13 km in the North-South direction and 7,5 km in the East-West direction. The thickness of the target interval is about 200 meters.

Figure 5.8 shows the base map of the study area and the relative locations of the wells inside the seismic volume. The real dataset consists of well-logs from four different locations. The four wells have AI, $V_P$, $V_S$, density, and porosity logs. The inversion grid is defined by 315 (inline) x 550 (crossline) x 128 (time) cells. The vertical slices were extracted from the input volumetric data in the inline direction. Well-logs are acquired at a much higher vertical resolution than the typical inversion grid size, which requires up-scaling the well logs to the inversion grid scale.

Three partial angle stack seismic volumes with mean incident angles of 5° (near), 15° (mid) and 25° (far) were considered (Figure 5.9). The near seismic data was assumed to be zero offset in the acoustic and porosity inversion workflows. The wavelets were extracted following a conventional seismic-to-well tie procedure. The detailed description of this step is beyond the scope of this study.

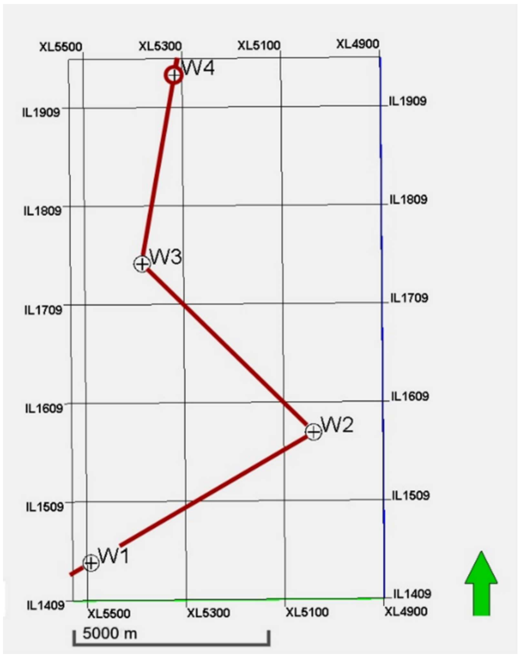


Figure 5.8: Base map of the study area showing the relative well location, the seismic bounding box and the vertical well section used to illustrate the results obtained (red line).

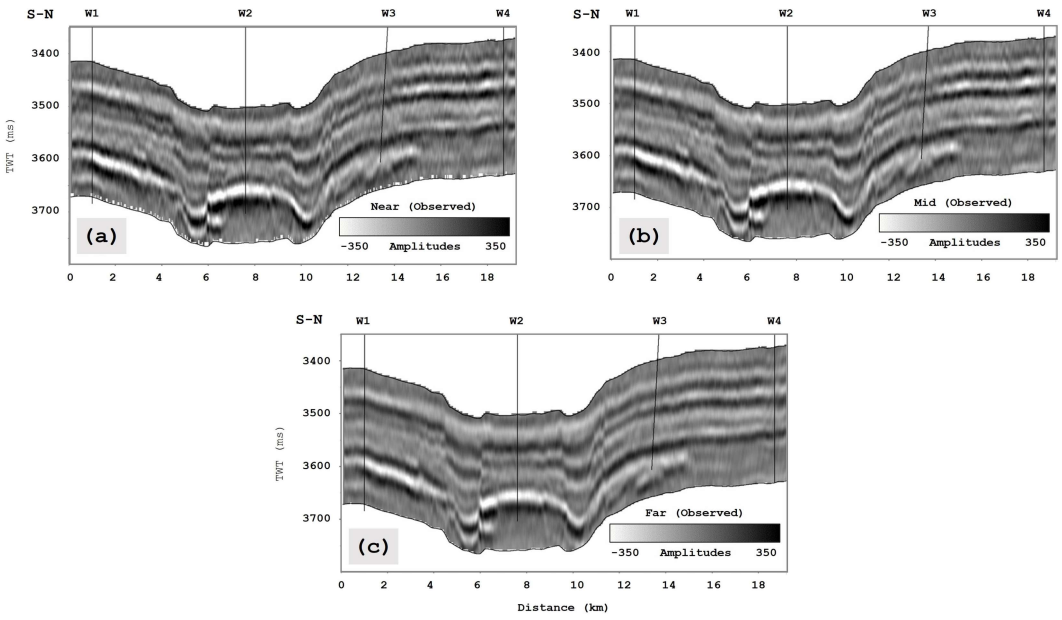


Figure 5.9: Vertical well section extracted from observed seismic data: **(a)** partial stacks of near offsets ($0° - 10°$ range angle); **(b)** partial stacks of mid offsets ($10° - 20°$ range angle); **(c)** partial stacks of far offsets ($20° - 30°$ range angle).

The spatial continuity patterns imposed in the DSS and Co-DSS methods were modeled using 3D exponential variograms. The vertical direction was modeled based on the available well-log data. Given the scarcity of data in the horizontal directions, we modeled these spatial correlations using seismic amplitudes. The horizontal variograms were computed considering a lag size (distances between pairs) of 200 meters and 20 lags, while the variogram in the vertical direction was computed with 30 lags of 2 milliseconds. We previously generated 100 models for each subsurface property (i.e., AI, $V_P$, $V_S$, density, and porosity) using DSS and Co-DSS conditioned on the existing well-log data.

We performed trial-and-error experimentation to tune the hyper-parameters of the network as well as to define the size of the train subset by monitoring the loss progress on the train and test sets. When trained on a train set composed of four geostatistical models (i.e., $4 \times 315 = 1260$ 2D vertical slices), the network was able to fit the train and test sets equally well. This indicates that the model does not over-fit the training data and is able to produce accurate outcomes for previously unseen geostatistical realizations. The number of epochs was empirically defined as well.

Using this dataset, we carried out the proposed inversion methods to invert 3D high-resolution AI, $V_P$, $V_S$, density, and porosity models from seismic reflection data, as described next. For each inversion performed, we trained the inverse network with a batch size of 32 for 350 epochs. We observed that after 300 epochs no relevant improvement is obtained for the objective function.

### 5.2.1 Acoustic Inversion

We first present the results from the proposed **Deep Physics-Driven Stochastic Acoustic Inversion**. We trained the inverse network in approximately 60 minutes using the computing infrastructure described above.

We start by showing in Figure 5.10 a DSS AI model **(a)**, its corresponding synthetic seismic **(b)**, the point-wise average **(c)** and the point-wise standard deviation **(d)** computed over the ensemble of DSS AI models. As expected, the synthetic seismic data have no structure and do not match the observed seismic data. The average AI model shows the influence of the well data around the wells, but at locations far from the wells, the values tend to the average value of the distribution. Similarly, as these models were generated with geostatistical simulation, the standard deviation at the well locations is null and increases with the distance.

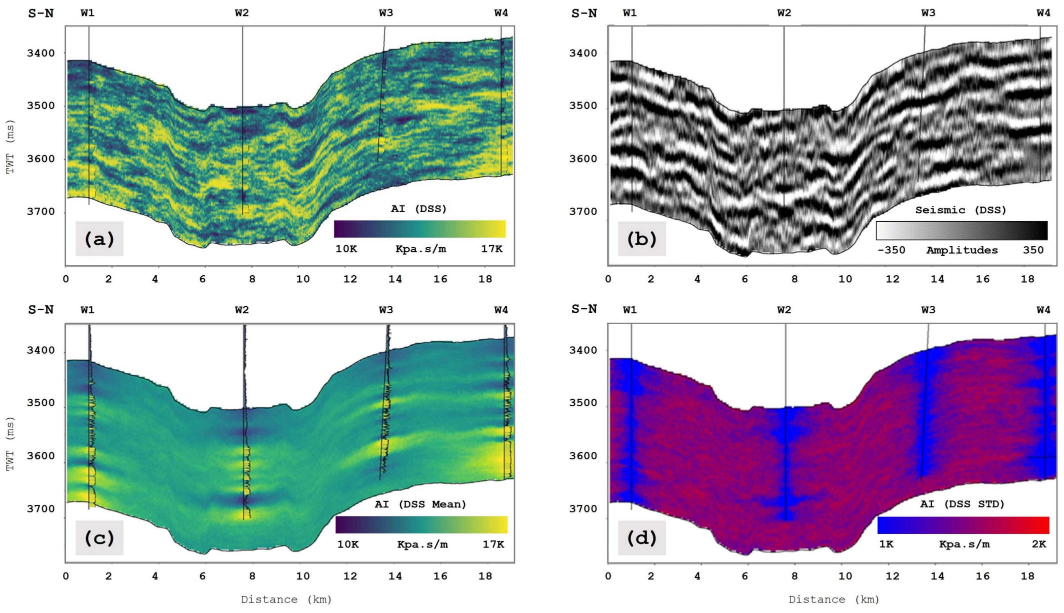


Figure 5.10: Vertical well section extracted from: **(a)** a DSS AI model, **(b)** synthetic seismic computed from (a), **(c)** point-wise average, and **(d)** point-wise standard deviation models computed over the set of DSS realizations.

Figures 5.11-a and -b show the vertical well sections extracted from: a DSS AI model and the result obtained when this DSS model and the observed seismic data are used as input to $f_\theta^{-1}$. To assess the robustness of the predictions we computed the point-wise average and the point-wise standard deviation models from the set of inverted AI models (Figures 5.11-c and -d, respectively).

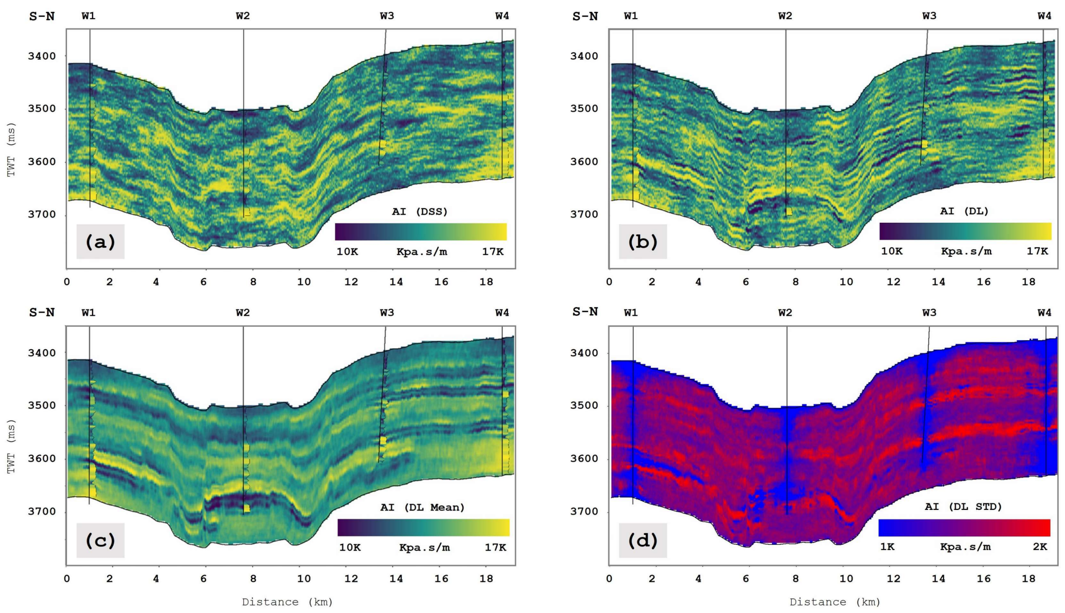


Figure 5.11: Vertical well sections extracted from: **(a)** a DSS AI model, **(b)** inverted AI model from (a) and observed seismic data, **(c)** point-wise average and **(d)** point-wise standard deviation models computed from the set of inverted AI models.

Figures 5.12-a and -b show the vertical well sections extracted from: the observed seismic data and the synthetic seismic data computed from the inverted AI model from 5.11-a. The point-wise average and the point-wise standard deviation models from the set of simulated seismic data are shown in Figures 5.12-c and -d, respectively.

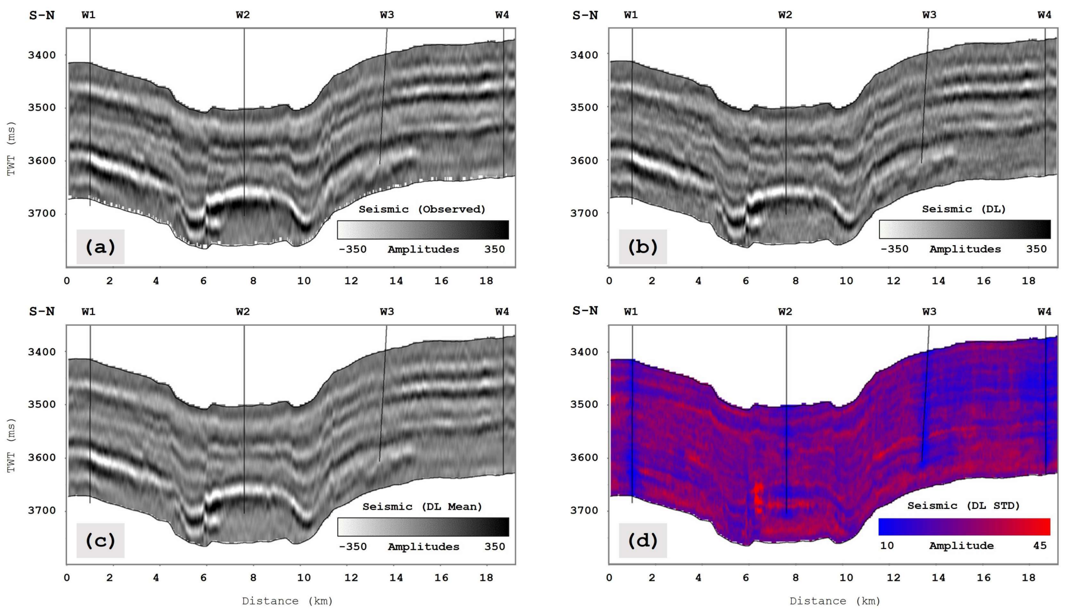


Figure 5.12: Vertical well section extracted from: **(a)** observed seismic data, **(d)** synthetic seismic computed from (a), **(c)** point-wise average and **(d)** point-wise standard deviation models computed from the set of synthetic seismic data.

The inverted AI model contains the structure of the observed seismic, while preserving the spatial continuity pattern from the DSS model (Figures 5.11-a and -b). The synthetic seismic data match the observed one in terms of amplitude content and location of the seismic reflections (Figures 5.12-a and -b).

The spatial continuity pattern of the point-wise average models agrees with the one observed in the observed seismic (Figures 5.11-c and 5.12-c). At the same time, they present the influence of the well data around the wells. The standard deviation (STD) models are null at the well locations and increase with the distance (Figures 5.11-d and 5.12-d). They also point to regions of higher uncertainty in locations where the signal-to-noise ratio of the original seismic is low, i.e., there is a larger variability when the geometry of the reflections is more complex (e.g., around well W2). This region of low STD values depends on the horizontal range of the variogram model imposed to build the ensemble of DSS AI models.

We next assess the results obtained at several locations within the inversion grid. Figure 5.13 shows a 3D visualization of an inverted AI model. Figure 5.14 shows inlines extracted from the inverted AI model along with the DSS AI model used as input, the synthetic seismic data, and the observed seismic data. For each output image, we show the correlation coefficient (CC), the mean absolute error (MAE) and the structural similarity index (SSIM) between the inputs and outputs. The global correlation coefficient between the observed and synthetic seismic data for this inverted AI model is 0.85.

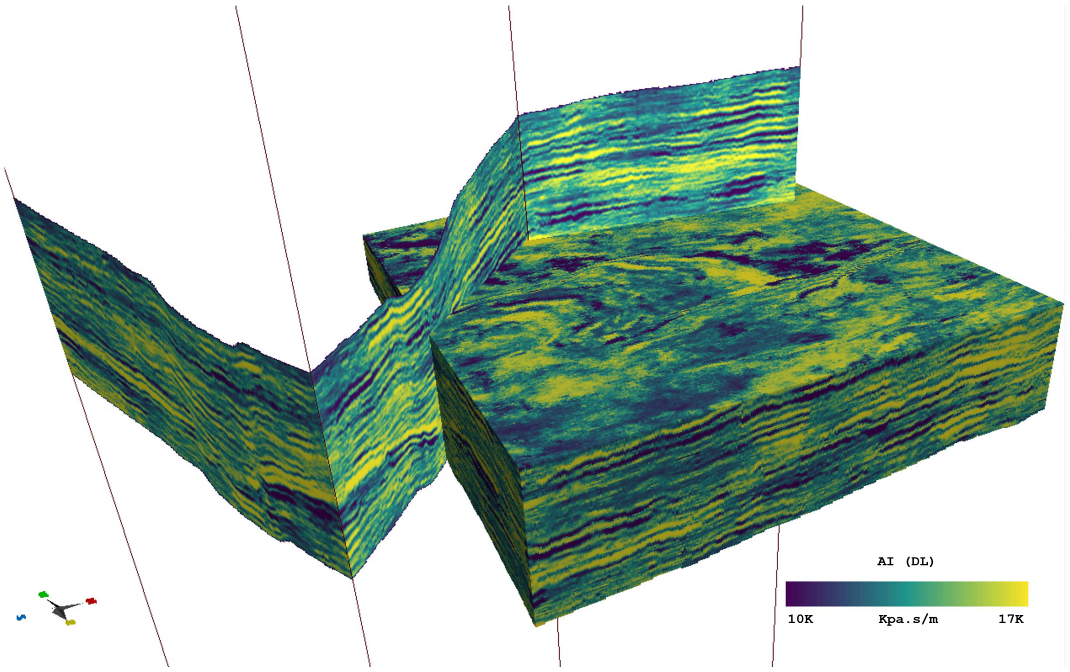


Figure 5.13: 3D visualization of an inverted AI model.

