Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Carlos Gomes de Carvalho Junior

Synthetic Data Generation and Augmentation for Data-Centric Deep Learning Seismic Inversion

São Carlos
2025

Carlos Gomes de Carvalho Junior

Synthetic Data Generation and Augmentation for Data-Centric Deep Learning Seismic Inversion

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Aprendizado de Máquina

Orientador: Hermes Senger

São Carlos
2025

Junior, Carlos Gomes de Carvalho

Synthetic data generation and augmentation for data-centric deep learning seismic inversion / Carlos Gomes de Carvalho Junior -- 2025.
89f.

Dissertação (Mestrado) - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador (a): Hermes Senger
Banca Examinadora: Hermes Senger, Alan Demétrius Baria Valejo, João Medeiros de Araújo
Bibliografia

1. Aumento de dados. 2. Inversão sísmica. 3. Deep learning. I. Junior, Carlos Gomes de Carvalho. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Arildo Martins - CRB/8 7180

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Carlos Gomes de Carvalho Junior, realizada em 03/10/2025.

Comissão Julgadora:

Prof. Dr. Hermes Senger (UFSCar)

Prof. Dr. Alan Demétrius Baria Valejo (UFSCar)

Prof. Dr. João Medeiros de Araújo (UFRN)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Dedico este trabalho aos meus pais.

Agradecimentos

Primeiramente agradeço a Deus por permitir que tudo acontecesse.

Agradeço também aos meus pais e à minha irmã, por todos os ensinamentos e por estarem sempre presentes.

Agradeço aos meus colegas de laboratório pela mútua troca de conhecimentos, em especial ao Lucas e ao Stevan.

Agradeço ao meu orientador, Prof. Dr. Hermes Senger, pela oportunidade de participar deste projeto e por toda orientação durante o trajeto.

Agradeço às agências de fomento que possibilitaram a realização do trabalho:

- à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001;
- à Financiadora de Estudos e Projetos (FINEP) pelo apoio financeiro por meio do contrato de pesquisa nº 01.23.0575.00 (Projeto Fast-FWI);
- ao RCGI – Research Centre for Greenhouse Gas Innovation (23.1.8493.1.9), sediado na Universidade de São Paulo (USP), financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP 2020/15230-5) e TotalEnergies.

"Dedication is a talent all on its own"
(Alphonse Elric)

Resumo

A Inversão por Aprendizado Profundo (DLI) desponta como uma alternativa promissora e computacionalmente eficiente à inversão sísmica tradicional. No entanto, sua adoção prática enfrenta um obstáculo central: a escassez de conjuntos de dados de treinamento amplos e diversificados. A literatura em DLI tem se concentrado majoritariamente no desenvolvimento de novas arquiteturas de redes neurais, relegando a um papel secundário as estratégias de treinamento orientadas por dados. Esta dissertação aborda tal lacuna ao investigar, de forma sistemática, o potencial de estratégias baseadas em dados sintéticos para aprimorar fluxos de trabalho em DLI.

Inicialmente, estabelecem-se as capacidades fundamentais de síntese de dados, com destaque para a geração de modelos de velocidade de alta fidelidade por meio de Modelos Probabilísticos de Difusão com Remoção de Ruído, bem como para a "engenharia" semanticamente controlável de características geológicas via Autoencoders Variacionais. Em seguida, é conduzido um estudo comparativo abrangente, no qual múltiplas arquiteturas de DLI são avaliadas quanto ao impacto de diferentes técnicas de aumento de dados ,desde métodos simples de interpolação até abordagens avançadas de difusão generativa.

Os resultados evidenciam que o aumento de dados pode promover ganhos expressivos de desempenho, revelando que estratégias mais complexas nem sempre superam alternativas mais simples, e que a escolha ótima depende da arquitetura do modelo em uso. A aplicabilidade prática do trabalho é demonstrada pelo desenvolvimento de uma solução estado-da-arte de DLI que, ao explorar um conjunto massivo de dados sintéticos, figurando entre as 1% melhores soluções em uma competição internacional de inversão sísmica.

Assim, esta dissertação contribui ao destacar o papel estratégico dos dados sintéticos, e ao defender que uma abordagem centrada em dados é capaz de trazer ganhos significativos para DLI.

Palavras-chave: modelos gerativos, dados sintéticos, modelos de velocidade, inversão sísmica, aumento de dados.

Abstract

Deep Learning Inversion (DLI) has emerged as a promising and computationally efficient alternative to traditional seismic inversion. However, its practical adoption is limited by the scarcity of large and diverse training datasets. Existing research in DLI has primarily emphasized the design of novel network architectures, while largely overlooking the potential of data-centric training strategies. This dissertation addresses this gap by systematically evaluating the effectiveness of synthetic data strategies for enhancing DLI workflows.

We first establish the fundamental capabilities of data synthesis, demonstrating the generation of high-fidelity velocity models using Denoising Diffusion Probabilistic Models and exploring semantically controllable geological feature engineering through Variational Autoencoders. Building on these foundations, we conduct a comprehensive comparative study that empirically assesses the impact of a spectrum of data augmentation techniques (ranging from simple interpolation to advanced generative diffusion) across multiple DLI architectures.

The results show that data augmentation consistently improves performance, while also revealing that more complex strategies do not always outperform simpler ones. Importantly, the choice of augmentation method is closely tied to the underlying model architecture, highlighting a critical cost-benefit trade-off in data strategy design. The practical relevance of this research is further demonstrated through the development of a state-of-the-art DLI solution that leveraged a massive-scale synthetic dataset to secure a top 1% ranking in an international seismic inversion competition.

This dissertation contributes to advancing data-centric approaches for seismic inversion, underscoring that the strategic use of synthetic data is essential to fully unlock the potential of DLI.

Keywords: generative models, synthetic data, velocity models, seismic inversion, data augmentation.

List of Figures

List of Tables

Glossary

AI Artificial Intelligence

CNN Convolutional Neural Network

CV Computer Vision

CRF Conditional Random Field

DDPM Denoising Diffusion Probabilistic Models

DL Deep Learning

DLI Deep Learning Inversion

EMA Exponential Moving Average

FWI Full Waveform Inversion

GAN Generative Adversarial Network

GWI Geophysical Waveform Inversion

GRU Gated Recurrent Unit

LSTM Long Short Term Memory

MSE Mean Squared Error

MAE Mean Absolute Error

MLP Multilayer Perceptron

PDE Partial Differential Equation

RNN Recurrent Neural Network

SCSE Spatial and Channel Squeeze and Excitation

SSIM Structural Similarity

VAE Variational Autoencoder

Contents

Chapter 1

Introduction

The accurate representation of subsurface structures through earth models is crucial for numerous geophysical applications. In a resource exploration setting, for example, subsurface mapping provides vital information for locating oil, gas, and other natural resources. However, the standard process of seismic data acquisition —via seismic surveys —is an expensive procedure (LIN et al., 2019), and transforming raw seismic data into earth models (seismic inversion) further requires computationally intensive processing steps. Currently, the most popular approach for producing these high-resolution models of the subsurface has been Full Waveform Inversion (FWI), a powerful physics-based technique that iteratively refines a model to match observed seismic data. However, the power of FWI is constrained by significant limitations (VIRIEUX; OPERTO, 2009). The method is computationally expensive, often requiring weeks or months of supercomputer time to process a single seismic survey, as it involves repeatedly solving complex partial differential equations. Furthermore, FWI is a notoriously ill-posed inverse problem, making its solutions highly sensitive to noise, and to the quality of the initial model used to start the process, with a suboptimal starting point leading to local minima.

The recent success of Deep Learning (DL) in a variety of fields has prompted its application to the seismic domain, and Deep Learning Inversion (DLI) has emerged as a promising alternative to traditional physics-based methods. DLI leverages deep neural networks to learn a direct mapping from seismic waveforms to subsurface velocity models. This data-driven approach offers a paradigm shift in efficiency; once a network is trained, generating a new model is orders of magnitude faster than a full FWI run, and it can bypass the need for an initial model. On the other hand, this advantage comes with a critical dependency on the existence of high quality, diverse training sets. The performance and generalizability of DLI methods are effectively tied to the volume and

quality of the data they are trained on. In geophysics, where high-quality, labeled field data is scarce and expensive to acquire (OVCHARENKO et al., 2019; REN et al., 2021), this data dependency presents an important challenge.

As an alternative to working with velocity models derived from real data, researchers often resort to synthetic data. The traditional way to synthetically create earth models is to use domain experts to craft models with a high degree of realism, often mimicking characteristics found in real world structures (REN et al., 2021). A different approach is to devise a framework or algorithm capable of producing earth models given an initial set of parameters, which has the benefit of being highly scalable, especially when compared with the traditional method. A third way, when sufficient velocity samples are available, is to use DL itself to generate the data.

DL has proven capable of achieving state of the art performance in many Computer Vision (CV) tasks. Particularly, generative models have demonstrated the ability to produce high-quality synthetic data that can mimic real-world scenarios (VAHDAT; KAUTZ, 2020; CHEN et al., 2018). This opens up new possibilities for generating earth models that can be used to supplement existing seismic datasets, improving traditional and deep learning-based inversion techniques.

While the potential of DLI is widely recognized, the majority of research in the field has focused on proposing novel network architectures to solve the inversion problem (WU; LIN, 2018; ZHU et al., 2023; ZHANG; LIN, 2020; ADLER; ARAYA-POLO; POGGIO, 2021). There has been a comparative lack of attention on a question of equal, if not greater, practical importance: how to effectively train these methods. This discrepancy is significant because, as has been demonstrated in other domains like CV, training strategies can make a significant difference in the practical performance of a model (BELLO et al., 2021; WIGHTMAN; TOUVRON; JÉGOU, 2021). In particular, data strategies such as data augmentation, have proven to be extremely effective, enabling networks to reach state-of-the art performance by improving robustness and reducing overfitting (HALEVY; NORVIG; PEREIRA, 2009; SUN et al., 2017).

In a general manner, this research aims to fill the gap in DLI works, that often overlook the usage of synthetic data strategies. The Specific Objectives (SOs) are:

- ❑ **SO1 - Create high-fidelity velocity models using diffusion** (Chapter 4): Demonstrate the capability for generating high-fidelity and diverse synthetic velocity models using Denoising Diffusion Probabilistic Models (DDPM).

- ❑ **SO2 - Create a method to semantically alter velocity models** (Chapter 5): Explore a new avenue for velocity model generation via the manipulation of geological features in existing velocity models .

- ❑ **SO3 - Evaluate data augmentation for seismic inversion** (Chapter 6): Conduct a comparative study to measure the impact of data augmentation techniques

on the performance of multiple DLI architectures under a scenario of limited computational resources.

- ❑ **SO4 - Construct a state-of-the-art DLI solution leveraging synthetic data** (Chapter 7): Use synthetic data techniques applied in the context of Kaggle's Geophysical Waveform Inversion (GWI) competition.

A review of the important theoretical concepts and related work is given in Chapters 2 and 3, respectively.

Chapter 2

Theoretical background

2.1 Velocity models

Earth models aim to provide a detailed representation of a section of the Earth's subsurface, encompassing its physical properties, structural features and geological formations. One way to achieve that is by making use of characteristics that are specific to certain kinds of material (such as velocity, density and reflectivity). With that distinction, it's possible to create visual representations of the underground structure.

Velocity models —earth models that use velocity as the main distinguishing factor —are the main concern of this research. They are used in a wide variety of geophysical applications, with the most notable being resource exploration, where a detailed map of the subsurface is crucial for informed decision making.

They are normally obtained by an initial exploration of the area of interest in the form of a seismic survey, followed by a computationally intensive data processing step. Seismic surveys can be conducted both in land and offshore, and are high cost operations that require specialized equipment, personnel and government authorization. At the same time, processing workflows can take weeks or months of supercomputer time, much of which is spent on inversion algorithms such as Full Waveform Inversion (FWI).

2.2 Full Waveform Inversion

A well-established technique to produce velocity models is through FWI (TARANTOLA, 1984), which is capable of constructing an earth model by minimizing the misfit between an initial guess and collected seismic data. The process is normally framed as two problems. The "forward problem", which studies the propagation of seismic waves

given a seismic source and the geological media; and the "inverse problem", that tries to reconstruct information about the geological media from discrete observations of seismic waves (CHEN; LEE, 2015).