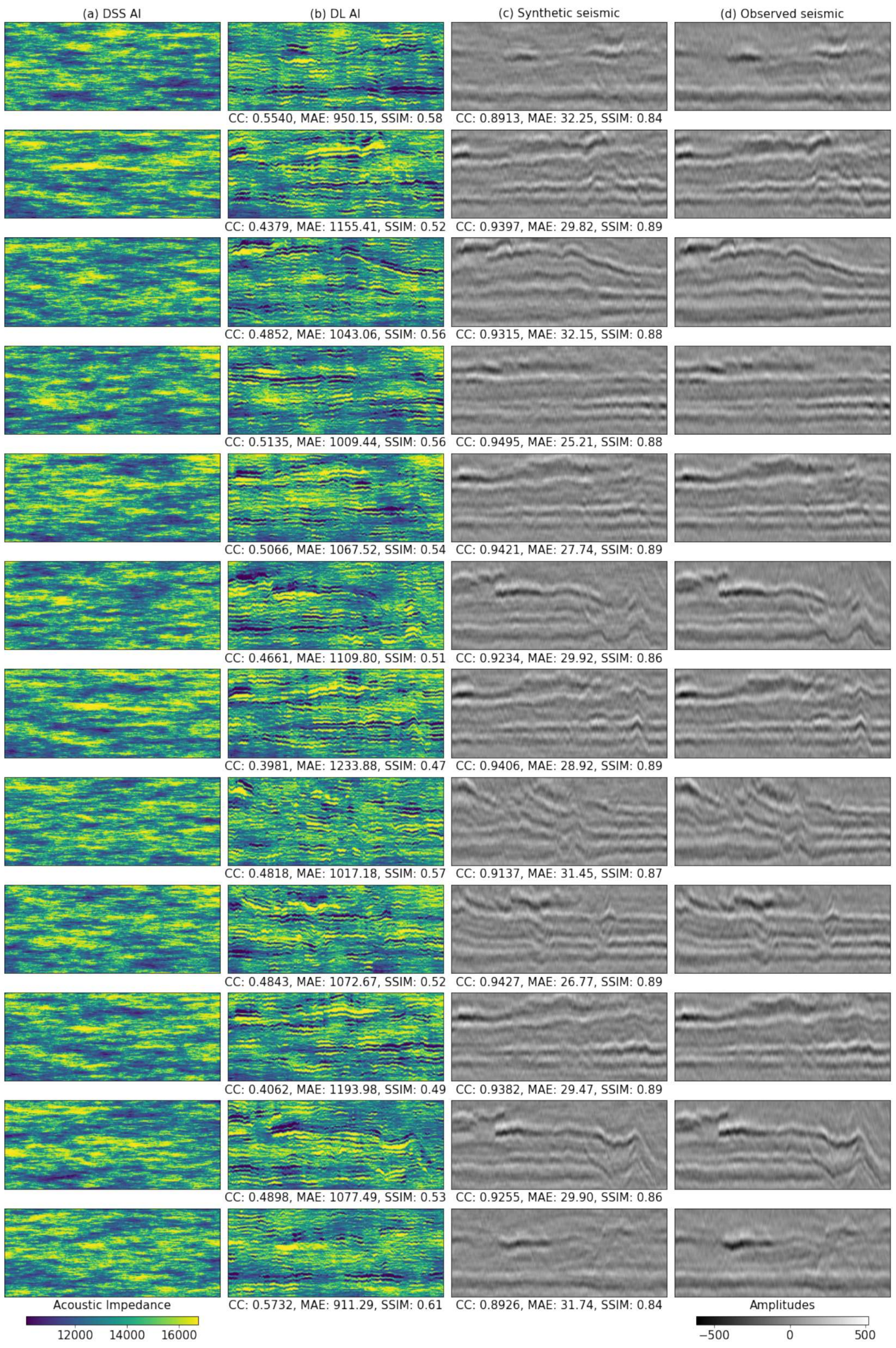


Figure 5.14: Inversion grid inlines extracted from: **(a)** the DSS AI model, **(b)** the inverted AI model, **(c)** the synthetic seismic computed from (b), and **(d)** the observed seismic. Below each predicted image are the correlation coefficient (CC), the mean absolute error (MAE) and the structural similarity index (SSIM) between the inputs and outputs.

Figure 5.15 compares the experimental variogram models (i.e., the spatial continuity pattern) between the DSS and the inverted AI models. The variograms for the inverted model are similar to the variograms of the DSS model and the ranges for all directions were also preserved.

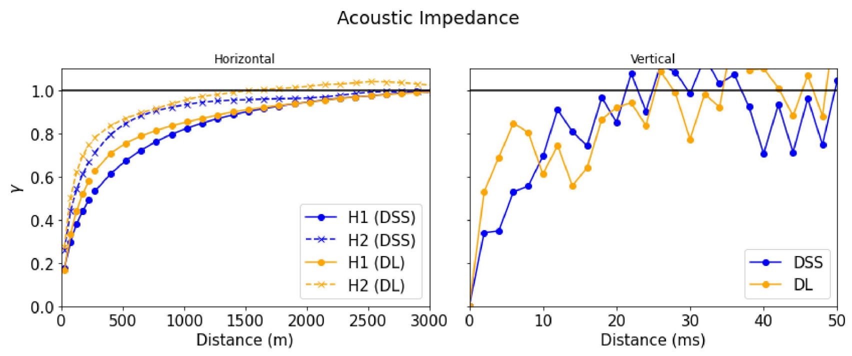


Figure 5.15: Comparison between the variograms obtained from the DSS AI model and the inverted AI model, calculated along the directions of maximum continuity (H1), minimum continuity (H2), and along the vertical direction.

We then evaluate the performance of the proposed inversion method at the well locations. Figure 5.16 shows a comparison between the measured AI well-logs and the predicted ones. For each well, we computed the correlation coefficient between the up-scaled well-log and the values extracted from the predicted point-wise average AI model at the well locations.

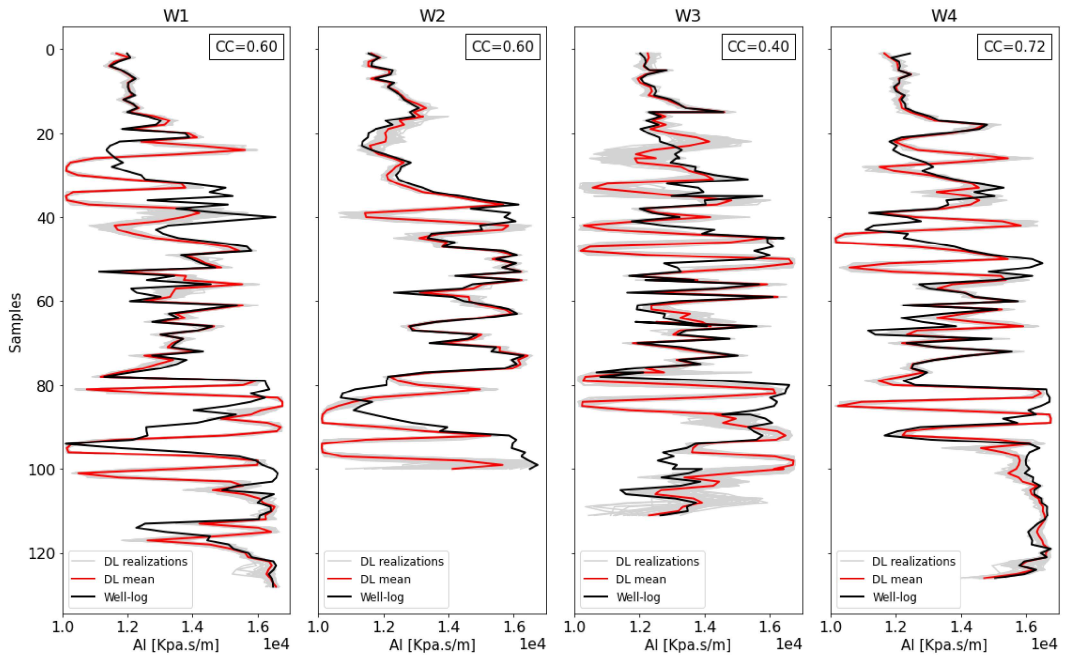


Figure 5.16: AI values at the well locations for the inverted AI models (light gray lines), the up-scaled well-logs (black lines), and the point-wise average over the set of inverted AI models (red lines). The correlation coefficients (CC) were computed between the up-scaled well-logs and the values extracted from the point-wise average AI model at the well locations.

Ultimately, the comparison between the prior and predicted statistics is an additional representation of the goodness of fit of the produced models. A comparison between the histograms from the input DSS model and the inverted AI models at the well locations is shown in Figure 5.17. The prior and the predicted histograms are similar in terms of shape and maximum and minimum.

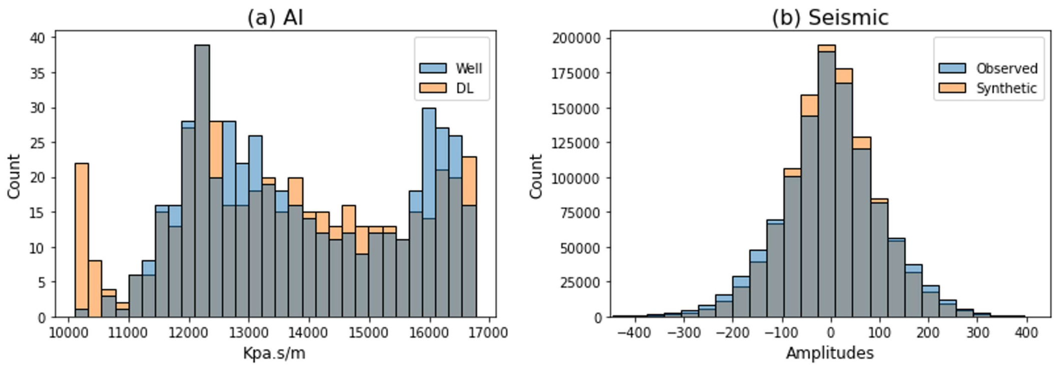


Figure 5.17: Original and predicted histograms for the AI models and the seismic data at the well locations.

### 5.2.2
### Amplitude-Versus-Angle (AVA) Inversion

We next present the results of the proposed **Deep Physics-Driven Stochastic AVA Inversion**. This AVA inversion ran at a similar time to that of the acoustic inversion ($\approx$ 60 minutes).

To assess the results obtained with the proposed method, we show one set of elastic property models of density, $V_P$ and $V_S$ generated with deep learning (Figure 5.18) and the synthetic partial angle stacks calculated from these models (Figure 5.19). The inverted elastic models do reproduce the DSS elastic models and exhibit the spatial patterns observed in the observed seismic reflection. The synthetic seismic data computed for each partial angle stacks also reproduce the location of the main seismic reflections and their amplitude content.

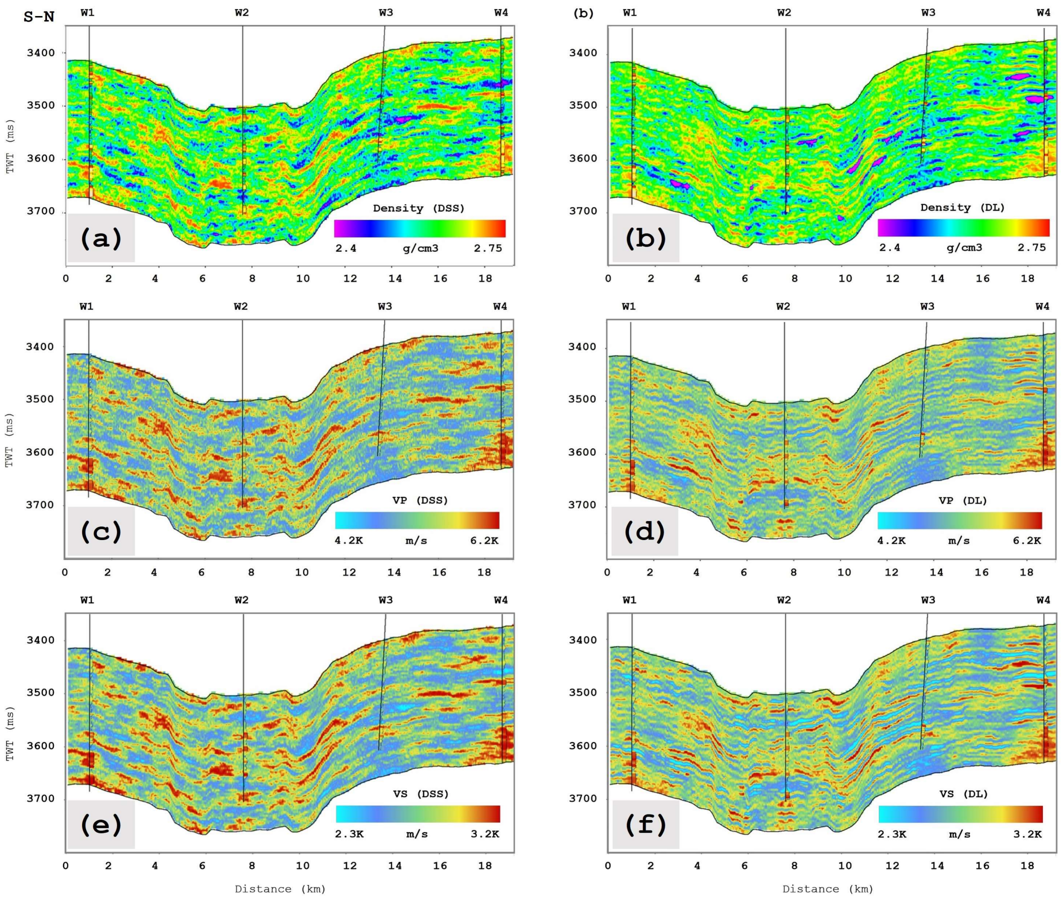


Figure 5.18: Vertical well sections extracted from: the DSS models of **(a)** density, **(c)** $V_P$, and **(e)** $V_S$, and the predicted models of **(b)** density, **(d)** $V_P$ and **(f)** $V_S$.

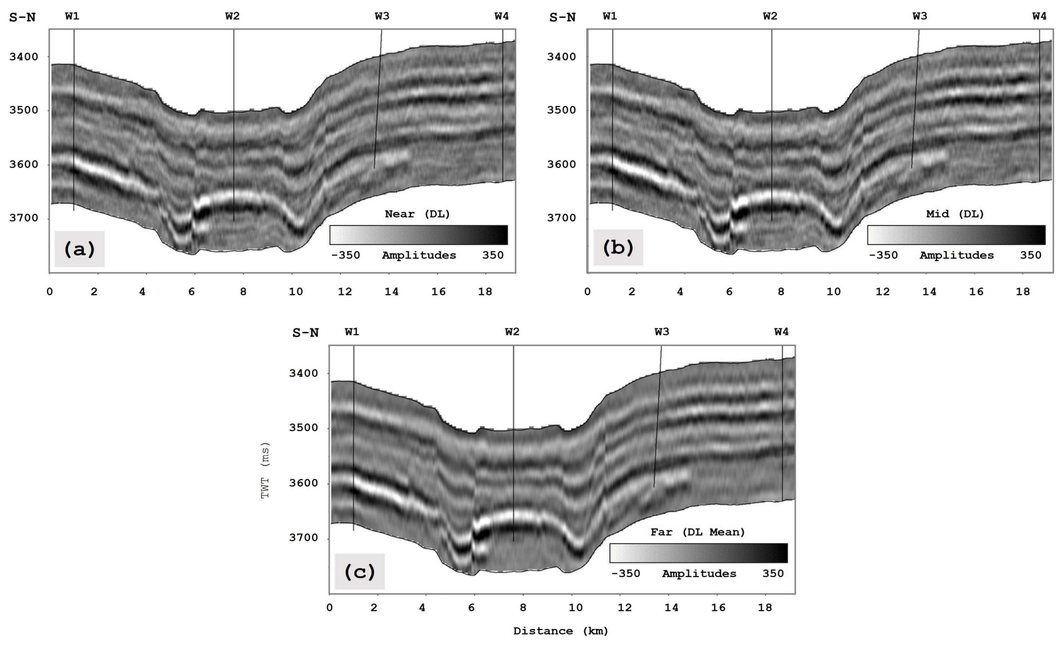


Figure 5.19: Vertical well section extracted from: **(a)** the synthetic near, **(b)** mid, and **(c)** far partial angle stacks simulated from the triplet of inverted elastic models in Figure 5.18.