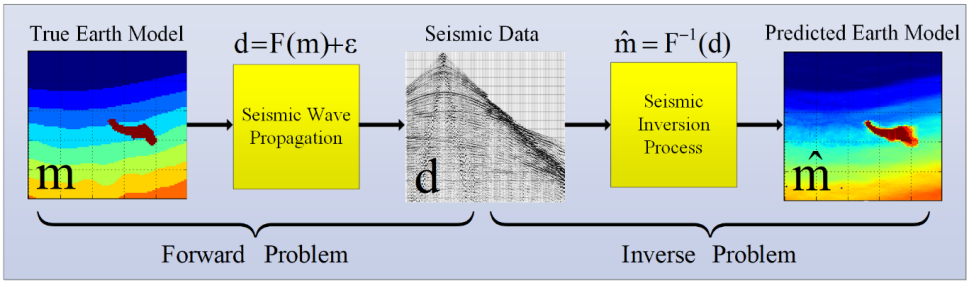

Figure 1 – Forward and Inverse problems. Source: (ADLER; ARAYA-POLO; POGGIO, 2021)

FWI is capable of generating high-resolution models of the subsurface, revealing important properties of the underground materials, such as density, anisotropy, attenuation and wave propagation speed. Such information is invaluable for making informed decisions on any task related to the underground.

At first, an initial earth model is created to represent the geological media and is used to solve the forward problem, producing synthetic observations (e.g. seismograms). Then, the synthetic observations are compared to the real observations (field data) and the differences serve as a way to correct the initial model. The corrected model can then be used as the starting model for the next iteration. This process is repeated until the discrepancy between the synthetic and real models are within a desired margin of error.

Much of the computational complexity of FWI comes from solving the forward problem. This process involves finding the numerical solution for Partial Differential Equations (PDEs) that model the propagation of waves in multi-layer subsurface materials. Many equations can be chosen, one of them being the second order acoustic wave equation with constant density:

$$\frac{1}{v_p^2}\frac{\partial^2 p(\boldsymbol{x},t)}{\partial t^2} - \nabla^2 p(\boldsymbol{x},t) = f(\boldsymbol{x},t), \tag{1}$$

Where v_p is the velocity of the pressure wave field, $f(\boldsymbol{x},t)$ is the source term and $p(\boldsymbol{x},t)$ is the pressure wave field parametrized by $\boldsymbol{x}$ and t, with $\boldsymbol{x}$ being the spatial coordinates and t the time.

To obtain a numerical solution, the PDEs need to be discretized using a method like finite differences, finite elements, finite volumes, etc. with finite differences with a regular grid being the most broadly used in industry.

In the time domain, and using matrix notation, the forward problem can be formulated as (VIRIEUX; OPERTO, 2009):

$$\boldsymbol{M}(\boldsymbol{x})\frac{d^2\boldsymbol{u}(\boldsymbol{x},t)}{dt^2} = \boldsymbol{A}(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x},t) + \boldsymbol{s}(\boldsymbol{x},t), \tag{2}$$

Where $\boldsymbol{M}$ and $\boldsymbol{A}$ are the matrices of mass and stiffness, $\boldsymbol{s}$ is the source, $\boldsymbol{u}$ is the seismic wave field, $\boldsymbol{x}$ are the spatial coordinates and t is the time. The next step is to minimize the misfit of the l_2-norm between the waves propagated through a proposed model and the observed data:

$$\min_{m} f(\boldsymbol{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \left\| \boldsymbol{d}_i^{pred}(\boldsymbol{m}, \boldsymbol{q}_i) - \boldsymbol{d}_i^{obs} \right\|_2^2, \tag{3}$$

Where $f(\boldsymbol{m})$ is the objective as a function of the proposed subsurface model $\boldsymbol{m}$. The index i runs over the total number of receivers n_s. The predicted data $\boldsymbol{d}_i^{pred}$ is represented by solving the following wave equation:

$$\begin{aligned} \boldsymbol{d}_i^{pred}(\boldsymbol{m}, \boldsymbol{q}_i) &= \boldsymbol{P_r} \boldsymbol{u}(\boldsymbol{m}) \\ \boldsymbol{u}(\boldsymbol{m}) &= \boldsymbol{A}(\boldsymbol{m})^{-1} \boldsymbol{P}_s^{\intercal} q_i, \end{aligned} \tag{4}$$

Where the matrix $\boldsymbol{A(m)}$ represents the discretized wave equation parameterized by the vector of the unknown velocity model $\boldsymbol{m}$ to be determined. The vector q_i is the time-dependent distribution of sources for the i-th trigger record. This source function is injected into the discretized grid by the adjoint (denoted by $\intercal$) of the constraint operator P_s. This last operator restricts the wave field to the place of origin. After applying the inverse of the wave operator, we simulate the observed data by restricting the wave field to the receiver locations using the constraint operator $\boldsymbol{P_r}$.

The term $\boldsymbol{d}_i^{pred} = \boldsymbol{F(m, q_i)}$ in Eq. 3 simulates a given model m_i for the velocities of wave propagation and a set of signal sources q_i. This term must be calculated for each receiver (usually in the order of hundreds) and repeated an enormous amount of times in a process that seeks to minimize the error between the captured real data and the simulated data. In other words, the calculation of $\boldsymbol{d}_i^{pred}$ (the forward problem) is the dominant term in the FWI process.

Apart from the high computational cost, FWI also suffers from being an ill-posed problem, with high sensitivity to noise and the starting model (VIRIEUX; OPERTO, 2009).

2.3 Deep Learning Inversion

DL is a subfield of Artificial Intelligence (AI) concerned with using deep neural networks to perform tasks without explicit programming. This data-driven approach has achieved state of the art performance in many fields, like computer vision(ZONG; SONG; LIU, 2023; FANG et al., 2023), natural language processing(YANG et al., 2019; KARL; SCHERP, 2023), speech recognition (CHUNG et al., 2021; BAEVSKI et al., 2020), etc. Among the most common components of architectures for DL are fully connected (also called MLP), convolutional, and attention layers, with each establishing a different relationship between inputs and outputs. In the past decade, researchers have applied DL to

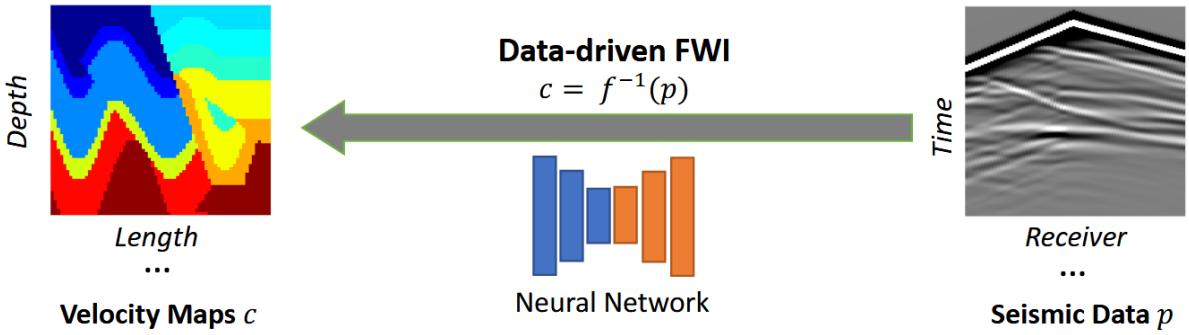

Figure 2 – Deep Learning Inversion consists of using a neural network to learn the inverse mapping between observed seismic data and the velocity map that generated such signals. Source: (DENG et al., 2023)

the seismic inverse problem with encouraging results(ADLER; ARAYA-POLO; POGGIO, 2021).

Most approaches either try to use DL to replace the entire FWI workflow in what's called DLI or to improve one of its aspects, such as performance or starting model estimation in a hybrid strategy. One of the advantages of replacing the entire workflow is the computational cost: neural networks are expensive to train but fast to run inference on new data. They can also perform inversion without requiring an initial earth model by framing the problem as image to image, where the input "image" would be a group of seismograms and the output would be the corresponding earth model.

The approach of improving the existing FWI procedure aims to incorporate DL in some stages of the workflow, for example by using DL to build the starting velocity model(CAMPOS; NOGUEIRA; NASCIMENTO, 2019; DHARA; SEN, 2022). In this setting, a neural network is used to create an initial computational model of the subsurface that FWI can iterate upon. The idea is to shorten the number of iterations required to reach a satisfactory model by using DL for a rough approximation and FWI for the fine details.

Although promising, deep learning approaches, in general, require a high volume of data for better generalization (HALEVY; NORVIG; PEREIRA, 2009; SUN et al., 2017), and due to the limited availability of high quality velocity models(OVCHARENKO et al., 2019; REN et al., 2021), researchers often have to find ways to circumvent that.

2.4 Image generation

Machine learning models can be categorized as either discriminative or generative. Discriminative models try to create a separation between data samples, while generative models try to produce new samples. One of the more popular generation topics is image generation: given a set of images, produce new images that look like they belong

to the original set. The standard way to tackle this problem is to assume the given images were sampled from an unknown distribution, then try to approximate this distribution, and finally sample the estimated distribution to create new images. Estimating the distribution, however, is no easy task. Some of the most popular approaches to tackle such problem include using Generative Adversarial Networks (GANs)(GOODFELLOW et al., 2014), VAE(KINGMA; WELLING, 2013) and Diffusion Models (Sohl-Dickstein et al., 2015; HO; JAIN; ABBEEL, 2020), which will be further explored in the following sections.

2.4.1 Generative Adversarial Networks

The working principle of GANs(GOODFELLOW et al., 2014) is the adversarial training. Two networks are pitted against each other: one termed the generator, tasked with the creation of new data; and the other is the discriminator, responsible for distinguishing real data from data created by the generator. In an unconditioned image synthesis setting, the generator takes as input a random vector and outputs an image. On the other hand, the discriminator acts as a binary classifier, taking an image as input and outputting the probability of it being real. If the generator is able to perfectly approximate the original data distribution, the discriminator performs as well as random guessing and equilibrium is reached.

This adversarial relationship is expressed as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{n}\sim p_{\boldsymbol{n}}(\boldsymbol{n})}[\log(1 - D(G(\boldsymbol{n})))] \tag{5}$$

Where D is the discriminator, G is the generator, $\boldsymbol{x}$ is a real data vector, $\boldsymbol{n}$ is a noise vector, p_{data} is the true data distribution and $p_{\boldsymbol{z}}$ is the noise distribution.

The first term in equation 5 is the expected value when feeding real data to the discriminator, while the second term is the expected value of the complement when feeding fake data to the discriminator. D is chosen such that it maximizes both terms, while G is chosen to minimize the second, as it's the only term it has influence over.

Optimality is reached when $p_g = p_{\text{data}}$, that is, when the generator perfectly approximates the true data distribution (GOODFELLOW et al., 2014). At this point, the discriminator's distribution, given by $\frac{p_{\text{data}}}{p_{\text{data}}+p_g}$, equals $\frac{1}{2}$, meaning it's unable to differentiate real from fake.

Since their introduction, and much due to the flexibility of the proposed framework, GANs were applied successfully to a wide variety of tasks, including image generation, data augmentation, super-resolution, semi-supervised learning. While capable of producing high quality samples, GANs are notoriously hard to work with. Problems like mode collapse, training convergence and stability are common (KODALI et al., 2017; SAATCHI; WILSON, 2017; GOODFELLOW, 2017).

2.4.2 Variational Autoencoders

Similar to GANs, VAEs(KINGMA; WELLING, 2013) also work with two networks. In this case, however, one is termed the encoder (or recognition model) and the other is the decoder (or generative model). They differ from traditional autoencoders by being probabilistic. Instead of learning a lower dimensional representation of the input that will be passed to to decoder, they instead model a probability distribution over latent variables by predicting its parameters, which will be sampled from and used to produce new data.

The sampling process introduces randomness to the network, which prevents back-propagation. This issue, however, is fixed by the reparameterization trick introduced by (KINGMA; WELLING, 2013). Training is conducted by using an objective that combines the standard reconstruction loss from autoencoders and an additional term that represents the prior belief over the distribution of latent variables, as follows:

$$\max_{\phi,\theta} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$$

Where the first term represents the reconstruction loss, and the second term is the KL Divergence between a prior belief $p(\mathbf{z})$ and the distribution learned by the encoder $q_\phi(\mathbf{z}|\mathbf{x})$. This approach is necessary to avoid the encoder learning only point probabilities for samples in the training set (LUO, 2022).