Figure 5.20 shows a 3D visualization of an inverted $V_S$ model. Figures 5.21 and 5.22 show inlines extracted from the triplet of inverted elastic models along with the input DSS models and the observed and synthetic partial angle stacks. In both figures, we provide a quantitative analysis of the similarity between the inputs and outputs by computing the correlation coefficient (CC), the mean absolute error (MAE), and the structural similarity index (SSIM).

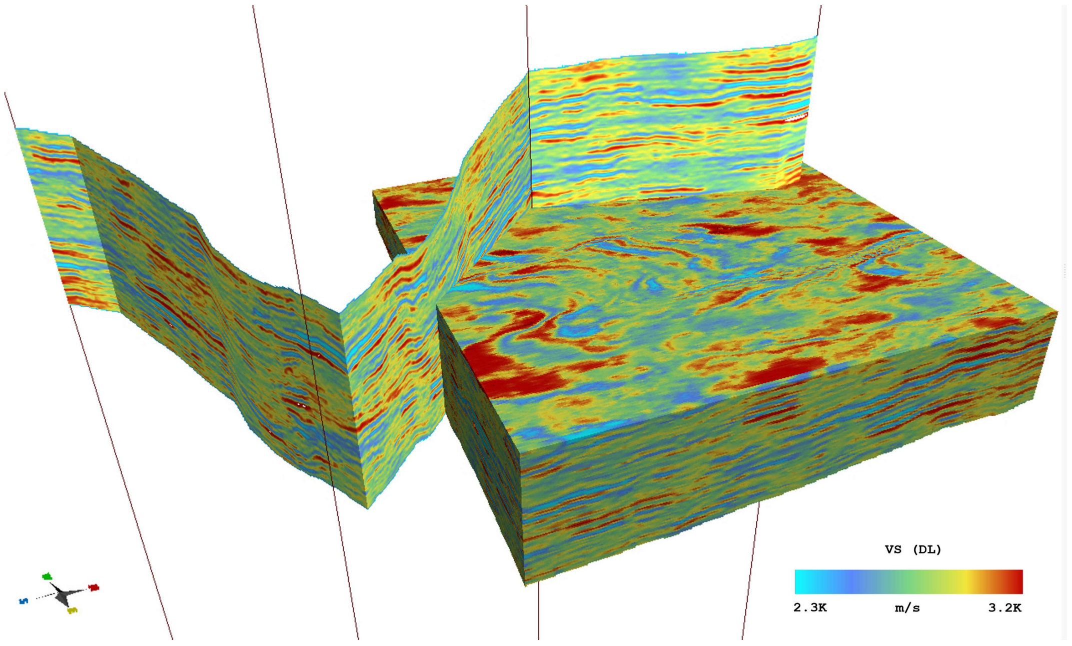


Figure 5.20: 3D visualization of an inverted $V_S$ model.

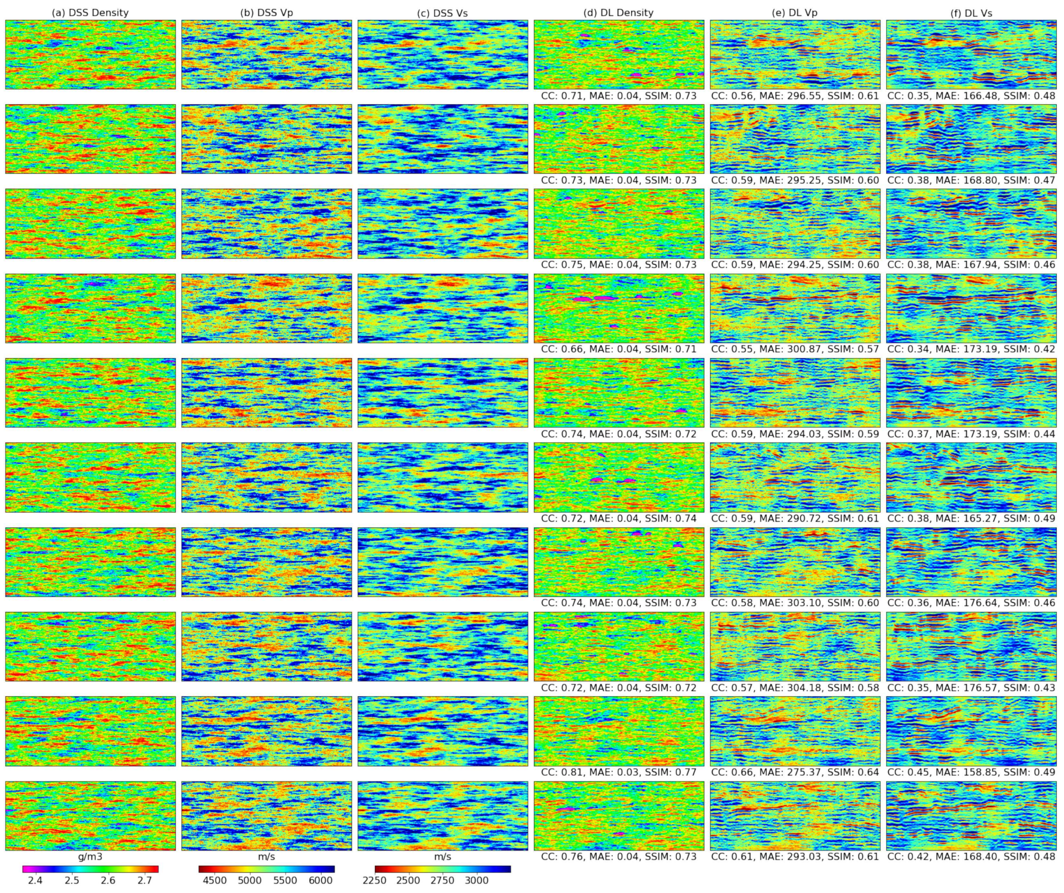


Figure 5.21: Inversion grid inlines extracted from: **(a)** the density, **(c)** $V_P$ and **(e)** $V_S$ DSS models, and **(b)** the inverted density, **(d)** the $V_P$ and **(f)** $V_S$ models. Below each predicted image are the correlation coefficient (CC), the mean absolute error (MAE), and the structural similarity index (SSIM) between the inputs and outputs.

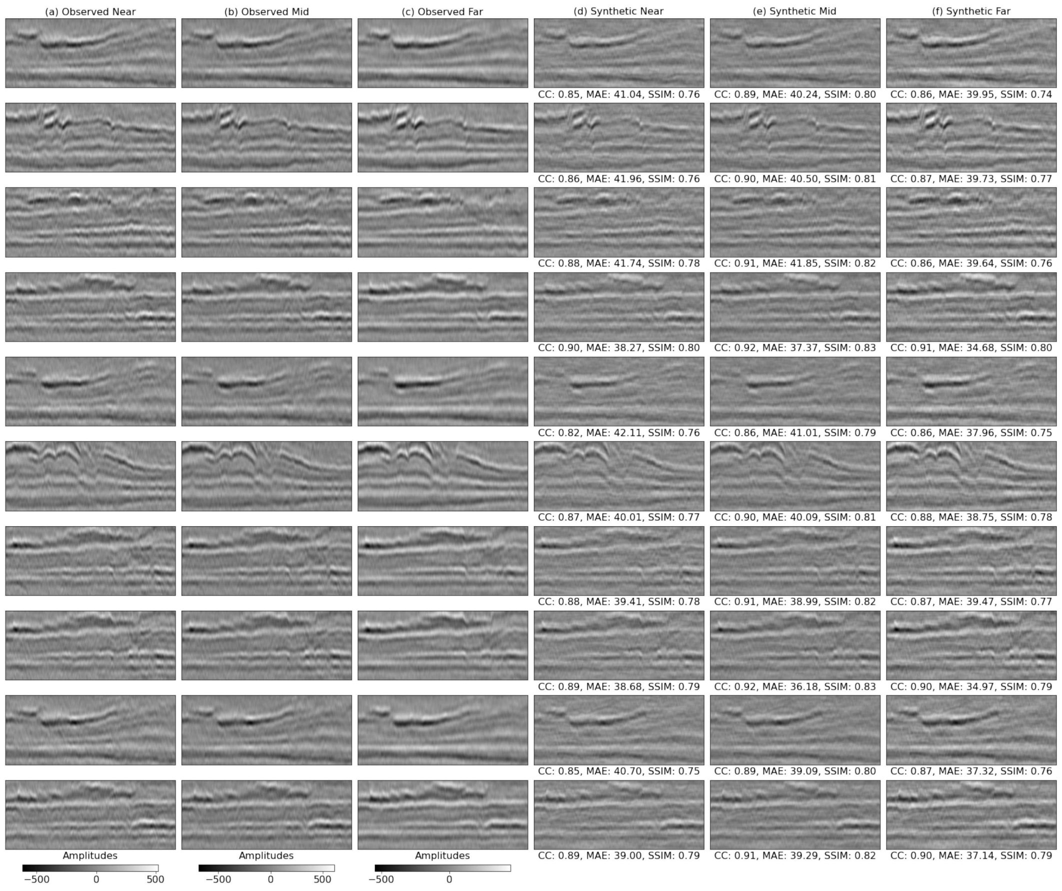

(a) Observed Near
(b) Observed Mid
(c) Observed Far
(d) Synthetic Near
(e) Synthetic Mid
(f) Synthetic Far
CC: 0.85, MAE: 41.04, SSIM: 0.76
CC: 0.89, MAE: 40.24, SSIM: 0.80
CC: 0.86, MAE: 39.95, SSIM: 0.74
CC: 0.86, MAE: 41.96, SSIM: 0.76
CC: 0.90, MAE: 40.50, SSIM: 0.81
CC: 0.87, MAE: 39.73, SSIM: 0.77
CC: 0.88, MAE: 41.74, SSIM: 0.78
CC: 0.91, MAE: 41.85, SSIM: 0.82
CC: 0.86, MAE: 39.64, SSIM: 0.76
CC: 0.90, MAE: 38.27, SSIM: 0.80
CC: 0.92, MAE: 37.37, SSIM: 0.83
CC: 0.91, MAE: 34.68, SSIM: 0.80
CC: 0.82, MAE: 42.11, SSIM: 0.76
CC: 0.86, MAE: 41.01, SSIM: 0.79
CC: 0.86, MAE: 37.96, SSIM: 0.75
CC: 0.87, MAE: 40.01, SSIM: 0.77
CC: 0.90, MAE: 40.09, SSIM: 0.81
CC: 0.88, MAE: 38.75, SSIM: 0.78
CC: 0.88, MAE: 39.41, SSIM: 0.78
CC: 0.91, MAE: 38.99, SSIM: 0.82
CC: 0.87, MAE: 39.47, SSIM: 0.77
CC: 0.89, MAE: 38.68, SSIM: 0.79
CC: 0.92, MAE: 36.18, SSIM: 0.83
CC: 0.90, MAE: 34.97, SSIM: 0.79
CC: 0.85, MAE: 40.70, SSIM: 0.75
CC: 0.89, MAE: 39.09, SSIM: 0.80
CC: 0.87, MAE: 37.32, SSIM: 0.76
CC: 0.89, MAE: 39.00, SSIM: 0.79
CC: 0.91, MAE: 39.29, SSIM: 0.82
CC: 0.90, MAE: 37.14, SSIM: 0.79
Amplitudes
−500
0
500
Amplitudes
−500
0
500
Amplitudes
−500
0

Figure 5.22: Inversion grid inlines extracted from: the observed **(a)** near, **(b)** mid, and **(c)** far partial angle stacks, and the synthetic **(d)** near, **(e)** mid, and **(f)** far partial angle stacks. Below each predicted image are the correlation coefficient (CC), the mean absolute error (MAE) and the structural similarity index (SSIM) between the inputs and outputs.

We then evaluate the performance of the proposed method at the location of well W1. Figure 5.23 shows a comparison between the measured well-logs and the predicted ones. The correlation coefficient between the up-scaled well-logs and the values extracted from the predicted point-wise average models at the well location are $\mathrm{CC}_{density} = 0.81$, $\mathrm{CC}_{V_P} = 0.715$ and $\mathrm{CC}_{V_S} = 0.585$.

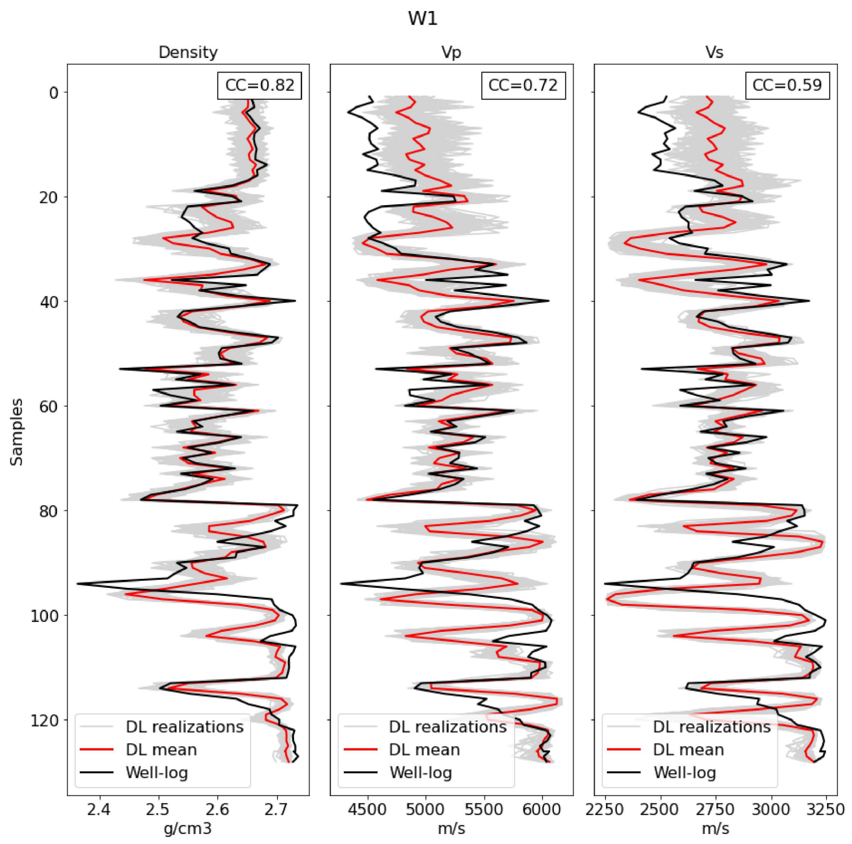


Figure 5.23: Density, $V_P$ and $V_S$ values at the location of well W1 for the inverted elastic models (light gray lines), the up-scaled well-logs (black lines), and the point-wise average over the set of inverted elastic models (red lines). The correlation coefficients (CC) are computed between the up-scaled well-logs and the values extracted from the point-wise average elastic models at the well location.

As we are using direct sequential simulation and co-simulation methods to generate the set of input AI models, these resulting elastic models do reproduce the marginal and joint distributions retrieved from the well-log data. Figure 5.24 illustrates the prior and predicted distributions of the elastic models at the well locations. The predicted elastic models do reproduce the prior distributions.

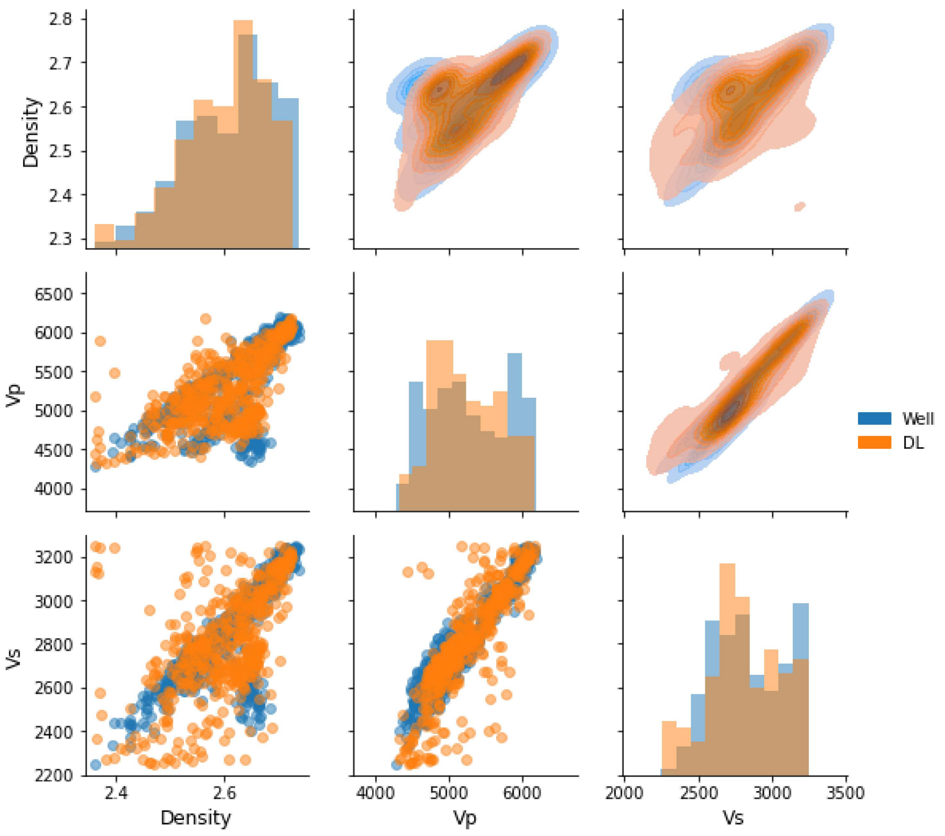


Figure 5.24: Comparison between the marginal and joint distributions extracted from the DSS elastic models and from the inverted elastic models at the well locations.