VAEs are notorious for their ability to generate a high diversity of samples(TRINH; HAMAGAMI, 2024), although they generally produce lower quality images when compared to other techniques such as GANs and Diffusion models(LARSEN et al., 2016; TRINH; HAMAGAMI, 2024).

2.4.3 Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM)(HO; JAIN; ABBEEL, 2020), called "diffusion models" from here on, are a family of generative models that work by deconstructing an image by progressively adding noise only to then learn the reverse process, that is, to reconstruct the original image from pure noise.

Diffusion models work in two steps: forward diffusion and backward or reverse diffusion. In the former, data is slowly destroyed by successive noise additions over T timesteps until its indistinguishable from random noise. In the latter, noise is progressively removed until we have a clean image.

The process of adding gaussian noise to an image $\boldsymbol{x}_t$ can be formulated as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$$

Where the intensity of noise added follows a variance schedule β that increases with each timestep t and $\mathbf{I}$ indicates the identity matrix. $\mathcal{N}$ indicates a multivariate gaussian

distribution, where the mean is determined by the variance schedule and the image at a previous timestep $\boldsymbol{x}_{t-1}$, and the covariance matrix is determined by the variance schedule and the identity matrix.

Calculating the reverse process, that is $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, is not immediately obvious, but given a high enough number of timesteps T and small enough values of β, it can be modeled as a gaussian:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu(\boldsymbol{x}_t, t), \Sigma(\boldsymbol{x}_t, t))$$

Where the variance $\Sigma(\boldsymbol{x}_t, t)$ can be fixed to a constant $\beta_t\mathbf{I}$, and the mean μ can be expressed in terms of $\boldsymbol{\epsilon}$, the noise added in the previous timestep:

$$\mu(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon_t})$$

With $\alpha_t := 1 - \beta_t$ and $\bar{\alpha_t} := \prod_{s=0}^{t} \alpha_s$.

A neural network can then be trained to estimate ϵ, given the noise timestep t, and for that matter a simple mean squared error loss was shown to work well in practice (HO; JAIN; ABBEEL, 2020).

$$\min_{\theta} \mathbb{E}[(\boldsymbol{\epsilon} - \boldsymbol{\epsilon_\theta}(\boldsymbol{x_t}, t))^2]$$

Diffusion models proved to be able to generate high quality images and diverse sample (DHARIWAL; NICHOL, 2021; ROMBACH et al., 2022), but that comes in the expense of a more costly sampling process involving T iterations of noise removal.

2.5 Style transfer

Style transfer(GATYS; ECKER; BETHGE, 2015) is a technique that involves decomposing an image into style and content. It is based on the idea that Convolutional Neural Networks (CNNs) work hierarchically, and along the network, feature maps start to increasingly care about the content of an input image, with earlier layers, in contrast, only caring about raw pixel values. These later feature maps are taken to construct the content component of an image. To obtain a component for style, it uses the correlations between the filter responses in each layer of the network, given by the Gram matrix.

By using gradient descent and defining a loss function for each of the style and content components, one can optimize an initial image to fit a defined style and content criteria. A CNN is used to extract the components from three images:

- ❑ $\boldsymbol{x}$: The target image that will be modified
- ❑ $\boldsymbol{a}$: The image that will serve as the style source for $\boldsymbol{x}$
- ❑ $\boldsymbol{p}$: The image that will serve as the content source for $\boldsymbol{x}$

The CNN remains unchanged in the process, and the optimization works directly on the target image $\boldsymbol{x}$, which starts as random noise. The optimization process can be expressed as the following:

$$\min_{\boldsymbol{x}} \alpha L_{\text{content}}(\boldsymbol{x}, \boldsymbol{p}) + \beta L_{\text{style}}(\boldsymbol{x}, \boldsymbol{a}) \tag{6}$$

Where α and β are scaling factors that indicate the contributions of content and style reconstructions, respectively. In the original work, L_{content} is the Mean Squared Error (MSE) between feature representations and L_{style} is the weighted MSE between the Gram matrices of the original image and the style source. In most cases, a perfect match of style and content isn't achieved, and there's a trade-off between the two.

2.6 Conclusion

The concepts presented in this chapter are important to understand the rest of this dissertation or are used by related work. One of the key advantages of DLI over traditional FWI is its computational efficiency stemming from neural network's fast inference, compared to FWI's iterative solving of the wave equation. Although, to achieve that, an expensive training step is necessary.

The techniques on Section 2.4 are either commonly used in DLI research or are further developed in this work. GANs and Style transfer are used as ways to generate new velocity models, with GANs also being applied for the inversion task itself. VAEs and diffusion models will both be employed to generate velocity models on the following chapters.

Chapter 3

Related work

The majority of existing works in DLI focus on solving the inversion problem, with not many studies prioritizing the analysis of strategies to enhance such methods. For this reason, we review in this chapter, works on the two topics that are the most closely related to our object of study: neural network architectures for Deep Learning Inversion (DLI), and methods to generate velocity models.

We start by providing an overview of some well-known architectures for DLI, highlighting their main characteristics and training routines, followed by the presentation of a wide variety of techniques for velocity model generation, grouping them in three groups, according to the main idea behind the generation process: expert-made, mathematical modeling and deep learning.

3.1 Architectures for DLI

Due to the flexibility of neural networks, architectures for DLI can vary significantly depending on the problem formulation, type of input data and expected output.

Wu and Lin (2018) introduces InversionNet, an encoder-decoder network architecture for seismic inversion built primarily with convolutional layers (fully convolutional). The encoder path is implemented by strided convolutions that gradually reduce the spatial size of the input seismic data until it is transformed into a high dimensional vector. The decoder path is implemented by transposed convolutions that transform the high dimensional vector into a feature map with increased spatial size, that serves as an input to a CRF that is used to produce the final 2D velocity model output. The network is trained on two distinct synthetic 2D datasets, one with flat structures, and the other with folded layers. Every sample in the training set contains exactly 1 geological fault which can be

oriented in different ways. The authors conduct experiments to evaluate inversion quality compared to physics-based methods, robustness and generalization capabilities, where InversionNets proves capable of outperforming the physics methods on the evaluated scenarios, while displaying moderate resistance to additive gaussian perturbations. The included CRF provides some benefit in terms of reducing inversion artifacts, but increases computational cost proportional to the number of nodes in the CRF.

Araya-Polo et al. (2018) proposes a deep neural network combining convolutions and fully connected layers for velocity inversion. As a way to reduce the computational cost while maintaining high information density, the authors perform a feature extraction step on the seismic data by constructing a semblance cube and use that as the network input. The network is trained and evaluated on a fixed set of 2D synthetic data, generated to conform to fixed set of parameters such as a maximum and minimum number of layers and geological faults.

Yang and Ma (2019) introduces a fully convolutional network for seismic inversion, closely following a UNet(RONNEBERGER; FISCHER; BROX, 2015) structure, with direct connections between intermediate representations. Max pooling layers are used for the initial compressing path, while transposed convolutions are used for the expanding path. The final output is truncated to match the desired velocity model size, and the network is trained on 2D synthetic data with salt bodies embedded into each velocity model. The authors conduct experiments evaluating the proposed model on a separated set of synthetic data, and use the learned weights for transfer learning to a different set with a smaller amount of data samples, including ones with no salt bodies. The networks outperforms traditional FWI for situations similar to the training set, with models containing salt bodies, but underperforms when salt bodies are absent. On a similar note, (ZHANG; LIN, 2020) performs experiments on the generalization capabilities of data-driven solutions and introduces an inversion method called VelocityGAN, that is based on the adversarial training framework using a network similar to InversionNet as the generator and a CNN as the discriminator. As a way to improve robustness of the model, the authors use a Wasserstein loss (ARJOVSKY; CHINTALA; BOTTOU, 2017) with gradient penalty for training. The model is initially trained and evaluated on synthetic data with a single geological fault, and then the generalization capabilities are assessed by testing on synthetic data with zero and two faults. In general, VelocityGAN outperforms the evaluated physics-based solutions and showcases basic generalization abilities. Generalization can be enhanced by fine tuning a previously trained model on a smaller set of samples from the new target distribution, in this case models with zero or two faults.

Adler, Araya-Polo and Poggio (2019) evaluates incorporating recurrent layers to the more traditional convolutional and fully connected compositions for seismic inversion. The main motivation is the fact that recurrent components are well suited to handle sequential data, such as the seismic traces obtained from a seismic survey. Three recur-

rent cells are evaluated: standard Recurrent Neural Network (RNN), Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) cells. The models are trained using a synthetically generated dataset, with the majority of created data containing salt bodies. Overall, the networks with recurrent components showed improved MSE and Structural Similarity (SSIM) when compared to the CNN baseline, while being considerably smaller when accounting the total number of parameters.

Zhu et al. (2023) proposes using fourier operations in the form of fourier blocks in neural network design. The model is called Fourier-DeepONet, and is based on DeepONets(LU et al., 2021), a construction that generalizes the learning problem to learning mappings between infinite-dimensional function spaces. Input processing is split in two parts: a "trunk" that takes the source parameters, and a "branch" that handles the seismic data. The results of both parts is then combined and fed to a sequence of fourier blocks that parameterizes operations over the low frequencies in the fourier space, that are then converted back with the inverse fourier transform. Fourier blocks can also have a variation where an internal UNet is included as a way to handle high frequency information. Fourier-DeepONet is trained and evaluated on synthetic velocity models from OpenFWI, where the input seismic data is calculated using multiple source frequencies and locations. The network showed strong performance for inversion with varying source frequencies when compared to InversionNet and VelocityGAN trained on the same data, but the computational complexity of the proposed fourier blocks is high, especially those with a UNet included.

3.2 Strategies for Velocity Model Generation

Due to the high value and strategic importance of seismic data, field data is rarely shared. For this reason, researchers often have to work with synthetically generated alternatives. Following, we review some strategies to artificially create velocity and earth models.

3.2.1 Expert-made models

A straightforward way to produce velocity models is having them be constructed by domain experts. Those works generally try to replicate structures or characteristics found in specific regions of the real world. Of this category, the most popular ones are:

Marmousi2 (MARTIN; WILEY; MARFURT, 2006): Created by the Allied Geophysical Laboratory in 2006, Marmousi2 is a 2D synthetic elastic dataset that expands upon its predecessor: Marmousi(VERSTEEG, 1994). The original was created in 1988 by the Institut Français du Pétrole and was based upon geology from the North Quenguela Trough in the Quanza Basin of Angola. The expanded version adds characteristics like a fully elastic model, greater variety of geological layers and increased depth and width.

The dataset contains complex structures in many configurations that aim to simulate the heterogeneity found in real-world subsurface environments. The dataset spans an area of 17 × 3.5km.