We also show in Figure 5.25 that the synthetic partial angle stacks do reproduce the histograms of the observed seismic data.

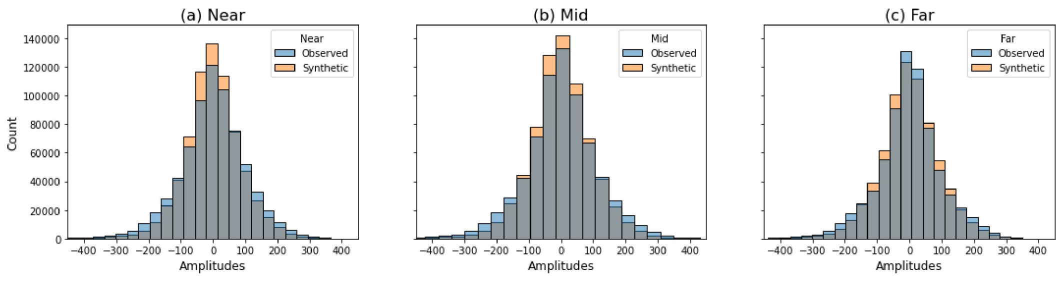


Figure 5.25: Original and predicted histograms of the seismic data at the well locations.

Finally, we show a comparison between the experimental variograms (i.e., the spatial continuity pattern) computed from the DSS and inverted elastic models. The horizontal variograms are shown in Figures 5.26-a and -c. Figures 5.26-b and -d present the vertical variograms. The variograms for the predicted values are similar to the variograms of the DSS models, and the ranges for all directions are also preserved.

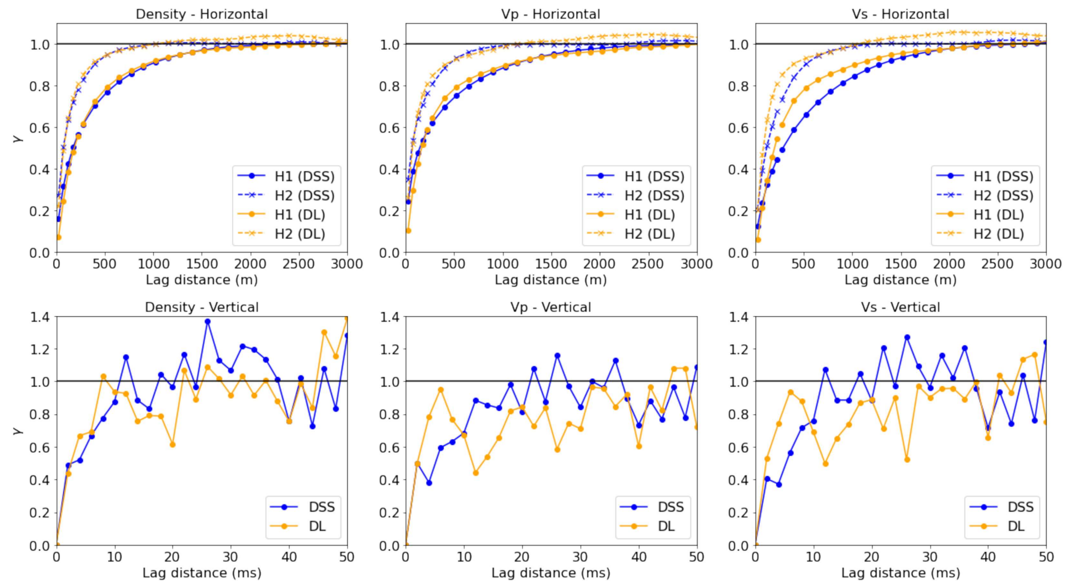


Figure 5.26: Comparison between the variograms obtained from the DSS and from the inverted elastic models, calculated along the directions of maximum continuity (H1), minimum continuity (H2), and along the vertical direction.

See Appendix B for complementary results on this AVA inversion.

### 5.2.3 Petrophysical Inversion

We further validate the proposed **Deep Physics-Driven Stochastic Petrophysical Inversion** on the same 3D real case application. Using the available well-data, we calibrated a rock physics model to infer acoustic impedance (AI) from porosity ($\phi$) based on a linear relation of the form:

$$AI = -35794 \times \phi + 16489 \tag{5-1}$$

Figure 5.27-b shows the inverted porosity model with the proposed method when the DSS model from Figure 5.27-a and the observed seismic data are used as input to $f_\theta^{-1}$. The inverted porosity model contains the structure of the observed seismic, while preserving the spatial continuity pattern from the DSS model.

The predicted AI model obtained from the inverted porosity model using equation 5-1 is shown in Figure 5.27-c. The synthetic seismic data retrieved from the predicted AI model is shown in Figure 5.27-d. The synthetic seismic data match the observed ones in terms of amplitude content and location of the seismic reflections.

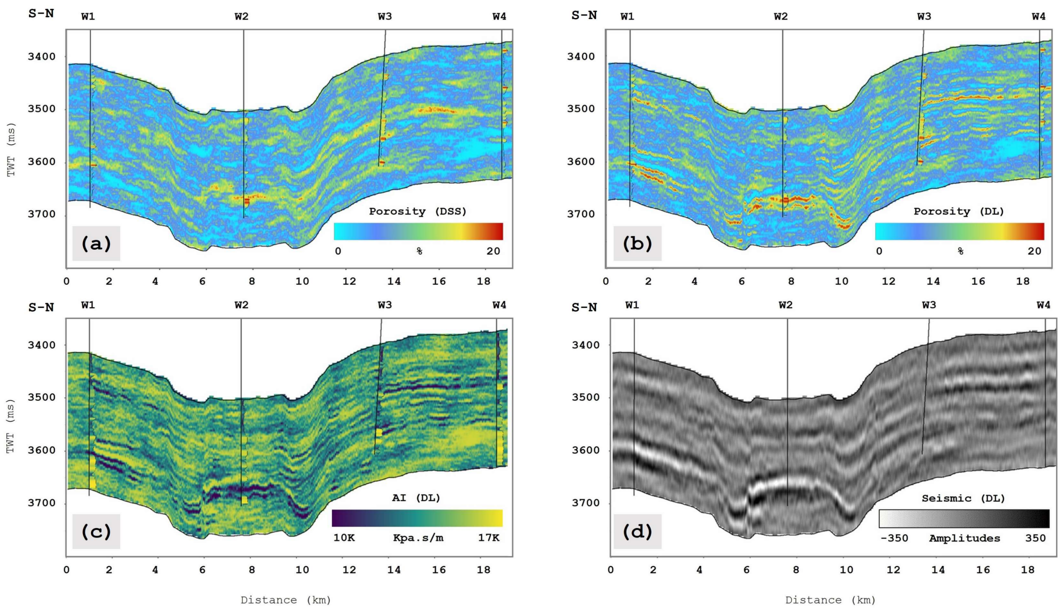


Figure 5.27: Vertical well sections extracted from: **(a)** a DSS porosity model; **(b)** the inverted porosity model from (a), **(c)** the predicted AI model from (b), and **(d)** synthetic seismic data from (c).

Figure 5.28 shows inlines extracted from the inverted porosity model along with the input DSS model, the predicted AI model, and the observed and synthetic seismic data. We provide a quantitative analysis of the similarity between the inputs and outputs by computing the correlation coefficient (CC), the mean absolute error (MAE) and the structural similarity index (SSIM).

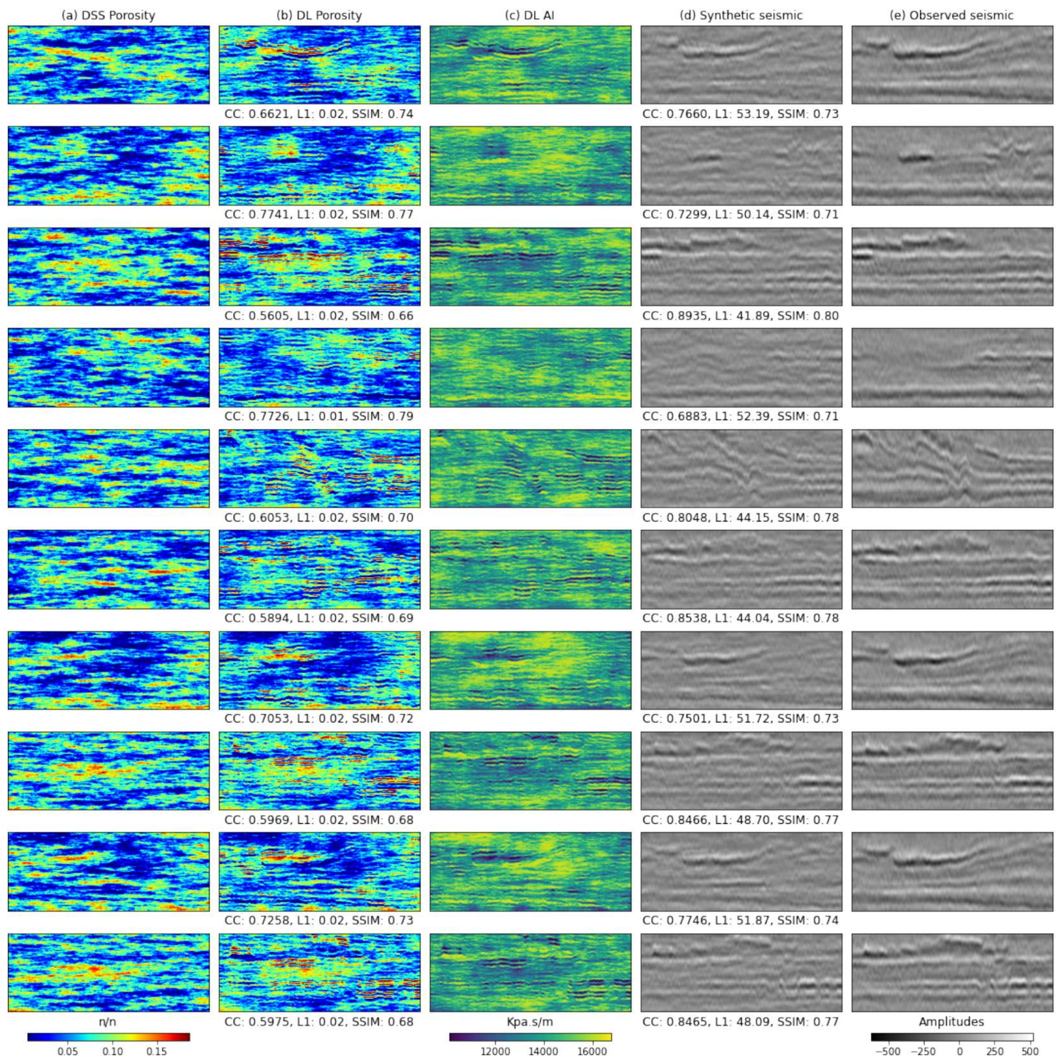


Figure 5.28: Inversion grid inlines extracted from: **(a)** a DSS porosity model, **(b)** the inverted porosity model, **(c)** the predicted AI model from (b), **(d)** the synthetic seismic data computed from (c), and **(e)** the observed seismic data. Below the porosity and synthetic seismic images are the correlation coefficient (CC), the mean absolute error (MAE) and the structural similarity index (SSIM) between the inputs and outputs.

We then evaluate the performance of the proposed method at the well locations. Figure 5.29 shows a comparison between the measured well-logs and the predicted ones. The correlation coefficient between the up-scaled well-logs and the values extracted from the point-wise average porosity model at the well location are $\mathrm{CC}_{W1} = 0.676$, $\mathrm{CC}_{W2} = 0.635$, $\mathrm{CC}_{W3} = 0.462$ and $\mathrm{CC}_{W4} = 0.743$.

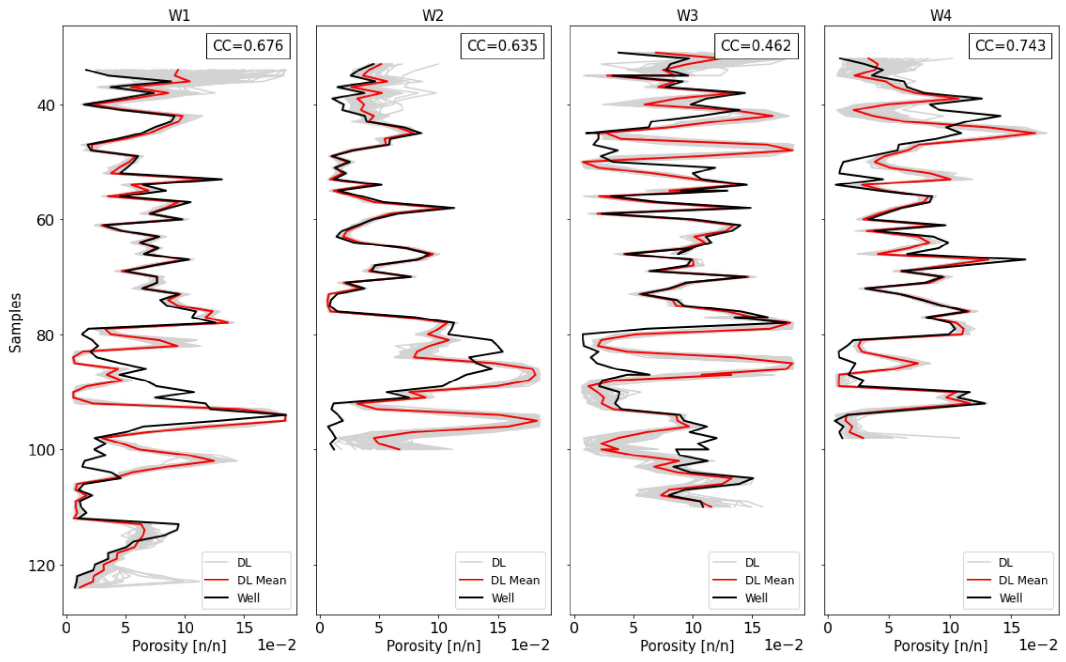


Figure 5.29: Porosity values at the well locations for the inverted porosity models (light gray lines), the up-scaled well-logs (black lines) and the point-wise average over the set of inverted porosity models (red lines). The correlation coefficients (CC) are computed between the up-scaled well-logs and the values extracted from the point-wise average AI model at the well locations.

Figure 5.30 illustrates the prior and predicted distributions of the porosity values extracted at the well locations. The inverted porosity models do reproduce the prior distributions.

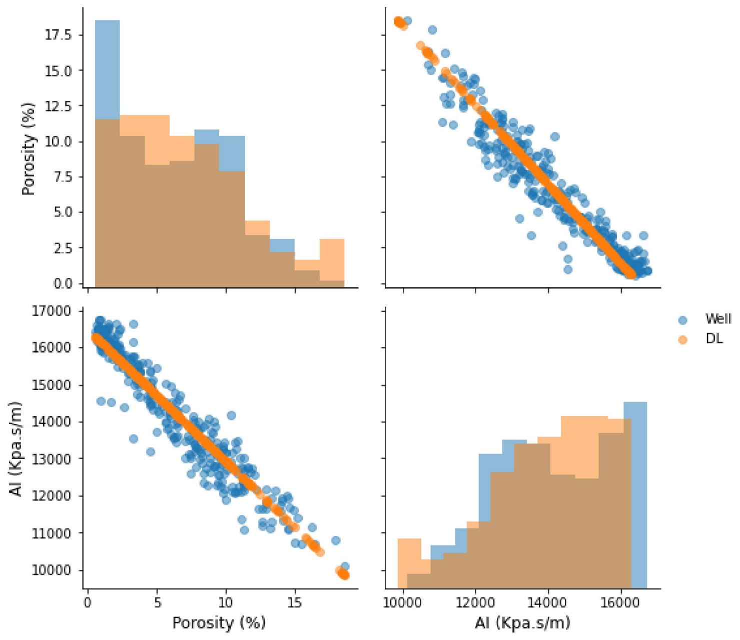


Figure 5.30: Comparison between the histograms extracted from the DSS porosity models and from the inverted porosity model at the well locations.

We also show in Figure 5.31 that the histogram of the synthetic seismic data do reproduce the histogram of the observed seismic data.

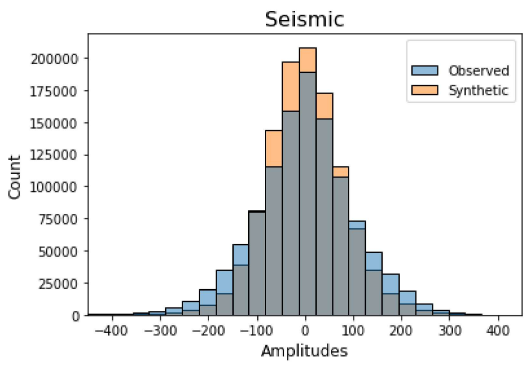


Figure 5.31: Original and predicted histograms of the seismic data at the well locations.

Finally, we show a comparison between the experimental variograms (i.e., the spatial continuity pattern) computed from the DSS and inverted porosity models. The variograms for the predicted values are similar to the variograms of the DSS model and the ranges for all directions are also preserved.

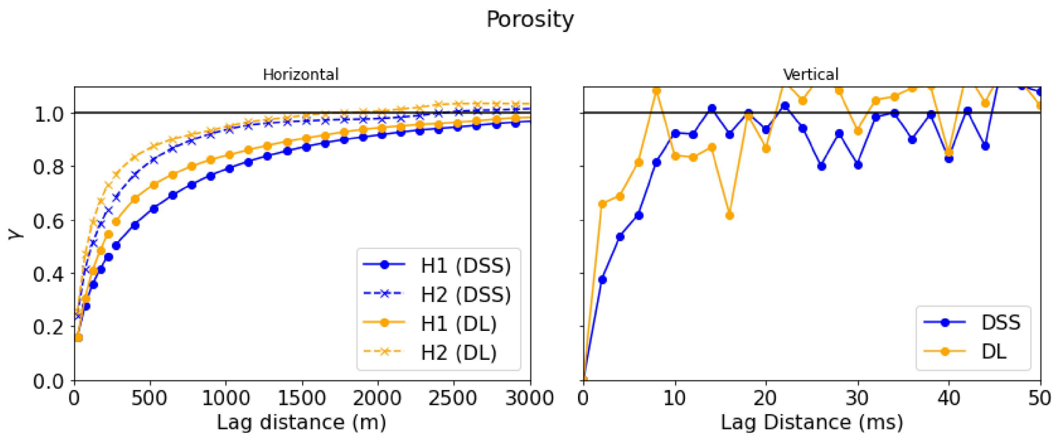


Figure 5.32: Comparison between the variograms obtained from the DSS and the inverted porosity models, calculated along the directions of maximum continuity (H1), minimum continuity (H2), and along the vertical direction.

Appendix B provides complementary results on this application example.

## 5.3 Comparison with the Geostatistical Seismic Inversion Method

We compare the results obtained with the proposed **Deep Physics-Driven Stochastic Acoustic Inversion** against the Geostatistical Seismic Inversion Method (GSI) method [64, 19]. For the real case application dataset, we ran the GSI method on a Xeon Gold 6242 processor with 32-cores (2.8GHz, DDR4-2933, 12GB) with 6 iterations and 100 realizations. We started from the same ensemble of geostatistical realizations used as input to our application example.

Figure 5.34 shows vertical sections extracted from inverted AI models and corresponding synthetic seismic data using both methods. A visual analysis of the inverted AI models indicates that the model inverted with our method presents higher lateral and vertical variability than the GSI one.

The point-wise average and standard deviation results over all the inverted models for both methods are shown in Figure 5.35. Both methods have considerable differences regarding the predicted spatial uncertainty as represented by the point-wise standard deviation.

In GSI, we often observe a decrease in the exploration of the uncertainty space due to fast convergence in initial iterations [119]. The variability of the inverted model with DL presents higher values and a more uniform spatial distribution. These results indicate that our method was able to explore far more of the uncertainty space when compared with the GSI method.

To corroborate this idea, we compare the inverted AI models in a lower dimension space computed with multidimensional scaling (MDS) [120]. MDS is a method for reducing the dimensionality of the data while maintaining the distances in the original high-dimensional space. It is used in data sciences for analyzing similarity or dissimilarity data. In seismic inversion, the distribution of the inverted models in the MDS space allows an estimation of how the uncertainty space is being sampled [121]. By plotting all the inverted AI models with both methods in the MDS space (Figure 5.33), it is easily recognizable that the parameter model space is considerably better explored by the models computed with the proposed method.

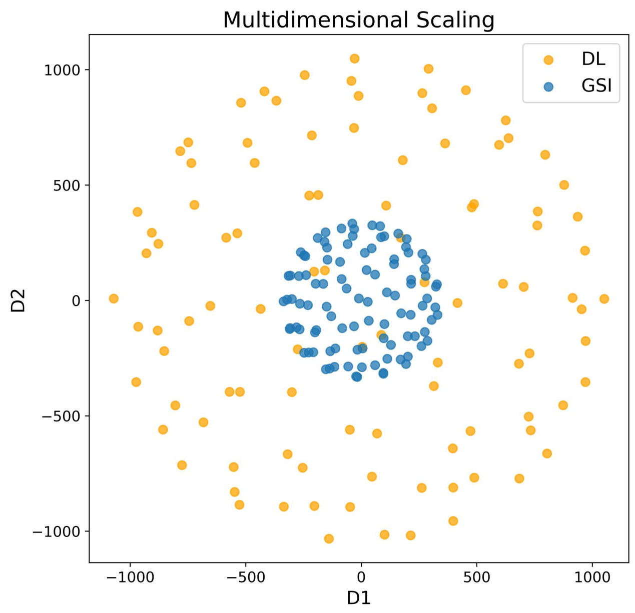


Figure 5.33: Inverted models with the proposed and with the GSI methods plotted in a MDS space of two dimensions - D1 and D2.

In terms of perceptual visual quality, the synthetic seismic data obtained with the proposed method does not present the artifacts caused by the trace-by-trace-based optimization from the GSI method. The correlation coefficient between the synthetic and observed seismic data of GSI is 0.88, quite close to the correlation coefficient obtained with the proposed method (0.85).

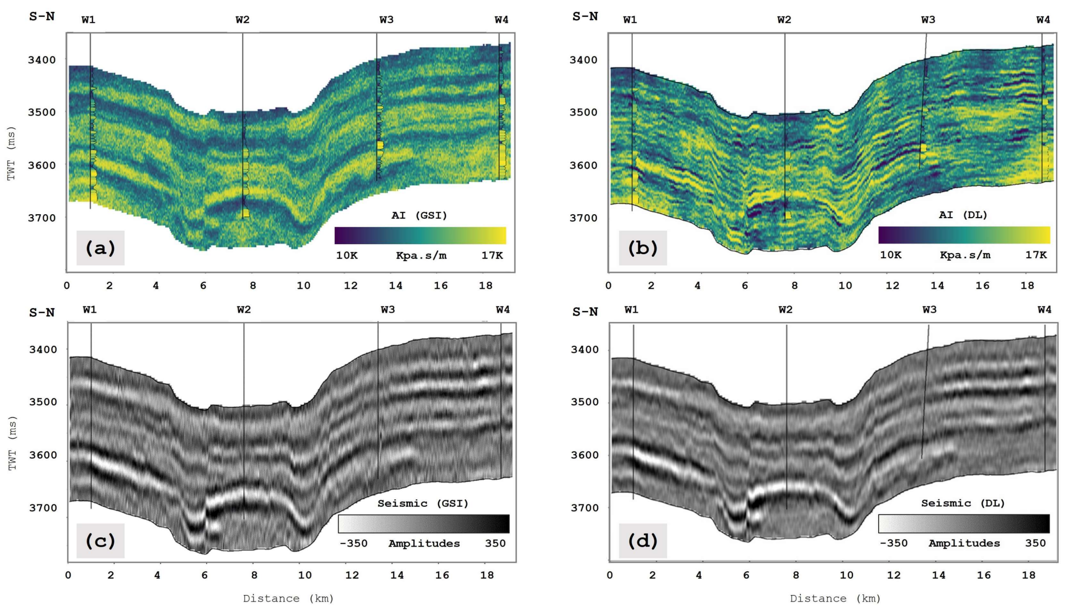


Figure 5.34: Comparison between vertical well sections extracted from an inverted AI model using: **(a)** the GSI method, **(b)** the proposed DL method, and the synthetic seismic obtained with: **(c)** the GSI method, and **(d)** the proposed DL method.

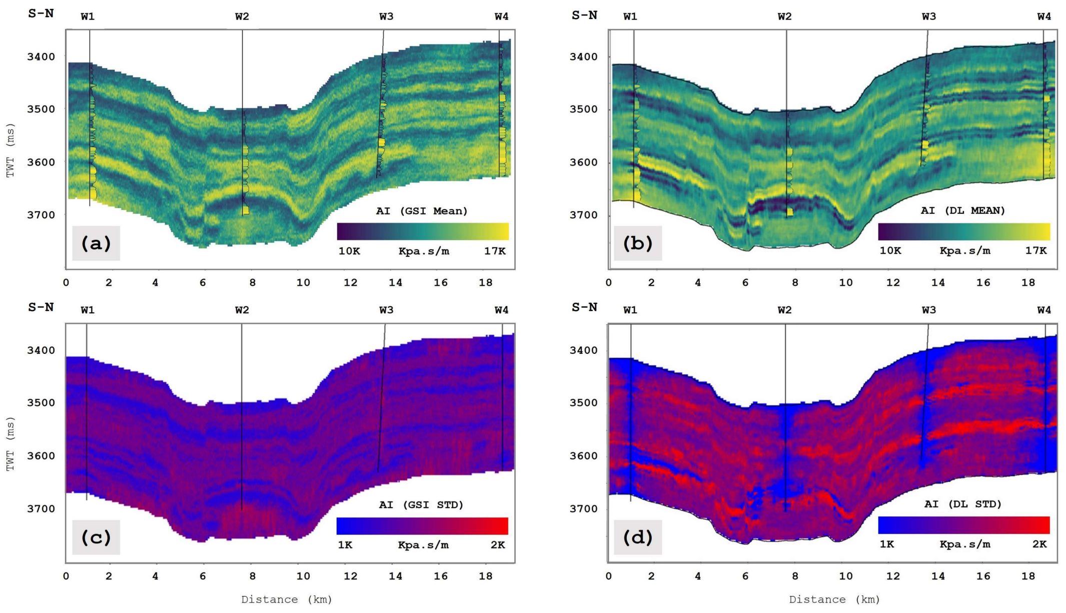


Figure 5.35: Comparison between vertical well sections extracted from point-wise average of the inverted AI models with: **(a)** the GSI method, **(b)** the proposed DL method, and point-wise standard deviation of the inverted AI models with: **(c)** the GSI method, and **(d)** the proposed DL method.

GSI is computationally expensive mainly due to the perturbation of the model parameter space, which is performed with stochastic sequential simulation and co-simulation. In this implementation, each individual realization is parallelized given the number of CPUs available in the system. It took 45 minutes to perform the simulation step for each iteration. The total processing time, considering all the calculations of the inversion procedure, was about 3 hours.

The time complexity for deep-learning models is evaluated in terms of the total time taken by the training and the prediction time. This may involve millions of calculations and be quite expensive computationally. However, most of these matrix operations are performed in parallel on GPUs which can considerably reduce the processing time. In practice, a comparison between the computational complexity of the GSI, and the proposed method is not straightforward as both methods leverage different parallelization strategies.

It took approximately one hour to train the inversion network for the real application example. Considering the initial step of generation of the ensemble of AI models with stochastic sequential simulation and the prediction step, the total processing time of our method was very similar to the GSI one. However, in the proposed method, as the network is trained on a subset of the prior models' ensemble, the training time does not scale with the number of estimates and the inference is performed almost in real time. As a consequence, for a large number of realizations, the computational cost of our method is dramatically improved when compared to the GSI one.

# 6
# Conclusion

We introduced a novel physics-driven DL methodology to invert 3D high-resolution subsurface models from reflection seismic data and a limited number of well-logs. There are two major advantages to our approach: 1) it provides uncertainty estimation about the inverted models and 2) it does not require annotated training data.

The method proposed herein builds on two neural networks that act as the seismic inverse and forward models. The "physics-driven" part comes from the forward network, which is used to convolve the source seismic pulse with a reflectivity series devised from the outcomes of the inverse network.

We base our approach on a geostatistical framework to assess uncertainty. An ensemble of prior simulated models with geostatistics is used to provide additional information and to induce stochasticity in the input layer of the inverse network. In contrast to other existing DL-based methods, our approach is fully unsupervised, only requiring prior information on the spatial distribution pattern and available well-log data.

The inverse network is trained to simultaneously minimize the misfits between the synthetic and observed seismic data, and between the inverted model and the geostatistical model. A deep convolutional neural network of the type encoder-decoder was used to help with the extraction and reconstruction of the relevant features from the seismic data and the geostatistical models.

The generalization and parameterization of the proposed method are discussed and verified in a synthetic one-dimensional case. We show that the size of the train set can be empirically defined by testing the inverse network against a non-overlapping test set. As shown by our real case application, the computational cost of the proposed methodology is not prohibitive. We performed a series of experiments to demonstrate the influence of the confidence matrix on the performance of the network.

The robustness of our methodology is demonstrated in a practical application example - a real 3D case study related to the Albian carbonate sequence in the Brazilian offshore. We carried out the proposed workflows for acoustic, amplitude-versus-angle (AVA), and petrophysical seismic inversion.

The absolute value of the model reconstruction loss, the absolute value of seismic data residuals, and the correlation coefficient between the inputs and outputs to the network are used to quantitatively evaluate our method. We additionally use the structural similarity index between inputs and outputs for a better assessment of the preservation of the structural information.

The application of the real case study demonstrated that the network successfully captured and preserved the desired features from the inputs - the spatial distribution from the prior simulated models, and the large-scale structures and spatial continuity from the seismic data. The inverted models presented a good fit to the observed seismic data.

A comparison with the existing geostatistical seismic inversion method (GSI) was carried out. The inversion results are consistent with those obtained from the GSI method. We achieved comparable performance, considering the quantitative metric. From the aspect of visual perception quality, our inversion results are more consistent with the target.

We assessed the exploration of the model parameters space by plotting the posterior models in a lower dimension space computed with multidimensional scaling (MDS) [120]. In spite of sampling from the same region, the solution space is considerably better explored by the models computed with the proposed method.

The GSI method is parallelized at the CPU level, and our method runs on multiple GPUs. For this reason, it is hard to compare the computational costs of both methods. It is important to stress that with our hardware configuration, it is possible to generate a larger number of alternative solutions faster than with the GSI method. This is because the computational cost related to our method is mainly due to the training stage. The cost related to prediction is virtually zero. Therefore, unlike the GSI approach, our solution is well suited to generating thousands of alternate solutions for better uncertainty quantification.

The proposed **Deep Physics-Driven Stochastic Seismic Inversion** method produces improvements over the baselines and achieves promising performance on a 3D real application example according to quantitative and qualitative evaluation metrics. These results indicate that our methodology may possess great potential for inverting many possible property models to assess uncertainties with relatively fast convergence and lower computation cost.

## 6.1 Future Work

We do believe that the proposed method could be successfully applied in a different sedimentary depositional environment, as long as the seismic data has relatively high quality and the spatial continuity patterns of the ensemble of geostatistical realizations used to train the network agree (at a large-scale) with the true subsurface geology. This will require a reasonable number of wells and/or a good geological knowledge. An exhaustive discussion of the generalization of the proposed inversion, however, is beyond the scope of this work.

We foresee models being trained with specific data belonging to different sedimentary depositional environments, such as carbonate reef deposits, and submarine fan deposits.

In terms of implementation, one alternative to the pseudo-3D approach is to use a stack of adjacent slices as input and produce a prediction for the central slice. This strategy gives the network the possibility to better capture 3D spatial information, with only a minor additional computational cost.

# 7
# Bibliography

1 PASZKE, A. et al. Automatic differentiation in pytorch. In: **NIPS-W**. [S.l.: s.n.], 2017. Cited 2 times in pages 10 and 63.

2 ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Cited in page 10.

3 LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural Computation**, v. 1, n. 4, p. 541–551, 12 1989. ISSN 0899-7667. Cited 3 times in pages 10, 25, and 34.

4 LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep learning**. [S.l.]: Nature Publishing Group, 2015. 436–444 p. Cited 3 times in pages 10, 26, and 34.

5 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. Cited 10 times in pages 10, 25, 26, 28, 32, 33, 36, 38, 44, and 47.

6 ALOM, M. Z. et al. **A state-of-the-art survey on deep learning theory and architectures**. [S.l.]: MDPI AG, 2019. 292 p. Cited 2 times in pages 10 and 34.

7 ALZUBAIDI, L. et al. **Review of deep learning: concepts, CNN architectures, challenges, applications, future directions**. [S.l.]: Springer International Publishing, 2021. v. 8. ISSN 21961115. ISBN 4053702100444. Cited 3 times in pages 10, 26, and 34.

8 HUANG, L.; DONG, X.; CLEE, T. E. A scalable deep learning platform for identifying geologic features from seismic attributes. **The Leading Edge**, v. 36, p. 249–256, 2017. Cited in page 10.

9 XIONG, W. et al. Seismic fault detection with convolutional neural network. **Geophysics**, Society of Exploration Geophysicists, v. 83, n. 5, p. O97–O103, sep 2018. ISSN 19422156. Cited in page 10.

10 WU, X. et al. Convolutional neural networks for fault interpretation in seismic images. In: . [S.l.]: Society of Exploration Geophysicists, 2018. p. 1946–1950. Cited in page 10.