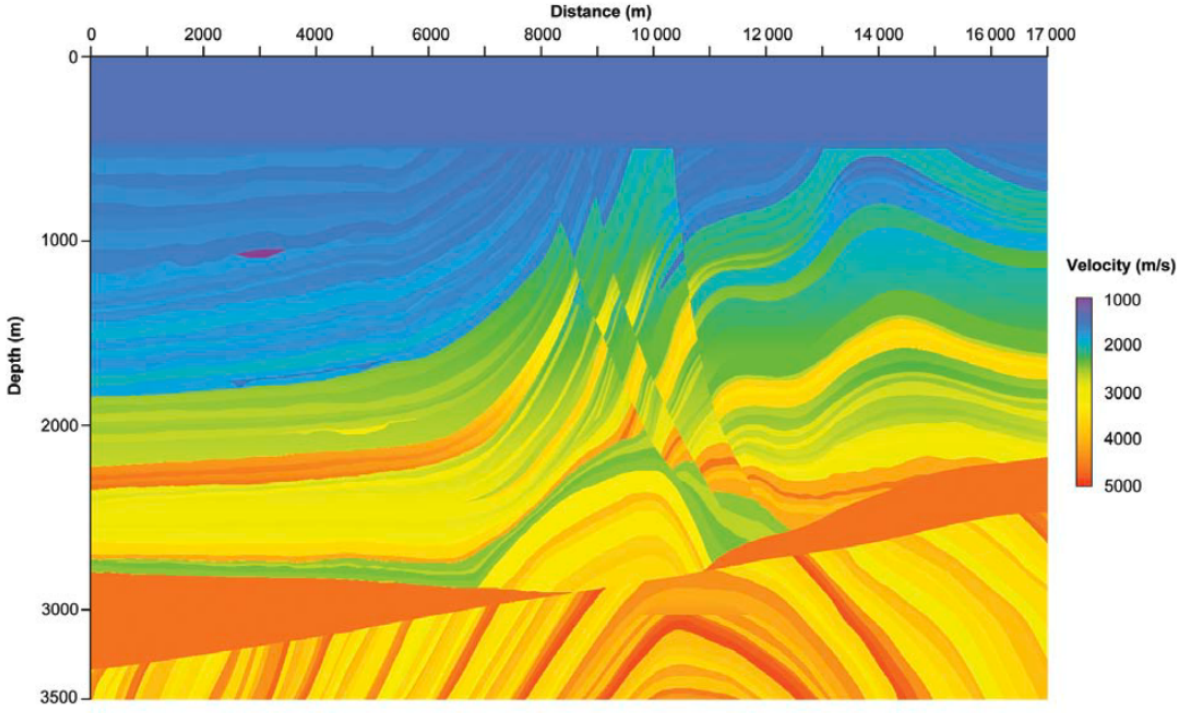

Figure 3 – P-wave velocity from Marmousi2 dataset. Source: (MARTIN; WILEY; MARFURT, 2006)

3D Salt and Overthrust models (AMINZADEH; BRAC; KUNZ, 19): Created in a collaboration between the Society of Exploration Geophysicists (SEG) and the European Association of Geoscientists and Engineers (EAGE) in 1997, the salt and overthrust models are two 3D synthetic acoustic datasets based on the Gulf of Mexico (salt) and South America (overthrust). They were validated by a multitude of experts in industry and academia and aimed to represent the complexity of structures encountered in exploration and test/benchmark 3D techniques and algorithms. The two datsets have the following dimensions: 13.5 × 13.5 × 4.2 km (salt) and 20 × 20 × 3.5 km (overthrust).

2004 BP Velocity Benchmark (BILLETTE; BRANDSBERG-DAHL, 2005): A 2D synthetic acoustic dataset that describes a 67km long and 12km deep region that combines features from the following locations: Gulf of Mexico, West Africa, Caspian Sea, Trindad and North Sea. The dataset was created by researchers at BP and was designed to cover several issues encountered when estimating migration velocity models in geophysically challenging areas around the world. It was originally used as a blind test of velocity model building techniques, and covers an area of 67 × 12km.

This generation method allows for the highest level of flexibility, although it is an expensive and time consuming. That being said, the high complexity of those models

comes at the cost of covering only a small region, which translates into insufficient data, especially for deep learning strategies.

3.2.2 Mathematical modeling

A mathematical modeling approach aim to devise an algorithm to create velocity models. In most cases, little information is needed to run the methods: a starting model and a set of parameters that will direct the algorithm's behavior, allowing for some degree of flexibility. This approach has the advantage of scale, being inexpensive to create a large amount of models, but the results are often simple, differing from most real-world applications.

Ren et al. (2021) proposes a framework to generate variable-sized, 3D velocity models capable of representing features such as folds, faults and salt domes. The framework adds interesting structures to an initially simple model containing just folding layers. Only after the structure is defined that the velocities values are assigned and the model is complete.

In a similar manner, Parasyris, Stankovic and Stankovic (2023) proposes a methodology to create 2D velocity models with folds and salt bodies. The structures can be build from either an initially simple model or from some geological prior, such as geological images, which allows it to incorporate domain information in the building process.

Deng et al. (2023) proposes OpenFWI, a large scale 2D/3D dataset for seismic inversion that includes a multitude of synthetic velocity maps. One of its main advantages is the volume of available data, with hundreds of thousand of velocity samples. Part of the 2D data is generated by starting from an earth model with flat layers of random width and iteratively applying randomly parameterized sine and linear functions to construct geological structures such as curves and faults.

Wang and Ma (2020) uses a subset of the COCO dataset (LIN et al., 2014) of natural images as a starting point for their velocity building method. Natural images are first converted to grayscale, then smoothed using a gaussian filter, and finally normalized to have velocity values within a range of 1500 m/s and 4500 m/s. Due to the relatively light amount of processing, in many cases the resulting velocity models preserve the features of the images that originated them, such as the distinct shapes of objects, vehicles and people. The authors make an observation that the physics of the inversion problem is the same, regardless if the wave equation is being propagated in a medium of geological significance or not. With forward simulation, they create a dataset of "seismic data"/"velocity model" pairs and use to train a DLI method. They conduct an initial study to find an appropriate dataset size and settle for around 80,000 samples, for which further experiments reveal an acceptable inversion quality.

3.2.3 Deep learning models

Earth models can also be created by DL approaches. In this method, a data-centric procedure is used to create new models, with neural networks being the most prominent technique.

As another strategy inspired by working with natural images, Deng et al. (2023) introduces a different approach to create velocity models by applying a deep neural network for style transfer from Marmousi2 (an expert made dataset) to the COCO dataset(LIN et al., 2014). The resulting models compose another fraction of the OpenFWI dataset. As a way to increase geological plausibility, the authors also incorporate a 1D background velocity map with linearly increasing velocity values. This term is weighted with the results of the style transfer to form the final output.

Similarly, Ovcharenko et al. (2019) demonstrates the use of an iterative style transfer approach by taking several simple "content models" such as random fields, linear gradients, and basic shapes and applies the style of Marmousi2 and BP 2004, two expert made models. The results show that the process can generate new, perceptually realistic models that mimic the complex layering and features of the style source.

Araya-Polo, Farris and Florez (2019) uses a GAN to create velocity models that are then used as training data in a DLI workflow. The neural network that is used for inversion is a CNN that is trained exclusively on the GAN-generated samples. Although the presented methodology is capable of generating a virtually unlimited amount of data, the results are provided for a training dataset of fixed size. Also working with GANs, Puzyrev et al. (2022) uses the simulation framework Badlands (SALLES et al., 2018) to create synthetic 3D subsurface models from a landscape evolution simulation. After the 3D models are produced, they are then sliced to obtain 2D models of density and stratigraphy. Those are used as input data to train StyleGAN2 (KARRAS et al., 2020) networks to create a cost-effective way to reproduce geological models that contain features similar to training samples. The authors highlight a possible application of the proposed method for DLI, but no experiments are conducted.

Wang, Huang and Alkhalifah (2024) proposes using diffusion models as a way to generate subsurface representations. The authors mainly explore conditional generation, which incorporates prior information such as class labels, well logs, and reflectivity images to guide the model to produce a velocity model that conforms to the prior. The network used to produce velocity models employs residual blocks with attention components organized in a UNet-like structure, and is trained on the mathematically modeled part of OpenFWI, meaning it excludes the portion obtained via style transfer. The proposed method is capable of closely approximating complex data distributions, while being able to generate a varied set of possible realizations of a single conditional constraint. Similarly to Puzyrev et al. (2022), the authors highlight the possible application of the proposed method for DLI, but provide no experiments.

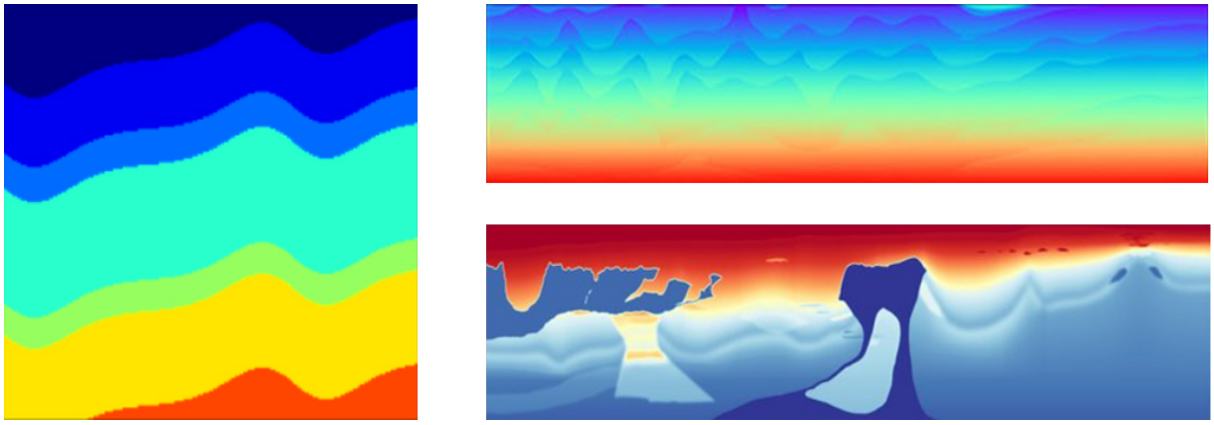

Figure 4 – Math modeling (PARASYRIS; STANKOVIC; STANKOVIC, 2023), deep learning (PUZYREV et al., 2022) top right) and expert made((BILLETTE; BRANDSBERG-DAHL, 2005)bottom right) models

3.3 Conclusion

Data scarcity is a pervasive issue that haunts any work trying to apply DL to the seismic domain. Researchers exploring this area have to find a way to deal with it. However, in the majority of cases, once the problem is sufficiently alleviated, meaning a sufficient amount of data is acquired to obtain decent results, efforts in this direction stop. Even in cases where a theoretically unlimited amount of data is available by producing earth models and running forward simulation, this venue is often ignored.

Running forward simulation to create new seismic data is a considerable cost, but what the recent history of DL showcases is that efforts on data collection and enhancement are not only tools to avoid the data scarcity problem, but also tools to enhance the performance of virtually any model, with it often being not just a tool, but the best tool.

The following chapters explore this comparatively overlooked research direction, initially following a line of research similar to Wang, Huang and Alkhalifah (2024) for the synthesis of velocity models, then exploring a more novel approach involving VAEs. To evaluate the effectiveness of the generated velocity models we compare some established DLI architectures on a more controlled setting with and without synthetic data, namely InversionNet (WU; LIN, 2018), VelocityGAN (ZHANG; LIN, 2020) and FourierDeepONet (ZHU et al., 2023). Finally, we utilize synthetic data to its full potential in an international seismic inversion competition.

Chapter 4

Diffusion for Unconditional Velocity Model Generation

The capability to produce sharp, well defined shapes is crucial to accurately represent subsurface structures. This is mainly because some underground formations, such as geological faults, are defined by an abrupt contrast of materials, translated into an abrupt contrast of velocities. When compared to other types of generative models, such as GANs and VAEs, diffusion models are able to generate high quality, sharp images in the natural domain (ROMBACH et al., 2022; DHARIWAL; NICHOL, 2021), this makes them a potentially suitable choice for the task.

Previous work from Wang, Huang and Alkhalifah (2024), under a similar line of reasoning, explored mainly conditional velocity model generation. In this setting, some information (condition) is used to guide the generation process, constraining the output to obey it. Wang, Huang and Alkhalifah (2024) uses well logs, class information, and seismic images as conditions to generate the velocity models. Unconditional generation allows for a more suitable approach when the main goal is to produce a large number of varied samples derived from the training data distribution. This follows naturally from the model not having any kind of constraints, enabling it to thoroughly explore the full range of possibilities.

In this chapter, we start the development of synthetic data strategies by creating an unconditional diffusion network to generate new velocity models. A new set of velocity models can be used for forward wave propagation, producing the pairs "velocity model/seismic data" necessary for DLI. The dataset used to train the diffusion model is OpenFWI, which includes samples with a variety of geological characteristics. The synthetic models then undergo a qualitative analysis of characteristics, to spot potential

problems when using them as training data for DLI.

4.1 Denoising UNet

The central piece of a diffusion model is a denoising network capable of estimating the noise ϵ added to an input image, given a timestep t. An image-to-image model, like UNet (RONNEBERGER; FISCHER; BROX, 2015), can be trained for this purpose.