11 ZHENG, Y. et al. Applications of supervised deep learning for seismic interpretation and inversion. **Leading Edge**, Society of Exploration Geophysicists, v. 38, p. 526–533, 7 2019. ISSN 19383789. Cited 2 times in pages 10 and 46.

12 DI, H. et al. Imposing interpretational constraints on a seismic interpretation convolutional neural network. **Geophysics**, v. 86, n. 3, p. IM63–IM71, 2021. Cited in page 10.

13 LI, F. et al. ADDCNN: An Attention-Based Deep Dilated Convolutional Neural Network for Seismic Facies Analysis with Interpretable Spatial-Spectral Maps. **IEEE Transactions on Geoscience and Remote Sensing**, Institute of Electrical and Electronics Engineers Inc., v. 59, n. 2, p. 1733–1744, feb 2021. ISSN 15580644. Cited in page 10.

14 LIU, M. et al. Seismic facies classification using supervised convolutional neural networks and semisupervised generative adversarial networks. **Geophysics**, Society of Exploration Geophysicists, v. 85, n. 4, p. O47–O58, jul 2020. ISSN 0016-8033. Cited in page 10.

15 ADLER, A.; ARAYA-POLO, M.; POGGIO, T. Deep learning for seismic inverse problems: Toward the acceleration of geophysical analysis workflows. **IEEE Signal Processing Magazine**, v. 38, p. 89–119, 3 2021. ISSN 1053-5888. Cited in page 10.

16 GRANA, D.; MUKERJI, T.; DOYEN, P. **Seismic Reservoir Modeling: Theory, Examples, and Algorithms**. [S.l.]: John Wiley & Sons, 2021. Cited in page 10.

17 DOYEN, P. **Seismic Reservoir Characterization: An Earth Modelling Perspective**. [S.l.]: EAGE Publications bv, 2007. Cited 6 times in pages 10, 11, 18, 22, 54, and 104.

18 BOSCH, M.; MUKERJI, T.; GONZALEZ, E. F. Seismic inversion for reservoir properties combining statistical rock physics and geostatistics: A review. **Geophysics**, v. 75, n. 5, 2010. ISSN 00168033. Cited 3 times in pages 10, 11, and 21.

19 AZEVEDO, L.; SOARES, A. **Geostatistical methods for reservoir Geophysics**. [S.l.]: Springer International Publishing, 2017. 1–141 p. ISSN 25093738. ISBN 9783319532004. Cited 8 times in pages 10, 11, 12, 22, 23, 49, 88, and 107.

20 AMADO, L. Chapter 3 - exploration and appraisal phases. In: AMADO, L. (Ed.). **Reservoir Exploration and Appraisal**. Boston: Gulf Professional Publishing, 2013. p. 9–12. ISBN 978-1-85617-853-2. Cited in page 10.

21 Michael J. Pyrcz, C. V. D. **Geostatistic Reservoir Modeling**. [S.l.: s.n.], 2014. ISBN 9780199731442. Cited 2 times in pages 11 and 106.

22 TARANTOLA, A. **Inverse problem theory and Methods for Model Parameter Estimation**. [S.l.: s.n.], 2004. 1–358 p. ISBN 0898715725. Cited 4 times in pages 11, 21, 22, and 40.

23 GONZALEZ, E.; MUKERJI, T.; MAVKO, G. Seismic inversion combining rock physics and multiple-point geostatistics. **Geophysics**, v. 73, 01 2008. Cited in page 12.

24 GRANA, D. et al. Stochastic inversion of facies from seismic data based on sequential simulations and probability perturbation method. **Geophysics**, v. 77, n. 4, p. M53–M72, 2012. Cited in page 12.

25 GRANA, D. et al. Probabilistic inversion of seismic data for reservoir petrophysical characterization: Review and examples. **Geophysics**, v. 87, n. 5, p. M199–M216, 2022. Cited in page 12.

26 DEUTSCH, C. V.; JOURNEL, A. **GSLIB: Geostatistical Software Library and User's Guide**. [S.l.]: Oxford University Press, 1998. (Applied geostatistics series). ISBN 9780195100150. Cited 3 times in pages 12, 13, and 106.

27 FERREIRINHA, T. et al. Acceleration of stochastic seismic inversion in OpenCL-based heterogeneous platforms. **Computers and Geosciences**, Elsevier Ltd, v. 78, p. 26–36, may 2015. ISSN 00983004. Cited in page 12.

28 MOSSER, L. et al. Rapid seismic domain transfer: Seismic velocity inversion and modeling using deep generative neural networks. In: 80TH EAGE CONFERENCE AND EXHIBITION 2018: OPPORTUNITIES PRESENTED BY THE ENERGY TRANSITION. **80th EAGE Conference and Exhibition 2018: Opportunities Presented by the Energy Transition**. [S.l.], 2018. Cited 3 times in pages 12, 46, and 47.

29 SUN, Y.; ARAYA-POLO, M.; WILLIAMSON, P. Data characterization and transfer learning for dl-driven velocity model building. In: . [S.l.]: Society of Exploration Geophysicists, 2021. p. 1475–1479. Cited 2 times in pages 12 and 46.

30 DAS, V. et al. Convolutional neural network for seismic impedance inversion. **Geophysics**, Society of Exploration Geophysicists, v. 84, n. 6, p. R869–R880, nov 2019. ISSN 19422156. Cited 3 times in pages 12, 45, and 46.

31 COLOMBO, D. et al. Deep-learning electromagnetic monitoring coupled to fluid flow simulators. **https://doi.org/10.1190/geo2019-0428.1**, Society of Exploration Geophysicists, v. 85, n. 4, p. WA1–WA12, jan 2020. Cited 2 times in pages 12 and 46.

32 PARK, M. J.; SACCHI, M. D. Automatic velocity analysis using convolutional neural network and transfer learning. **Geophysics**, v. 85, n. 1, p. V33–V43, 2020. Cited 2 times in pages 12 and 46.

33 KARPATNE, A. et al. Theory-guided data science: A new paradigm for scientific discovery from data. **IEEE Transactions on Knowledge and Data Engineering**, v. 29, p. 2318–2331, 2017. Cited 2 times in pages 12 and 46.

34 SUN, J.; INNANEN, K.; HUANG, C. Physics-guided deep learning for seismic inversion with hybrid training and uncertainty analysis. **Geophysics**, v. 86, 01 2021. Cited 3 times in pages 12, 47, and 48.

35 BISWAS, R. et al. Prestack and poststack inversion using a physics-guided convolutional neural network. **Interpretation**, v. 7, p. SE161–SE174, 2019. Cited 2 times in pages 12 and 47.

36 ALFARRAJ, M.; ALREGIB, G. Semi-supervised sequence modeling for elastic impedance inversion. **arXiv**, 2019. Cited 4 times in pages 12, 45, 47, and 48.

37 HU, W. et al. Physics-guided self-supervised learning for low frequency data prediction in fwi. p. 875–879, 2020. Cited 2 times in pages 12 and 47.

38 JIN, P. et al. Cyclefcn: A physics-informed data-driven seismic waveform inversion method. p. 3867–3871, 2020. Cited 2 times in pages 12 and 47.

39 CHEN, H. et al. Optimization-inspired deep learning high-resolution inversion for seismic data. **Geophysics**, v. 86, n. 3, p. R265–R276, 2021. Cited 2 times in pages 12 and 47.

40 FENG, R. et al. An unsupervised deep-learning method for porosity estimation based on poststack seismic data. **Geophysics**, v. 85, p. M97–M105, 2020. Cited 2 times in pages 12 and 47.

41 HüLLERMEIER, E.; WAEGEMAN, W. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. **Machine Learning**, v. 110, p. 457–506, 2021. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-021-05946-3>. Cited in page 12.

42 GAWLIKOWSKI, J. et al. A survey of uncertainty in deep neural networks. 7 2021. Disponível em: <https://arxiv.org/abs/2107.03342>. Cited in page 12.

43 GAL, Y.; GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: **Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48**. [S.l.]: JMLR.org, 2016. (ICML'16), p. 1050–1059. Cited 2 times in pages 12 and 48.

44 CHAN, S.; ELSHEIKH, A. H. **Parametrization and generation of geological models with generative adversarial networks**. [S.l.]: arXiv, 2017. Cited in page 12.

45 DUPONT, E. et al. Generating realistic geology conditioned on physical measurements with generative adversarial networks. **arXiv: Machine Learning**, 2018. Cited in page 12.

46 RICHARDSON, A. Generative adversarial networks for model order reduction in seismic full-waveform inversion. 6 2018. Cited 2 times in pages 12 and 48.

47 LALOY, E. et al. Training-image-based geostatistical inversion using a spatial generative adversarial neural network. **Water Resources Research**, v. 54, 01 2018. Cited 2 times in pages 12 and 48.

48 MOSSER, L.; DUBRULE, O.; BLUNT, M. J. Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. **Mathematical Geosciences**, v. 52, p. 53–79, 2020. ISSN 1874-8953. Cited 2 times in pages 12 and 48.

49 AUSTRALIA, E. I. **Marine seismic surveys: what you need to know - Energy Information Australia**. Cited in page 16.

50 AUSTRALIA, G. **Seismic - Geoscience Australia**. Disponível em: <https://www.ga.gov.au/scientific-topics/disciplines/geophysics/seismic>. Cited in page 16.

51 MUTLAQ, M. H. A.; MARGRAVE, G. F. Tutorial: AVO inversion. **CREWES Research Report**, v. 22, p. 2.1–2.23, 2010. Cited in page 19.

52 ZOEPPRITZ, K. **Uber reflexion und durchgang seismischerwellen durch Unstetigkerlsflaschen**. 1919. Berlin, Uber Erdbebenwellen VII B, Nachrichten der Koniglichen Gesellschaft derWissensschaften zu Gottingen, math-phys. Kl. pp. 57–84. Cited in page 19.

53 SHUEY, R. T. A simplification of the zoeppritz equations. **Geophysics**, v. 50, n. 4, p. 609–614, 1985. Cited 2 times in pages 19 and 59.

54 BOUSFIELD, M. et al. Using base function decomposition in adaptive filtering for seismic pulse retrieving using base function decomposition in adaptive filtering for seismic pulse retrieving. 01 2016. Cited in page 20.

55 COOKE, D. Generalized linear inversion of reflection seismic data. **Geophysics**, v. 48, 06 1983. Cited in page 21.

56 BULAND, A.; OMRE, H. Bayesian linearized AVO inversion. **Geophysics**, Society of Exploration Geophysicists, v. 68, n. 1, p. 185–198, feb 2003. ISSN 00168033. Cited in page 21.

57 LORTZER, G. J.; BERKHOUT, A. J. An integrated approach to lithologic inversion - part I: theory. **Geophysics**, Society of Exploration Geophysicists, v. 57, n. 2, p. 233–244, feb 1992. ISSN 00168033. Cited in page 21.

58 SEN, M. K.; STOFFA, P. L. Nonlinear one-dimensional seismic waveform inversion using simulated annealing. **Geophysics**, Society of Exploration Geophysicists, v. 56, n. 10, p. 1624–1638, feb 1991. ISSN 00168033. Cited in page 21.

59 MA, X. Q. Simultaneous inversion of prestack seismic data for rock properties using simulated annealing. **Geophysics**, Society of Exploration Geophysicists, v. 67, n. 6, p. 1877–1885, nov 2002. ISSN 00168033. Cited in page 21.

60 MALLICK, S. Model-based inversion of amplitude-variations-with-offset data using a genetic algorithm. **Geophysics**, Society of Exploration Geophysicists, v. 60, n. 4, p. 939–954, feb 1995. ISSN 00168033. Cited in page 21.

61 VIRIEUX, J.; OPERTO, S. An overview of full-waveform inversion in exploration geophysics. **Geophysics**, v. 74, n. 6, p. WCC1–WCC26, 2009. Cited in page 21.

62 GUO, Q.; RIVERA, C. Elastic reflection fwi for subsalt velocity reconstruction: Benchmark test with massive salt body. In: . [S.l.: s.n.], 2019. p. 1600–1604. Cited in page 21.

63 GRANA, D.; Della Rossa, E. Probabilistic petrophysical-properties estimation integrating statistical rock physics with seismic inversion. **Geophysics**, Society of Exploration Geophysicists, v. 75, n. 3, may 2010. ISSN 00168033. Cited 2 times in pages 21 and 22.

64 SOARES, A.; DIET, J. D.; GUERREIRO, L. Stochastic inversion with a global perturbation method. In: **Petroleum Geostatistics 2007**. [S.l.: s.n.], 2007. ISBN 9073781485. Cited 3 times in pages 22, 23, and 88.

65 CAETANO, H. **Integration of seismic information in reservoir models: Global Stochastic Inversion**. Tese (Doutorado) — Universidade Técnica de Lisboa, Instituto Superior Técnico, 2009. Cited in page 22.

66 SOARES, A. Direct sequential simulation and cosimulation. **Mathematical Geology**, Kluwer Academic/Plenum Publishers, v. 33, n. 8, p. 911–926, 2001. ISSN 08828121. Cited 5 times in pages 22, 49, 61, 63, and 107.

67 MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943. Cited in page 25.

68 ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, p. 386–408, 1958. ISSN 1939-1471. Cited in page 25.

69 KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. 2017. Cited 2 times in pages 30 and 63.

70 DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of Machine Learning Research**, v. 12, p. 2121–2159, 07 2011. Cited in page 30.

71 TIELEMAN, T.; HINTON, G. **Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude**. [S.l.]: COURSERA: Neural networks for machine learning, 2012. Cited in page 30.

72 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, p. 533–536, 1986. Cited in page 31.

73 ZHAO, X. et al. A surrogate model with data augmentation and deep transfer learning for temperature field prediction of heat source layout. **Structural and Multidisciplinary Optimization**, Springer, v. 64, n. 4, p. 2287–2306, 2021. Cited in page 33.

74 KIM, Y. et al. Deep learning framework for material design space exploration using active transfer learning and data augmentation. **npj Computational Materials**, Nature Publishing Group, v. 7, n. 1, p. 1–7, 2021. Cited in page 33.

75 DESAI, A. D. et al. Vortex: Physics-driven data augmentations for consistency training for robust accelerated mri reconstruction. **arXiv preprint arXiv:2111.02549**, 2021. Cited in page 33.

76 SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 06 2014. Cited in page 33.

77 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. Imagenet classification with deep convolutional neural networks. **Neural Information Processing Systems**, v. 25, 01 2012. Cited in page 34.

78 ZHOU, Y.; CHELLAPPA, R. Computation of optical flow using a neural network. **IEEE 1988 International Conference on Neural Networks**, p. 71–78 vol.2, 1988. Cited in page 36.

79 HE, K. et al. Deep residual learning for image recognition. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 2016-December, p. 770–778, 2016. Cited 4 times in pages 37, 38, 54, and 55.

80 BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on Neural Networks**, v. 5, p. 157–166, 1994. Cited in page 37.

81 BENGIO, Y.; GLOROT, X. Understanding the difficulty of training deep feed forward neural networks. **International Conference on Artificial Intelligence and Statistics**, p. 249–256, jan. 2010. Cited in page 37.

82 RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. nov. 2015. Cited in page 37.

83 SZEGEDY, C. et al. Going deeper with convolutions. **CoRR**, abs/1409.4842, 2014. Cited in page 37.

84 HUANG, G.; LIU, Z.; WEINBERGER, K. Q. Densely connected convolutional networks. **CoRR**, abs/1608.06993, 2016. Cited in page 37.

85 IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: **32nd International Conference on Machine Learning, ICML 2015**. [S.l.]: International Machine Learning Society (IMLS), 2015. v. 1, p. 448–456. ISBN 9781510810587. Cited in page 37.

86 ISOLA, P. et al. Image-to-image translation with conditional adversarial networks. **CoRR**, abs/1611.07004, 2016. Cited 4 times in pages 39, 41, 43, and 45.

87 LU, Y. et al. Sketch-to-image generation using deep contextual completion. **arXiv preprint arXiv:1711.08972**, 2017. Cited in page 39.

88 VINAY, A. et al. Sketch to image using cnn and gan. In: **2019 9th International Conference on Advances in Computing and Communication (ICACC)**. [S.l.: s.n.], 2019. p. 295–299. Cited in page 39.

89 ZHANG, Y. et al. An enhanced gan model for automatic satellite-to-map image conversion. **IEEE Access**, v. 8, p. 176704–176716, 01 2020. Cited in page 39.

90 LIU, C. et al. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: **CVPR**. [S.l.: s.n.], 2019. Cited in page 39.

91 LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. **CoRR**, abs/1411.4038, 2014. Cited 3 times in pages 39, 40, and 42.

92 RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. **CoRR**, abs/1505.04597, 2015. Cited 3 times in pages 39, 42, and 43.

93 NOVIKOV, A. A. et al. Fully convolutional architectures for multi-class segmentation in chest radiographs. **CoRR**, abs/1701.08816, 2017. Cited in page 39.

94 MCCANN, M. T.; JIN, K. H.; UNSER, M. Convolutional neural networks for inverse problems in imaging: A review. **IEEE Signal Processing Magazine**, v. 34, n. 6, p. 85–95, 2017. Cited in page 39.