Training samples consist of pairs of "noisy image"/"added noise" that can be obtained by simply generating random noise, storing it and adding it to a clean training image. The intensity of the noise added depends on the timestep, so the timestep should also be stored. As the process of generating and adding noise is cheap and can be executed in the CPU, it happens on demand, while the network is training. Figure 5 illustrates the denoising UNet architecture used. The down and up blocks consisted of groups of convolutional layers and convolutional layers with attention. Down blocks reduce spatial dimensions to half, while up blocks double them. To incorporate timestep information, we used sinusoidal encodings and an Multilayer Perceptron (MLP) for time embedding.

When training diffusion models, it is important to choose a high enough number of timesteps T. The reason is that forward diffusion should destroy the original input, to the point where the end result is indistinguishable from random noise. At the same time, the destruction process should happen by incrementally adding small amounts of noise at each step, otherwise we can't make the assumption that reverse diffusion is Gaussian. A small T implies that, either the noise increments are not small, or forward diffusion does not result in random noise. The final network is trained with $T = 1000$, meaning that 1000 possible degrees of noise exist.

4.2 Training data - OpenFWI

Training data consisted of datasets from OpenFWI(DENG et al., 2023). OpenFWI is a large scale 2D/3D benchmark for seismic inversion that includes a multitude of pairs "velocity model"/"seismic data", created with DLI in mind. The full collection is subdivided into datasets representing different subsurface structures (interfaces, flat layers, faults, folds) each with an easy (A) and hard (B) version. The 2D OpenFWI datasets, as well as the number of samples in each, is displayed in Table 1. Each 2D velocity model represents a region of $700m \times 700m$ on an uniform grid of with $10m$ cells, translating into a 70×70 matrix. Velocity values range from $1500\,m/s$ to $4500\,m/s$. In order to work with velocity models as images, velocity values were normalized to $[0, 1]$, enabling them to be interpreted as grayscale pictures.

OpenFWI 2D velocity models don't necessarily obey a rule of increasing velocity with depth, with the only exception being models from the Style-A and Style-B datasets. This

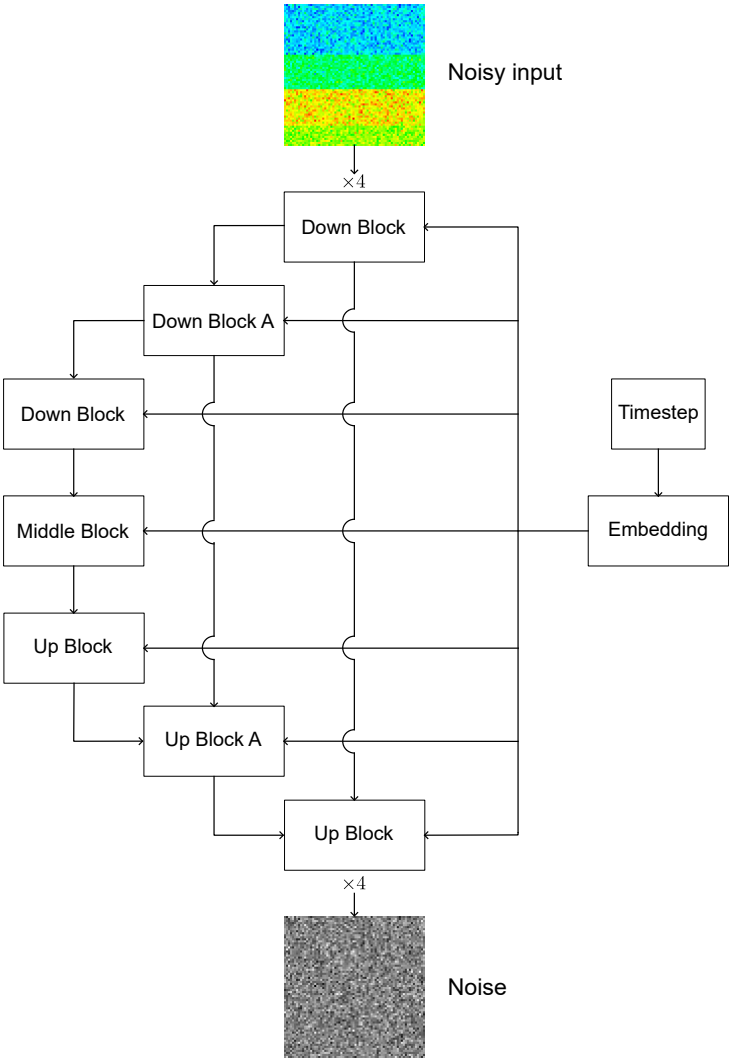

Figure 5 – Diagram of diffusion UNet architecture. Training input is an image with noise added and the corresponding timestep, and output is the noise that was added to the image. Incorporating timestep information allows the network to behave differently under different noise intensities. Blocks with an "A" indicate the presence of attention components.

translates into some unrealistic configurations, such as a flat layer of water in between two flat layers of rock. Even Style-A and Style-B have some unrealistic elements due to being based on style transfer from natural images, resulting in formations that are often uncommon in the underground. Nevertheless, the task of inverting those models from seismic data is still challenging: assumptions that could be safely made in a traditional setting, can't be made for OpenFWI. Figure 6 illustrates velocity model samples from each of the 10 OpenFWI datasets.

Instead of using all 408,000 training samples, we opted to work with each dataset individually. This translates into training a denoising UNet for each of the 10 datasets. One advantage of this approach when compared to using every available sample is that the network's task is simplified: instead of learning the joint distribution of all the datasets,

Table 1 – Number of velocity model/seismic data pairs and type of geological structures in each of the 2D OpenFWI datasets

Dataset	**# Training**	**# Validation**	**Structures**	**Origin**
FlatVel-A	24,000	6,000	Flat	Math modeling
FlatVel-B	24,000	6,000	Flat	Math modeling
CurveVel-A	24,000	6,000	Folds	Math modeling
CurveVel-B	24,000	6,000	Folds	Math modeling
FlatFault-A	48,000	6,000	Flat, Faults	Math modeling
FlatFault-B	48,000	6,000	Flat, Faults	Math modeling
CurveFault-A	48,000	6,000	Folds, Faults	Math modeling
CurveFault-B	48,000	6,000	Folds, Faults	Math modeling
Style-A	60,000	7,000	-	Deep learning (style transfer)
Style-B	60,000	7,000	-	Deep learning (style transfer)
Total	408,000	62,000		

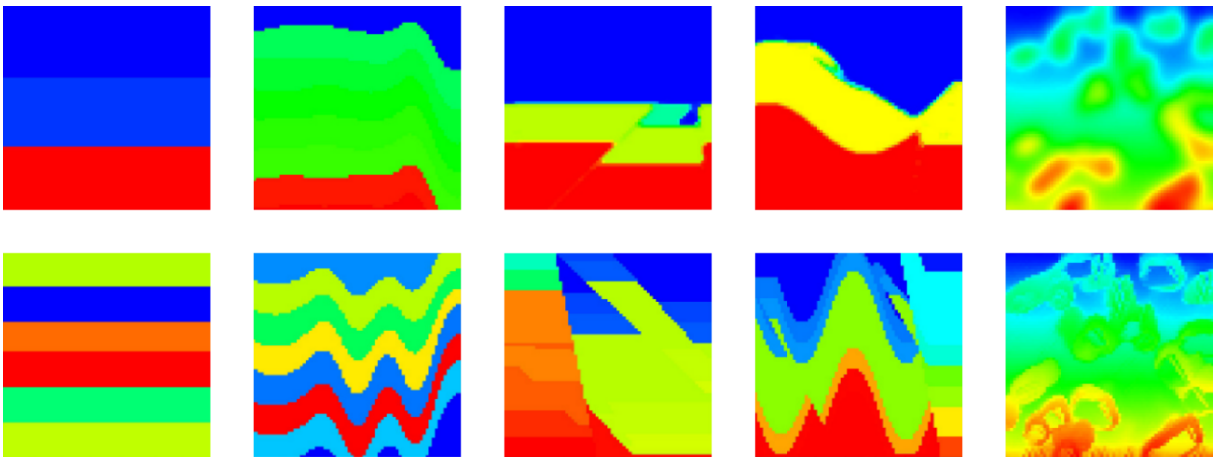

Figure 6 – Samples from each of the OpenFWI 2D datasets. From top to bottom, left to right: FlatVel-A, CurveVel-A, FlatFault-A, CurveFault-A, Style-A, FlatVel-B, CurveVel-B, FlatFault-B, CurveFault-B, Style-B. Colormap was omitted for readability. Colors close to blue indicates lower velocities (similar to that of water), while colors closer to red indicate higher velocities.

it learns the simpler distribution of a single one.

4.3 Evaluating synthetic velocity models

When running inference, sampling over a big number of timesteps is computationally expensive, as each generated sample will require T passes on the denoising network. As a way to alleviate this problem, what generally works well in practice is to analyze the inference results and choose the lowest number of timesteps that achieve a desired level of quality. Sampling from a model trained with a higher number of timesteps can be

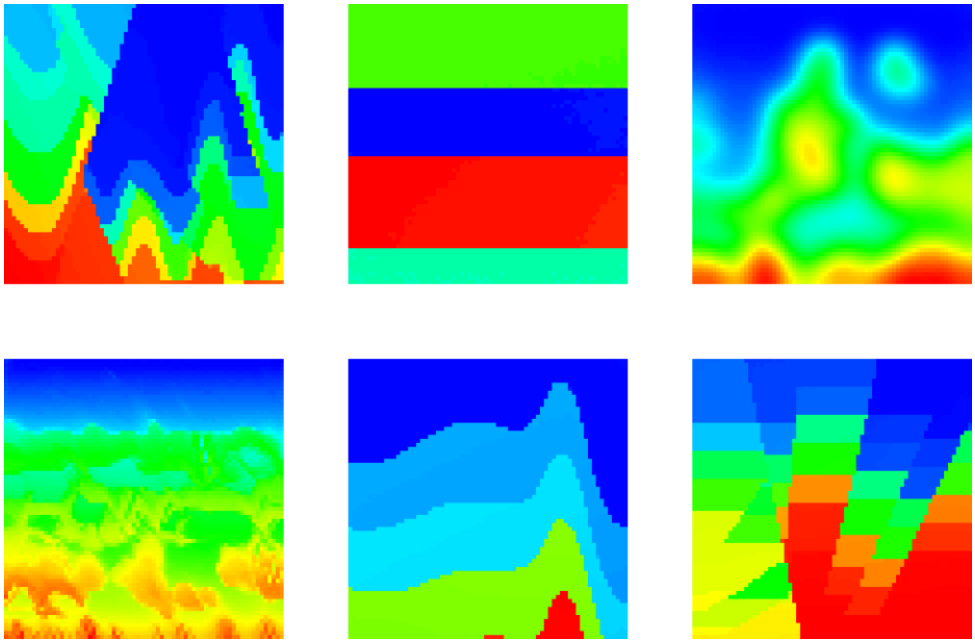

Figure 7 – Velocity models generated from the dataset-specific diffusion networks with $T = 200$. Each network was trained in an OpenFWI subset, and generates samples corresponding to that subset. Samples shown are from diffusion networks trained with the following subsets, from top to bottom, left to right: CurveVel-B, FlatVel-A, Style-A, Style-B, CurveVel-B, FlatFault-B

achieved by skipping timestep values during inference. Instead of running the network for all timesteps $\{999, 998, ..., 1, 0\}$, we can work with 200 instances of evenly spaced timesteps $\{999, 994, ..., 9, 4\}$. Figure 7 shows samples from the diffusion models inference process using the diffusion UNet for specific OpenFWI datasets.

The produced models highly resemble the original datasets, maintaining the prevalent characteristics of each, such as faults and curves on CurveFault, straight layers on FlatVel, and irregular shapes with soft/sharp edges following a velocity increase on Style-A/B. This turns them into possibly useful assets for applications on OpenFWI that could benefit from more data, with the most straightforward one being training deep neural networks. The diffusion models were even capable of replicating finer, more detailed patterns, such as the ones illustrated in Figure 8. In general, OpenFWI data possesses some unusual characteristics, probably derived from implementation details of the generation process, that can be mistaken for generation artifacts if the training data is not carefully inspected. Figure 9 shows, for example, that the presence of a 1-pixel width strip at the bottom of CurveFault-B models containing highly distinct velocity values is, in fact, a common occurrence on the original set. They are much more present in CurveFault-B than in other datasets, probably due to the more elaborate generation process required to have flat sections, curved layers and geological faults.

The most noticeable artifact from the generation process seems to be the model's inability to maintain a uniform velocity value in some cases where it most likely should

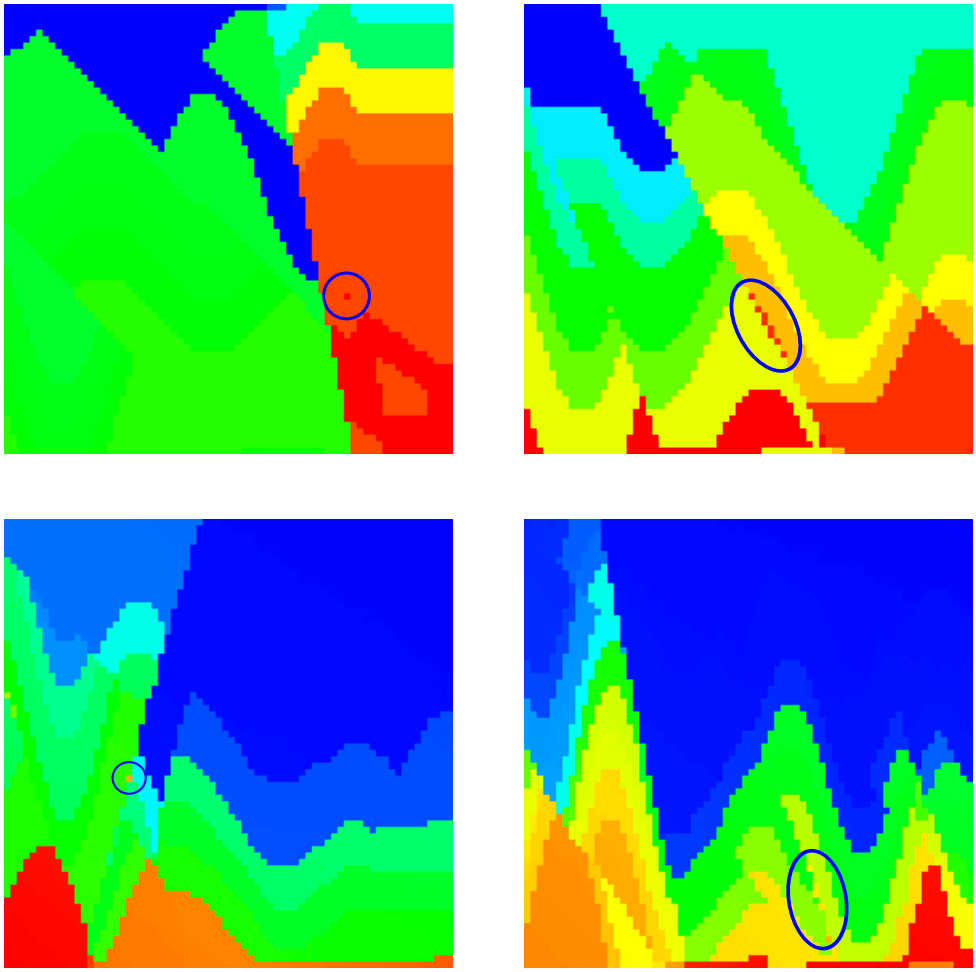

Figure 8 – Showcasing (blue highlights) some characteristics present in the OpenFWI training data (top) that are also present in the diffusion-generated models (bottom): standalone velocity islands (left), and velocity trails following a fault structure (right)

have, as highlighted in Figure 10. However, as this condition happens sporadically and doesn't compromise the overall structure of the generated data, as the non-uniform values are still close to the target, it was judged a minor issue.

4.4 Conclusion

In this Chapter we elaborated on some of the inner workings and practical considerations of implementing an unconditional diffusion model to generate synthetic subsurface velocity structures. Namely the possibility to add noise on demand during training, and the option to use a different number of timesteps for generation. The developed solution proved capable of producing samples that are highly similar to the source distribution, successfully addressing Specific Objective 1 (SO1), even displaying intricate patterns from the training data. However, we identified what appears to be a problem in maintaining the uniformity of velocity values along a geological layer. Due to its low frequency, it was

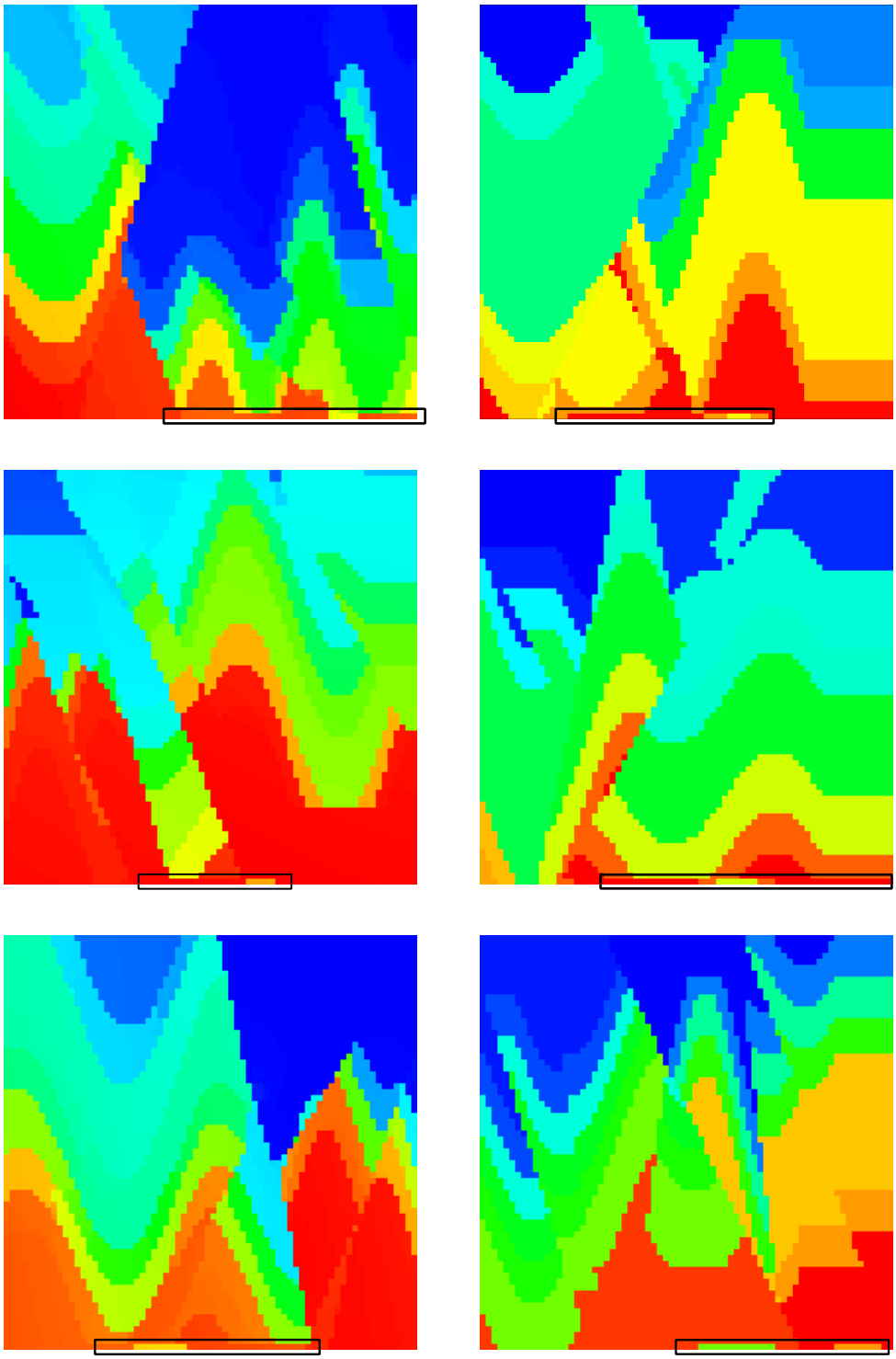

Figure 9 – Comparison between diffusion-generated models (left) and original (right) for CurveFault-B. The diffusion model correctly replicates the 1-pixel width pattern with varying velocities from the original data

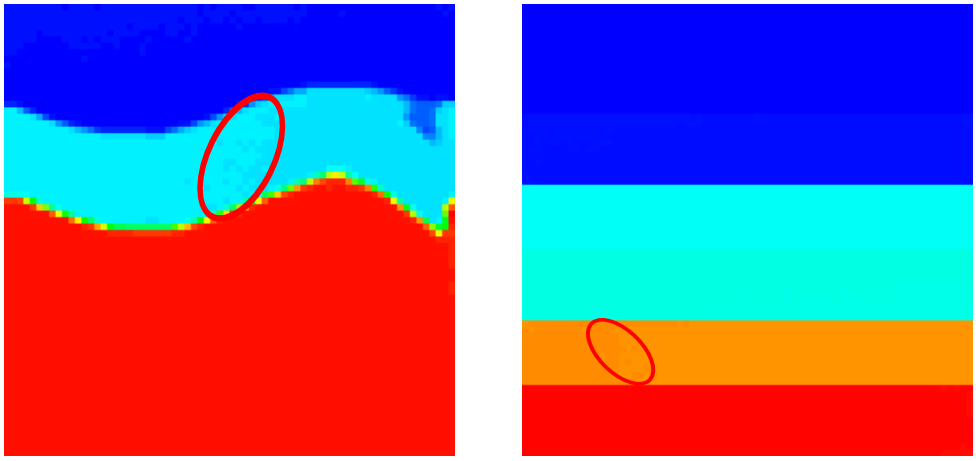

Figure 10 – Diffusion artifact of non-uniform velocity (highlighted in red) on models generated for CurveFault-A (left) and FlatVel-A (right).

judged a minor issue, but could be mitigated by including a post-processing step that reduces the number of velocity values present.

Chapter 5

VAEs for fine control over velocity models

One of the main advantages of VAEs compared to other generative models is their natural ability to learn informative representations of the data. This characteristic broadens their range of applications to any activity that can make use of their learned representations, such as anomaly detection and compression. In this chapter, we make use of the VAE's learned representation to create a method for fine control of velocity models, being able to alter existing subsurface maps to include or remove desired geological characteristics.

5.1 VAEs encourage latent space structure

Learning the underlying distribution of a generative process is often challenging due to the high dimensionality of the data involved, as acquiring a sufficient amount of data samples to densely populate a high-dimensional space is unfeasible in most cases. Latent variable models propose an alternative formulation to the generation process by introducing lower-dimensional unobserved components, the latent variables $\mathbf{z}$. An intuitive interpretation of the latent space is as representing higher-level characteristics that somehow describe the high-dimensional data. This allows us to express the generative process by factorizing the joint distribution of the data and the latent variables $p(\mathbf{x}, \mathbf{z})$ as follows:

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{z}) &= p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \\
p(\mathbf{x}) &= \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[p(\mathbf{x}|\mathbf{z})]
\end{aligned} \tag{7}
$$

Sampling from $p(\mathbf{x})$ then becomes a two-step procedure, where we first sample a latent variable from $p(\mathbf{z})$, and obtain $\mathbf{x}$ by sampling from the conditional distribution $p(\mathbf{x}|\mathbf{z})$:

1. $\mathbf{z} \sim p(\mathbf{z})$

2. $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$

VAEs are latent variable models that use approximation to enable training in cases where the integral in equation 7 is still intractable. As introduced in section 2.4.2, VAEs approximate the true posterior over latent variables $p(\mathbf{z}|\mathbf{x})$, and in their training objective this approximation is regularized to conform to a prior belief of the distribution over latents $p(\mathbf{z})$:

$$
\max_{\phi,\theta} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))
$$

Typically, $p(\mathbf{z})$ is chosen to be a standard Gaussian, and the KL-divergence term has the effect of clustering together latent representations for a same data point. A successful training with the KL component sufficiently reduced translates to a latent space where small variations on the representation don't cause big changes in the decoded representation in the data space. This characteristic is what is often referred to as the encouraged latent space structure.