95 ONGIE, G. et al. Deep learning techniques for inverse problems in imaging. 5 2020. Cited in page 39.

96 SCARLETT, J. et al. Theoretical perspectives on deep learning methods in inverse problems. 6 2022. Cited in page 39.

97 SPRINGENBERG, J. T. et al. **Striving for Simplicity: The All Convolutional Net**. 2015. Cited in page 41.

98 VINCENT, P. et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. **J. Mach. Learn. Res.**, v. 11, p. 3371–3408, 2010. Cited in page 42.

99 FU, J. et al. Scene segmentation with dual relation-aware attention network. **IEEE Transactions on Neural Networks and Learning Systems**, v. 32, n. 6, p. 2547–2560, 2021. Cited in page 42.

100 ZHANG, H. et al. Co-occurrent features in semantic segmentation. In: **2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2019. p. 548–557. Cited in page 42.

101 WANG, Z. et al. Image quality assessment: From error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, p. 600–612, 4 2004. ISSN 1057-7149. Cited 2 times in pages 43 and 63.

102 ZHAO, H. et al. **Loss Functions for Neural Networks for Image Processing**. [S.l.]: arXiv, 2015. Cited in page 44.

103 GATYS, L. A.; ECKER, A. S.; BETHGE, M. A neural algorithm of artistic style. **CoRR**, abs/1508.06576, 2015. Cited in page 44.

104 JOHNSON, J.; ALAHI, A.; FEI-FEI, L. Perceptual losses for real-time style transfer and super-resolution. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 9906 LNCS, p. 694–711, 2016. Cited in page 44.

105 SAJJADI, M. S. M.; SCHöLKOPF, B.; HIRSCH, M. **EnhanceNet: Single Image Super-Resolution Through Automated Texture Synthesis**. [S.l.]: arXiv, 2016. Cited in page 44.

106 ZHANG, R. et al. **The Unreasonable Effectiveness of Deep Features as a Perceptual Metric**. [S.l.]: arXiv, 2018. Cited in page 44.

107 SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv 1409.1556**, 09 2014. Cited in page 44.

108 DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. **2009 IEEE conference on computer vision and pattern recognition**. [S.l.], 2009. p. 248–255. Cited in page 44.

109 RöTH, G.; TARANTOLA, A. Neural networks and inversion of seismic data. **Journal of Geophysical Research: Solid Earth**, v. 99, p. 6753–6768, 1994. Cited in page 45.

110 GHOLAMI, A.; ANSARI, H. R. Estimation of porosity from seismic attributes using a committee model with bat-inspired optimization algorithm. **Journal of Petroleum Science and Engineering**, v. 152, p. 238–249, 2017. ISSN 0920-4105. Cited in page 45.

111 ARAYA-POLO, M. et al. Deep-learning tomography. **The Leading Edge**, v. 37, p. 58–66, 2018. Cited in page 45.

112 ROJAS-GóMEZ, R. et al. Physics-consistent data-driven seismic inversion with adaptive data augmentation. In: . [S.l.: s.n.], 2020. Cited in page 46.

113 REN, Y. et al. A physics-based neural-network way to perform seismic full waveform inversion. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 112266–112277, 2020. ISSN 21693536. Cited in page 47.

114 WANG, Y. et al. Well-Logging Constrained Seismic Inversion Based on Closed-Loop Convolutional Neural Network. **IEEE Transactions on Geoscience and Remote Sensing**, Institute of Electrical and Electronics Engineers Inc., v. 58, n. 8, p. 5564–5574, aug 2020. ISSN 15580644. Cited in page 47.

115 ZHU, J.-Y. et al. Toward multimodal image-to-image translation. **CoRR**, abs/1711.11586, 2017. Cited in page 47.

116 WIATRAK, M.; ALBRECHT, S. V.; NYSTROM, A. Stabilizing generative adversarial networks: A survey. 9 2019. Cited in page 48.

117 KENNETT, B. **Seismic Wave Propagation in Stratified Media**. [S.l.]: ANU E Press, 2009. (DOAB Directory of Open Access Books). ISBN 9781921536731. Cited in page 55.

118 MAVKO, G.; MUKERJI, T.; DVORKIN, J. Copyright page. In: _____. **The Rock Physics Handbook**. 3. ed. [S.l.]: Cambridge University Press, 2020. p. iv–iv. Cited in page 59.

119 AZEVEDO, L. et al. Geostatistical seismic inversion with self-updating of local probability distributions. **Mathematical Geosciences**, v. 53, p. 1073–1093, 2021. ISSN 1874-8953. Cited in page 89.

120 KRUSKAL, J. B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. **Psychometrika**, v. 29, p. 1–27, 1964. ISSN 1860-0980. Cited 2 times in pages 89 and 93.

121 AZEVEDO, L. et al. Multidimensional scaling for the evaluation of a geostatistical seismic elastic inversion methodology. **Geophysics**, v. 79, p. M1–M10, 2014. Cited in page 89.

122 KRIGE, D. G. **A statistical approach to some mine valuation and allied problems on the Witwatersrand**. Dissertação (Mestrado) — M. Sc. in Engineering University of the Witwatersrand, 1951. Cited in page 104.

123 ALABERT, F. **Stochastic Imaging of Spatial Distributions Using Hard and Soft Information**. Dissertação (Mestrado) — Stanford University, 1987. Cited in page 106.

124 ISAAKS, E. H. **The application of Monte Carlo methods to the analysis of spatially correlated data.** Tese (Doutorado) — Stanford University, 1990. Cited in page 106.

# A
# Geostatistics

Geostatistics is defined as a set of statistical methods for describing random variables (RVs) distributed in space (i.e., regionalized variables) that are spatially correlated. The set of all these RVs defined over some field of interest ($A \subset \mathbb{R}^3$) is known as a random function (RF), which is typically denoted by $Z(\mathbf{u})$, where **u** is a localization vector inside the field.

Geostatistical spatial interpolation is a well-established tool to interpolate a reservoir's internal properties of interest between wells [17]. Kriging is the most known geostatistical algorithm for estimating unobserved locations given nearby observations. The method was developed by Georges Matheron in 1960 based on the seminal work [122] of Daniel Krige in mining engineering. Geostatistics relies on the decision of stationarity to provide the best linear unbiased estimate for an unsampled location **u**.

A fundamental assumption of geostatistical methods is the second order (i.e., two-point) stationarity of a RF $Z(\mathbf{u})$. By considering the first two moments of the spatial law, a RF $Z(\mathbf{u})$ is said to be second-order stationarity if:

1. The expectation $\mathbf{E}[Z(\mathbf{u})]$ is constant over the entire field domain, which means that they do not depend on the location **u**:

$$\mathbf{E}[Z(\mathbf{u})] = \mu, \forall \mathbf{u} \in A \tag{A-1}$$

2. The covariance between a pair of RVs, $C[(Z(\mathbf{u}), Z(\mathbf{u}+\mathbf{h})]$, only relies on the separation vector **h**:

$$C[(Z(\mathbf{u}), Z(\mathbf{u}+\mathbf{h})] = C(\mathbf{h}), \forall \mathbf{u} \in A \tag{A-2}$$

The stationarity of the covariance implies the stationarity of the variance:

$$\begin{aligned} C(0) =& \mathbf{E}[Z(\mathbf{u}) \cdot Z(\mathbf{u}+0)] - \mu^2, \\ =& \mathbf{E}[Z(\mathbf{u})^2] - \mu^2, \\ =& \mathrm{Var}[Z(\mathbf{u})] = \sigma^2, \forall \mathbf{u} \in A, \end{aligned} \tag{A-3}$$

The theory behind geostatistical estimation and simulation requires the use of the variogram or covariance. The variogram, or semi-variogram, is another second-order moment considered in geostatistics, defined as the half of the variance of the increment in a given direction:

$$\begin{aligned} \gamma(h) =& \frac{1}{2}\mathrm{Var}[Z(\mathbf{u}) - Z(\mathbf{u}+\mathbf{h})], \\ 2\gamma(\mathbf{h}) =& \mathrm{Var}[Z(\mathbf{u}) - Z(\mathbf{u}+\mathbf{h})], \\ =& \mathbf{E}\Big[\Big(Z(\mathbf{u}) - Z(\mathbf{u}+\mathbf{h})\Big)^2\Big], \\ =& \frac{1}{N(h)} \sum_{i=1}^{N(h)} \Big[(Z(\mathbf{u}) - Z(\mathbf{u}+\mathbf{h})\Big]^2 \end{aligned} \tag{A-4}$$

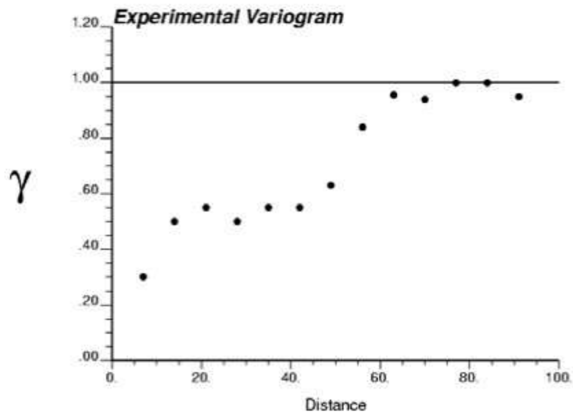


Figure A.1: Example of an experimental variogram.

where $N(\mathbf{h})$ is the number of pairs of observations with the distance **h** within the sample data. The experimental variogram is plotted as a two-dimensional graph (Figure A.1) to describe the spatial variability between all pairs of points at several distances.

Under the stationarity assumptions, it is possible to derive that the covariance and variogram are equivalent:

$$C(\mathbf{h}) = C(0) - \gamma(\mathbf{h}) \qquad \text{(A-5)}$$

## A.1 Mean Values Estimates

Based on nearby observations $(Z(\mathbf{u}_\alpha), \alpha = 1, ..., N)$, kriging gives the best unbiased linear estimate of an unobserved point $(Z^*(\mathbf{u}_0)$. The general form of the Kriging estimator is:

$$Z^*(\mathbf{u}_0) = \mu(\mathbf{u}) + \sum_{\alpha=1}^{N} \lambda_\alpha \big(Z(\mathbf{u}_\alpha) - \mu(\mathbf{u}_\alpha)\big) \qquad \text{(A-6)}$$

Simple Kriging (SK) requires knowledge of the stationary mean $(\mathbf{E}[Z(\mathbf{u})] = \mu, \forall \mathbf{u} \in A)$ and the residual between the unobserved point and its mean is predicted based on the residuals between the $N$ samples:

$$Z^*(\mathbf{u}) - \mu = \sum_{\alpha=1}^{N} \lambda_\alpha \big(Z(\mathbf{u}_\alpha) - \mu\big) \qquad \text{(A-7)}$$

A linear system of equations is devised by posing the problem as an optimization problem subject to minimizing the estimation variance. The weights $\lambda_\alpha$ are obtained by solving this system of $N$ equations with $N$ unknown weights, known as the Simple Kriging (SK) system:

$$\sum_{\alpha=1}^{N} \lambda_\alpha C(\mathbf{u}_\alpha - \mathbf{u}_\beta) = C(\mathbf{u} - \mathbf{u}_\beta), \mathbf{u}_\beta = 1, ..., N \qquad \text{(A-8)}$$

The covariance values are retrieved from the variogram model, a positive definite model of spatial variability valid within all the field A. As the experimental variogram cannot be used directly, an appropriate variogram model is chosen by matching the shape of the theoretical curve to the experimental calculated points.

The variogram model ensures that the variance will always be positive or equal to zero, and thus guarantees the existence of a solution for the SK system.

## A.2
## Stochastic Simulation

Stochastic simulation is the process of generating multiple equally probable solutions (aka as realizations) for a random variable within a random function model [26]. The probability distribution function obtained from the set of realizations at several locations allows the assessment of uncertainty about the spatial distribution of the attribute being modeled.

As an estimation method, Simple Kriging provides the most likely value at each location and creates an estimate that is often too smooth. This smoothing is directly proportional to the kriging variance:

$$\begin{aligned} \text{Var}[Z^*(\mathbf{u})] &= C(0) - \sigma^2_{SK}(\mathbf{u}), \\ \sigma^2_{SK}(\mathbf{u}) &= \sum_{\alpha=1}^{n} \lambda_\alpha C(\mathbf{u} - \mathbf{u}_\alpha) \end{aligned} \tag{A-9}$$

where $\sigma^2_{SK}(\mathbf{u})$ is the Kriging variance. Because of the overestimation of low values and underestimation of high values, the samples histogram and spatial variability are not reproduced.

Conversely, geostatistical stochastic simulation methods calculate a good/reasonable estimate at each location and are able to reproduce the actual histogram and spatial variability (i.e., variogram) of the samples. Among several algorithms that can be devised to create stochastic simulations, the Sequential Gaussian Simulation (SGS) [123, 124] is one of the most commonly used in reservoir modeling applications [21]. The algorithm is based on a recursive decomposition of the joint probability, $p$, of a set of M random variables, $Z$, and N conditioning data:

$$\begin{aligned} &p(z_{N+1} \le Z_{N+1}, \cdots, z_{N+M} \le Z_{N+M} \mid z_1, \cdots, z_n) = \\ &\quad p(z_{N+1} \le Z_{N+1} \mid z_1, \cdots, z_n) \cdot p(z_{N+2} \le Z_{N+2} \mid z_1, \cdots, z_n, z_{N+1}) \cdot \\ &\qquad \cdots \\ &\qquad \cdot\, p(z_{N+M} \le Z_{N+M} \mid z_1, \cdots, z_n, z_{N+1}, \cdots, z_{N+M-1}) \end{aligned}$$

As a Gaussian-based simulation method, SGS works with normal scores of the original data and thus requires a prior normal score transformation step. The simulation grid is visited along a random path that visits all the cells of the simulation grid. At each location, the kriging estimate, and kriging variance are computed based on direct observations (i.e., known data points) and previously simulated grid cells.

The simple kriging estimate $Z^*_{SK}(\mathbf{u})$ and kriging variance $\sigma^2_{SK}(\mathbf{u})$ are used to define the parameters of the distribution of $Z^*(\mathbf{u}) \sim \mathcal{N}(Z^*_{SK}(\mathbf{u}), \sigma^2_{SK}(\mathbf{u}))$. Monte Carlo simulation of a realization $Z^*(\mathbf{u})$ is obtained by drawing a uniform random number $p^{(l)} \in [0,1]$ and retrieving the ccdf) (complementary cumulative distribution function) $p^{(l)}$ quantile.

The simulated value is then assigned as conditioning data for the next location along the random path. As the random path changes each time the simulation runs, the conditioning data at each location changes and therefore results in alternative models. Ultimately, the simulated values are then transformed back into their original distribution space.

Under the multi-Gaussian assumption, SGS suffers from some limitations in reproducing highly skewed or multimodal distributions [19]. The direct sequential simulation (DSS) method [66] attempts to overcome this problem by direct sampling from an auxiliary probability distribution function built from the global conditional distribution function as predicted from the experimental data.

# B
# Complementary Results

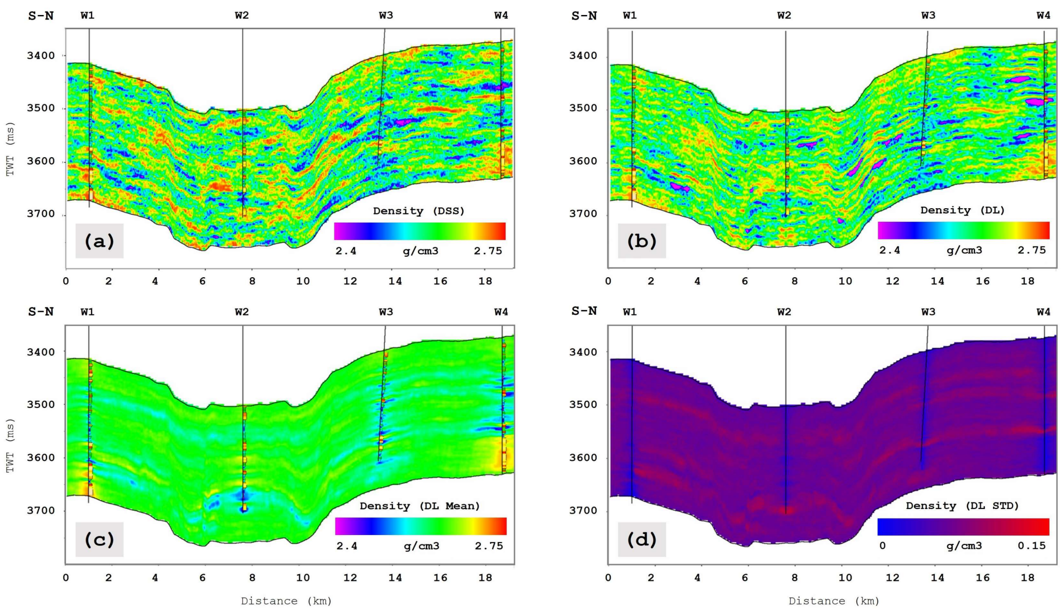


Figure B.1: Vertical well sections extracted from: **(a)** a DSS density model, **(b)** inverted density model from (a), **(c)** point-wise average, and **(d)** point-wise standard deviation models computed from the ensemble of inverted density models.