5.2 A VAE for Velocity Models

The network architecture of a standard VAE is similar to the deterministic autoencoder, having two main components: an encoder network and a decoder network. The main difference is the introduction of sampling using the encoder output as parameters.

To create a VAE for velocity models, we trained an encoder-decoder network on the entire OpenFWI 2D training data. Details about the dataset are described in Table 1. Figure 11 illustrates the network architecture at high level. Both the encoder and decoder were implemented with a convolutions and transposed convolutions as the operations for down/upsampling. Latent representation had dimension $4 \times 4 \times 4$, resulting in representation vectors of size 64.

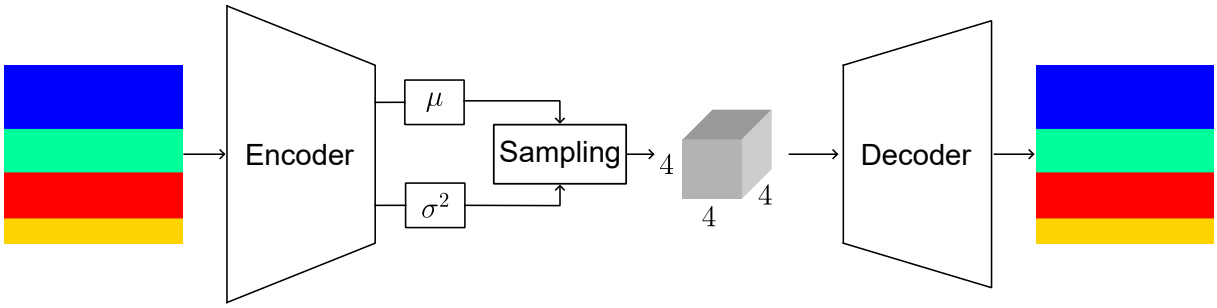

Figure 11 – VAE for velocity models. The training process consists on reconstructing the input velocity model after sampling using the parameters produced by the encoder. The output of the decoder is compared with the reference input to calculate the reconstruction error. The distribution produced by the encoder is also compared with a prior belief over the distribution of latent variables in the KL-divergence component.

5.3 Attributing meaning to latent representations

Although useful for interpretability, with similar latent representations decoding to similar outputs, the encouraged latent space structure is not sufficient to associate meaning to the latent variables. Some works on VAEs, such as β-VAEs (HIGGINS et al., 2017) partially tackle this problem by introducing changes in the objective function (the β term) to increase the level of disentanglement of learned latent variables, allowing for better associations between variables and their respective characteristics. Other more general approach involves calculating semantic vectors by averaging latent representations of concepts (RADFORD; METZ; CHINTALA, 2015). The benefit of this approach is that the averaged characteristics gives a clear semantic interpretation, with the downside being the need for labels in order to calculate each average. With that being said, OpenfWI data is indirectly labeled by its subdivision in datasets with distinct characteristics.

The first step to implement the manipulation of semantically meaningful features of velocity models involves using a trained VAE's encoder to calculate latent representations of the entire OpenFWI data. Then, using the source dataset of each representation, we associate labels the latent vectors. For example, a velocity model from FlatVel-A would get its latent representation generated by the VAE encoder, and would receive the labels "flat" and "A". Possible labels contained geological characteristics (flat, fault, curve) and if the example was easy (A) or hard (B). Then, by averaging all samples with a specific label, we can generate the label vector for that label.

The process of adding characteristics to velocity models, then becomes straightforward: combine the label vectors to already existing latent representations. A direct approach, similar to (RADFORD; METZ; CHINTALA, 2015), would be to use latent space arithmetic. We can add the label vector to a representation that doesn't have said label, then decode the result to visually assess if the velocity model now has the label. To maintain

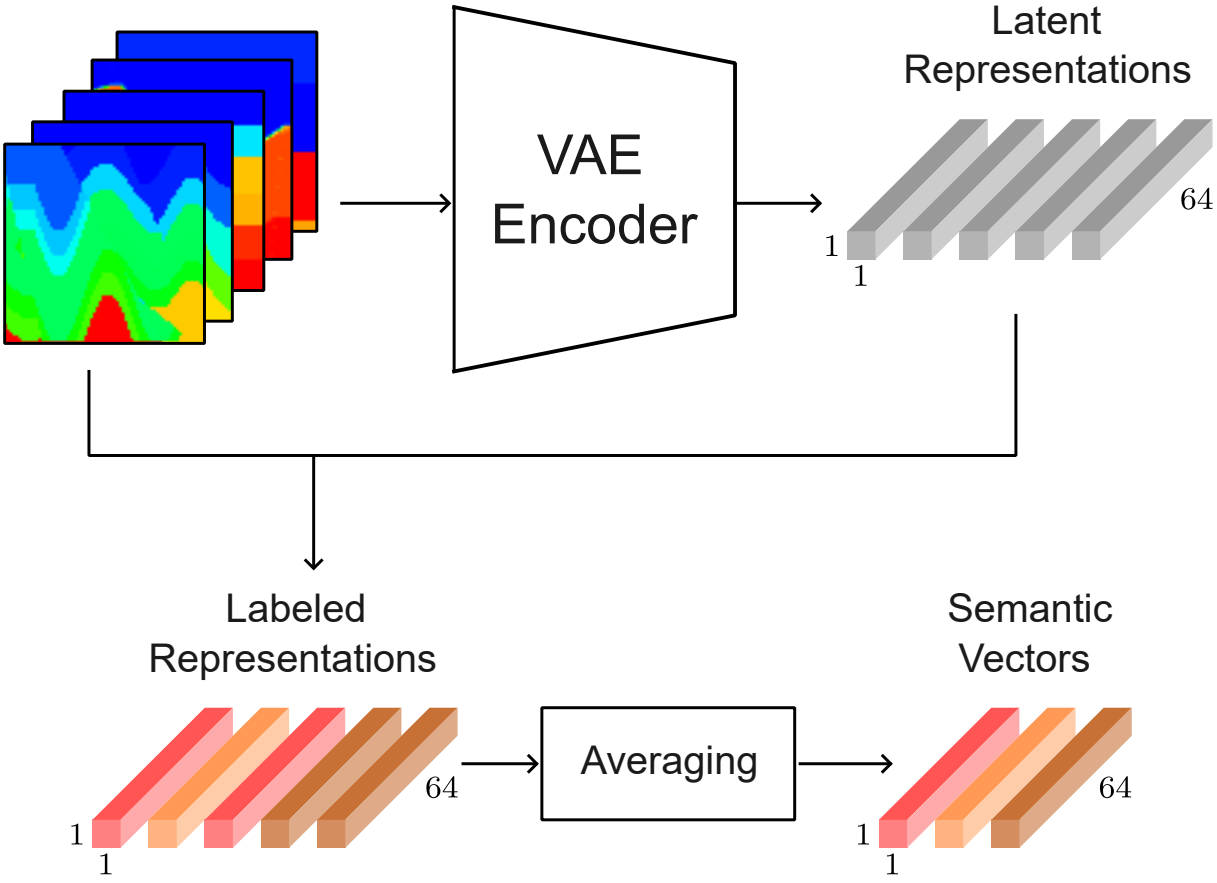

Figure 12 – Generating semantically meaningful vector representations of OpenFWI by using a VAE's encoder to obtain latent representations, using the dataset of origin as labels and averaging to obtain semantic vectors related to the labels.

the resulting vector withing the expected range of values, we limit the contribution of the label vector and the representation to α and $(1-\alpha)$, respectively, giving us:

$$V_L = V(1-\alpha) + \alpha L \quad \alpha \in [0,1],$$

where V_L denotes the velocity model V with the label vector L added. Figures 13 and 14 illustrate the operation of adding each of the 5 semantic or label vectors "flat", "curve", "fault", "a", and "b" to a simple velocity model (top left) with two distinct values of α.

The vector arithmetic approach, did not yield satisfactory results. More significant changes to the velocity model only surface after increasing the contribution of the label vector to very high levels, and the changes incurred don't resemble the vector's label. In Figure 14, after adding the "fault" vector, the flat layers of the example velocity model become more curved, with no faults present.

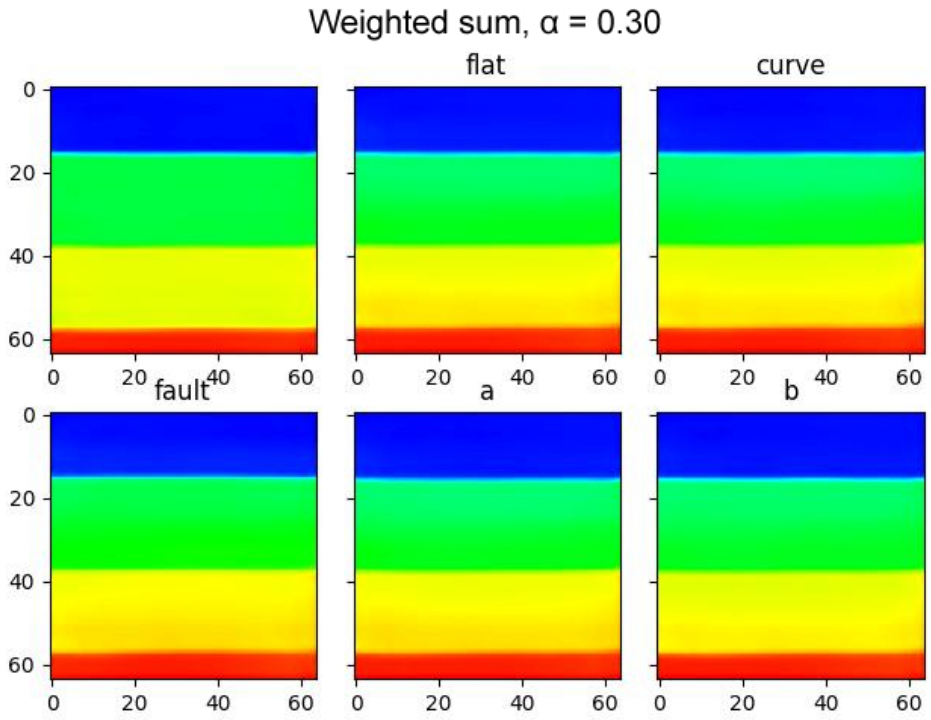

Figure 13 – Vector arithmetic with $\alpha = 0.3$. Top left is the original velocity model, then the same velocity model with added components for each label vector: flat, curve, fault, a, and b

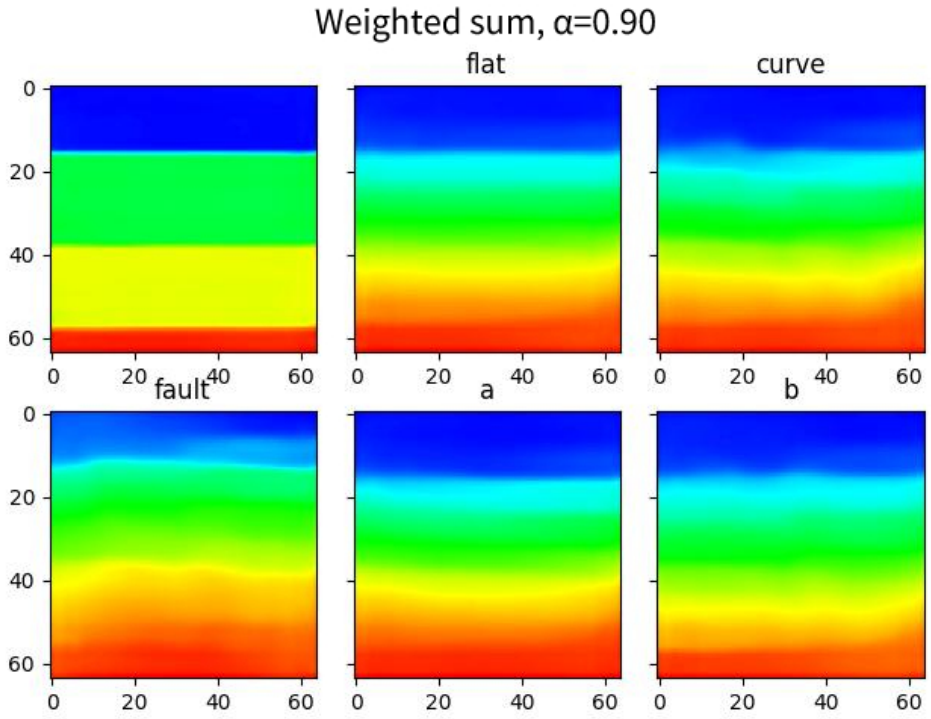

Figure 14 – Vector arithmetic with $\alpha = 0.9$. Top left is the original velocity model, then the same velocity model with added components for each label vector: flat, curve, fault, a, and b

5.3.1 Orthonormal basis with semantic components

An alternative approach for the manipulation of velocity model features involves constructing a base for the latent space, with the label vectors as components. This base can also be made orthogonal to introduce the notion of disentanglement. Algorithm 1 (WILLSE,) describes a procedure to generate a basis from a set of linearly independent vectors.