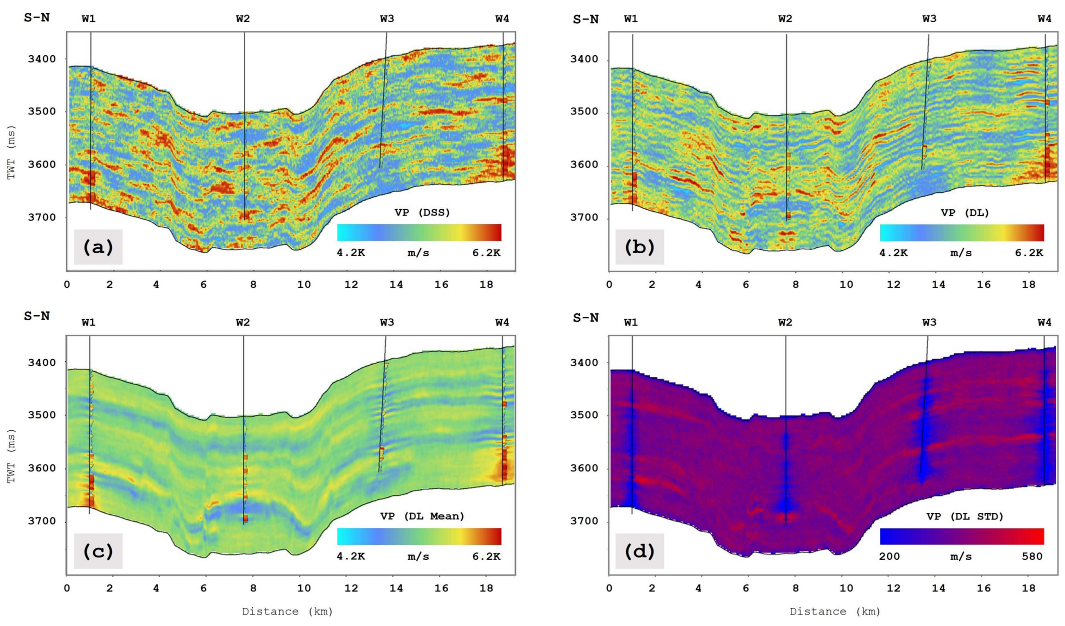


Figure B.2: Vertical well sections extracted from: **(a)** a DSS S-wave velocity ($V_P$) model, **(b)** inverted S-wave velocity model from (a), **(c)** point-wise average, and **(d)** point-wise standard deviation models computed from the ensemble of inverted S-wave velocity models.

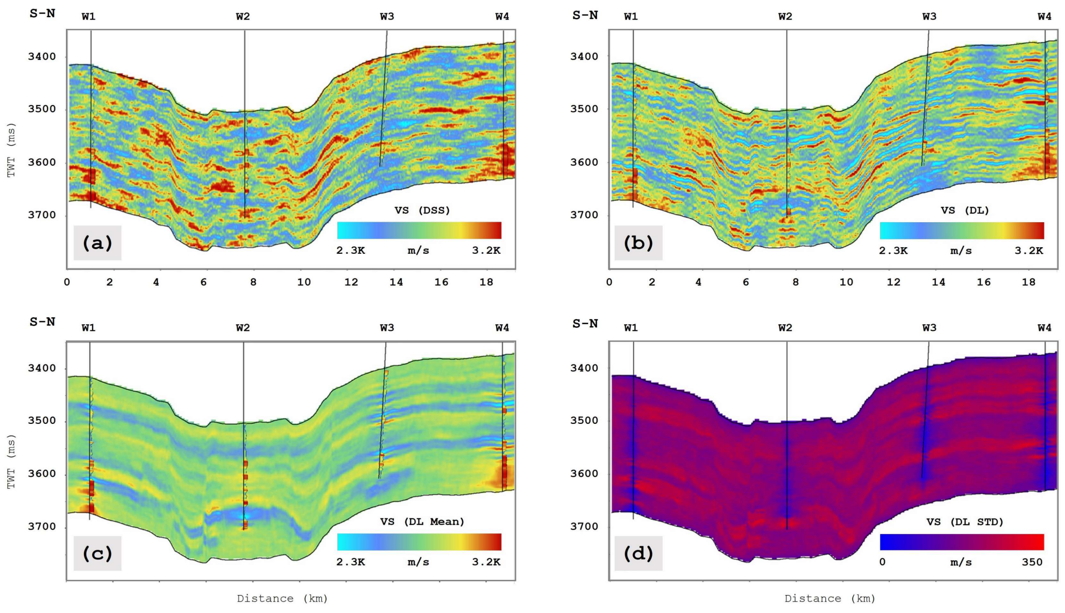


Figure B.3: Vertical well sections extracted from: **(a)** a DSS S-wave velocity ($V_S$) model, **(b)** inverted S-wave velocity model from (a), **(c)** point-wise average, and **(d)** point-wise standard deviation models computed from the ensemble of inverted S-wave velocity models.

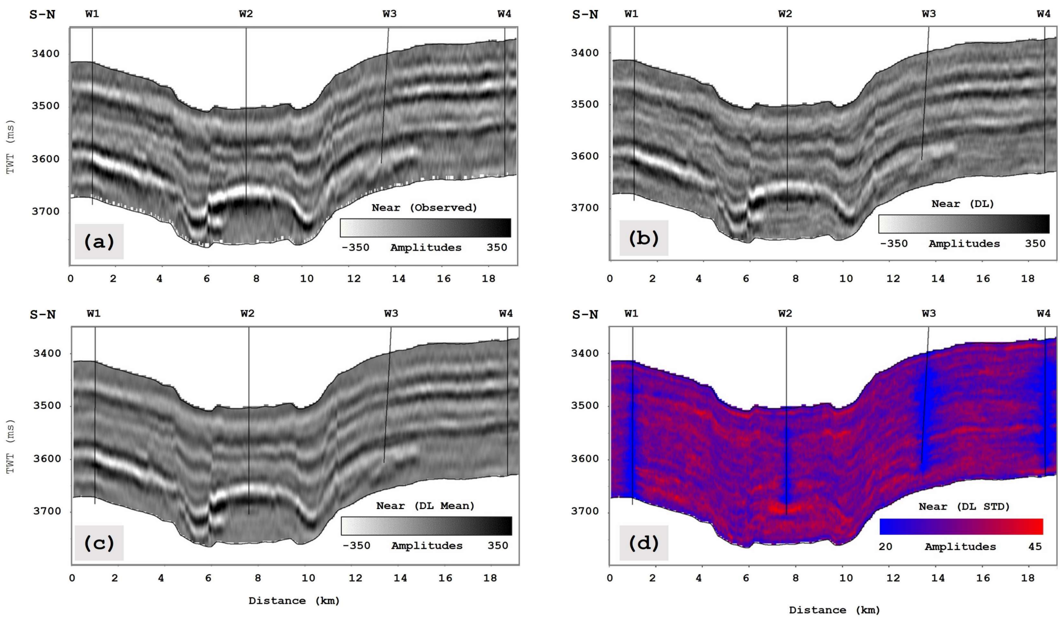


Figure B.4: Vertical well sections of partial stacks of near offsets ($0° - 10°$ range angle) extracted from: **(a)** observed seismic data, **(d)** synthetic seismic calculated from the inverted elastic models, **(c)** point-wise average, and **(d)** point-wise standard deviation models computed from the synthetic seismic ensemble.

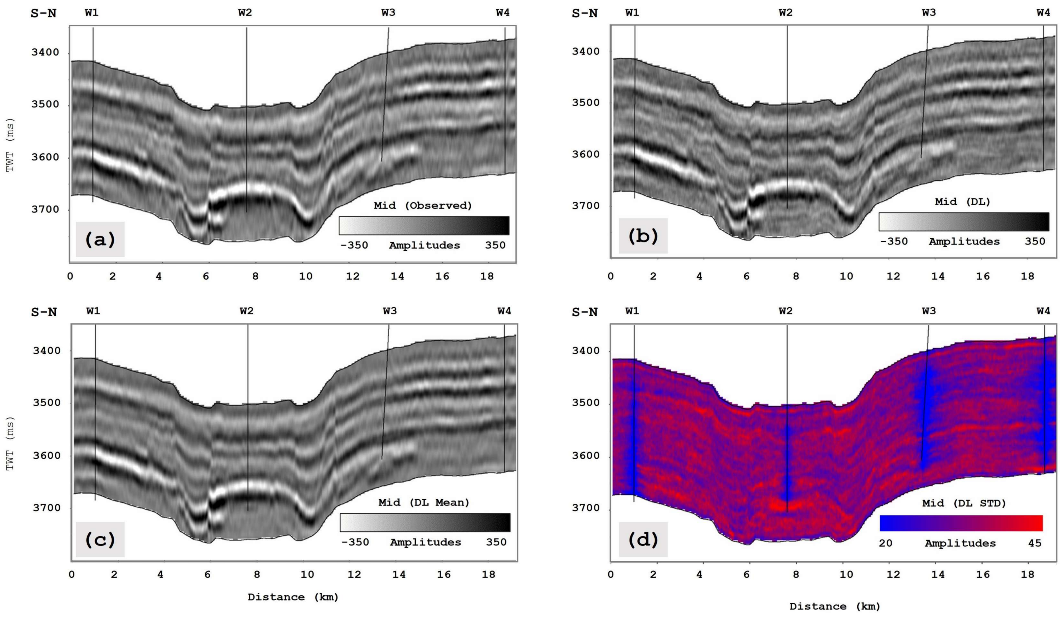


Figure B.5: Vertical well sections of partial stacks of mid offsets ($10° - 20°$ range angle) extracted from: **(a)** observed seismic data, **(d)** synthetic seismic calculated from the triplet of inverted elastic models, **(c)** point-wise average, and **(d)** point-wise standard deviation models computed from the ensemble of synthetic seismic data.

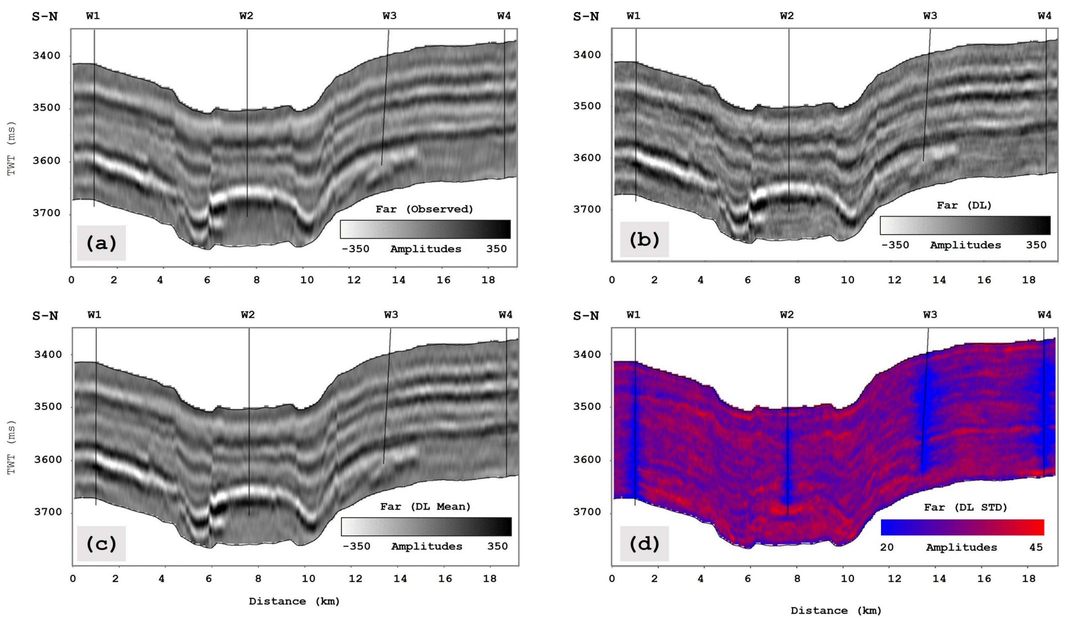


Figure B.6: Vertical well sections of partial stacks of **far** offsets ($20^\circ - 30^\circ$ range angle) extracted from: **(a)** observed seismic data, **(d)** synthetic seismic calculated from the triplet of inverted elastic models, **(c)** point-wise average, and **(d)** point-wise standard deviation models computed from the synthetic seismic ensemble.

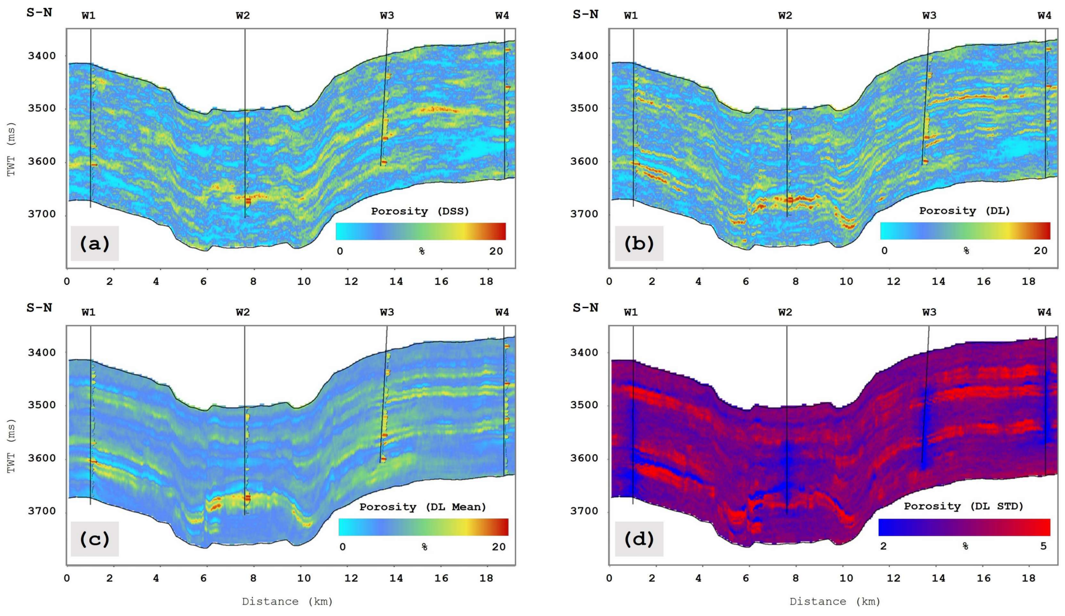


Figure B.7: Vertical well sections extracted from: **(a)** a DSS porosity model, **(b)** inverted porosity model from a, **(c)** point-wise average and **(d)** point-wise standard deviation models computed from the ensemble of inverted porosity models.

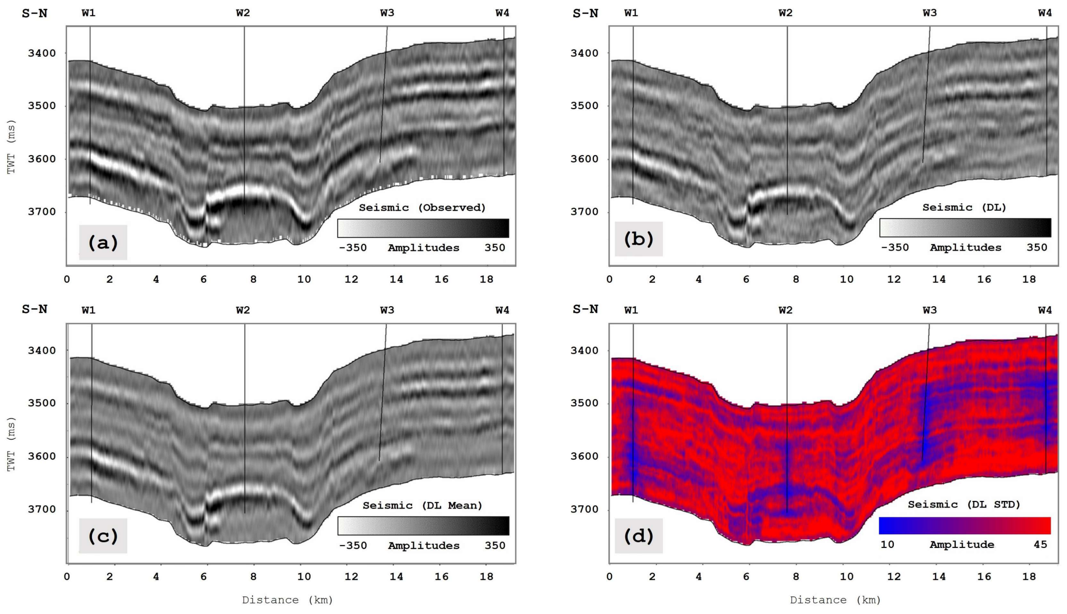


Figure B.8: Vertical well section extracted from: **(a)** observed seismic data; **(d)** synthetic seismic computed from the AI model predicted from the inverted porosity model; **(c)** point-wise average and **(d)** point-wise standard deviation models computed from the ensemble of synthetic seismic data.