Algorithm 1 Extending a linearly independent set of vectors to a basis

Require:
An ordered, linearly independent set of vectors $S = \{s_1, \ldots, s_k\}$ in a finite vector space V.
An ordered basis $B = \{v_1, \ldots, v_n\}$ for V.
$k \leq n$.

Ensure:
An ordered basis B_{ext} for V such that $S \subseteq B_{ext}$.

1: $B_{ext} \leftarrow S$ ▷ Initialize the new basis with the vectors from S
2: **for** $i \leftarrow 1$ to n **do** ▷ Iterate through each vector of the existing basis B
3: **if** $v_i \notin \text{span}\{B_{ext}\}$ **then** ▷ Check if v_i is independent of the vectors found so far
4: $B_{ext} \leftarrow B_{ext} \cup \{v_i\}$ ▷ If independent, add it to the new basis
5: **end if**
6: **end for**
7: **return** B_{ext}

By choosing the starting set S to be equal to the label vectors $\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3, \mathbf{l}_4, \mathbf{l}_5\}$, we can apply Algorithm 1 to obtain a base B_{ext} for the latent space. The only consideration is that S should be linearly independent. This can be verified by assuring the equation

$$c_1\mathbf{l}_1 + c_2\mathbf{l}_2 + c_3\mathbf{l}_3 + c_4\mathbf{l}_4 + c_5\mathbf{l}_5 = \mathbf{0}$$

has only the trivial solution of all coefficients $c_1 \ldots c_5$ being zero.

With a basis defined, we can use the Gram-Schmidt process to transform B_{ext} into an orthonormal set B_{norm} that is also a basis. The Gram-Schmidt process is an iterative algorithm that preserves the initial vector's direction, but increasingly changes all subsequent ones to maintain orthogonality. To minimize changes in the semantic components, it is important to operate on them first.

To allow for semantically meaningful changes to velocity models we first obtain their latent representation by using the VAE encoder, then apply a change of basis to B_{norm}. Under this base, changes to the components derived from the labels should translate to meaningful changes related to the labels. Then, we undo the change of basis operation and decode the representation back to the data space, where it can be viewed as a velocity model. Figures 15 and 16 illustrate the result of this process.

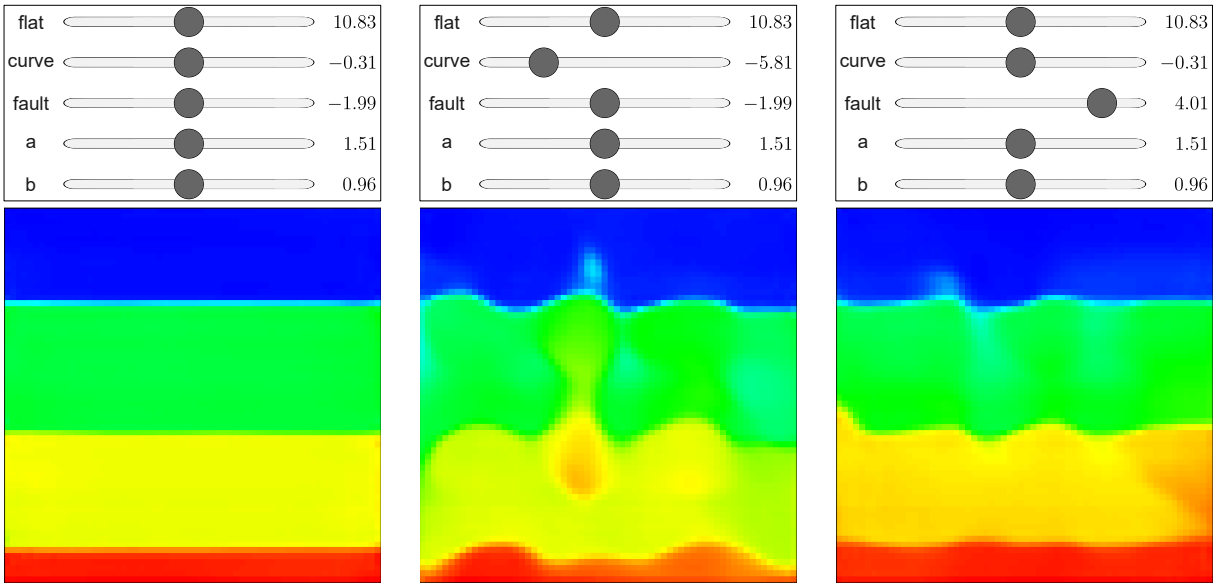

Figure 15 – Changing a flat velocity model (left) by adjusting the "curve" (middle) and "fault" (right) components. The middle of the sliders indicate the value of the components for the unaltered velocity model.

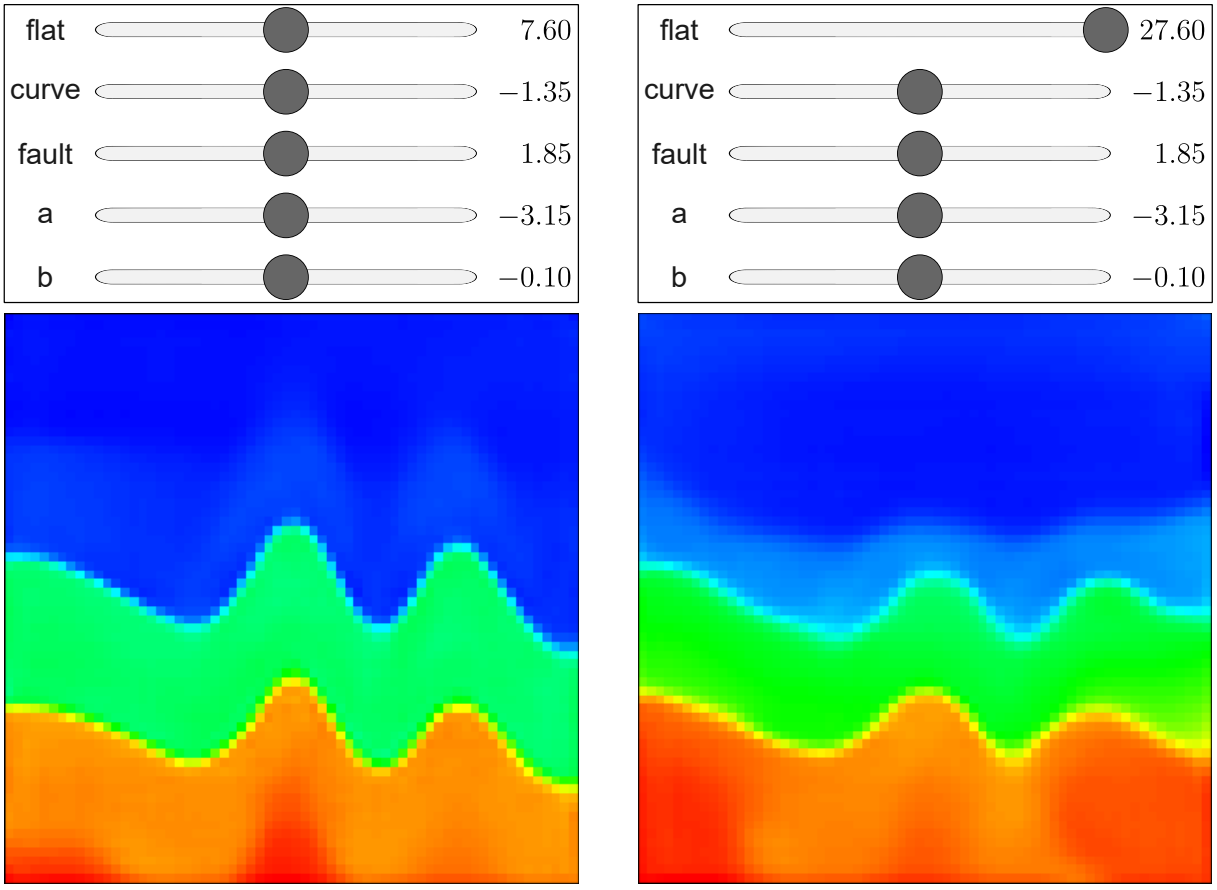

Figure 16 – Changing a curved velocity model by increasing the component related to "curve". The middle of the sliders indicate the value of the components for the unaltered velocity model.

5.4 Discussion and Limitations

Although the results obtained from the orthonormal basis definitely show improvement when compared to the basic vector arithmetic, it allows only a limited level of control of the desired characteristics. For example, an important consideration might be a way to control the position of the desired geological features. While this is theoretically possible with the current approach, it would require specific labels indicating the position of the characteristics, something that our weak labels derived from the dataset don't provide.

The disentanglement of features is also a goal that was only partially achieved. As the orthonormality was obtained via the Gram-Schmidt process, it involved changing the original label vectors. This creates situations where a change in what is supposed to be a label component may cause collateral changes related to other labels, as is the case in Figure 15, when the "fault" component is changed and small curves are added to the model. The best result in terms of disentanglement was with the "flat" component (Figure 16), and that is because it was always the first component used in the Gram-Schmidt process, and therefore it suffered the least changes. A straightforward improvement would be to repeatedly run the Gram-Schmidt process, each time with a different label vector being the first, but this would increase the computational cost of finding a basis by 5 in our case, as we work with 5 label vectors. This would increase the time to find every new basis from minutes to hours, and would clearly not scale with the number of label vectors.

Another possible difficulty of this method is finding linearly independent semantic vectors to apply Algorithm 1, although that is much less of a concern when working with real-valued vectors, but could pose a problem if the latent representations learned by the VAE are too similar, scenario that could lead to the semantic vectors not being linearly independent.

5.5 Conclusion

In this Chapter we investigated a novel approach for velocity model "engineering" that enables the control of some geological features. It stands out from mathematical modeling techniques by enabling existing models to be adapted, as opposed to having a fixed set of parameters when created. It, however, is a computationally expensive process due to the basis extension and the Gram-Schmidt procedure, and highly reliant on having labeled data, which is definitely a consideration if useful labels are hard to acquire. Due to the fact that perfect disentanglement of the geological features was not obtained, Specific Objective 2 (SO2) was only partially achieved. As the experiments in the rest of this dissertation focus more on the creation of new velocity models, we move away from this line of velocity model engineering, however, it remains an interesting research topic due to its potential of selectively altering semantically meaningful aspects of an earth model.

Chapter 6

Evaluating Data Augmentation Strategies for Deep Learning-Based Seismic Inversion

In traditional deep learning workflows, data augmentation provides a solution to the data scarcity problem: by creating new samples, it can expand the size and diversity of the training set, improving model robustness and reducing overfitting. However, for seismic inversion, it is an expensive procedure. Generating labels for new earth models via an augmentation technique requires a full recalculation of the corresponding seismic data, meaning numerically solving Partial Differential Equations (PDEs) to simulate wave propagation. This discourages its application when developing DLI solutions, leading to a multitude of works not using a staple practice of modern deep learning (WU; LIN, 2018; ZHANG; LIN, 2020; ZHU et al., 2023).

This chapter aims to evaluate the benefit of data augmentation to existing DLI methods by conducting a systematic, comparative study of data augmentation strategies for deep learning-based seismic inversion. We investigate three distinct strategies representing a spectrum of increasing complexity and computational cost: (1) a simple interpolation-based method, analogous to Mixup (ZHANG et al., 2017); (2) a parts-based recombination method, analogous to CutMix (YUN et al., 2019); and (3) a state-of-the-art generative approach using a diffusion model (HO; JAIN; ABBEEL, 2020).

To ensure our findings are robust, we evaluate these strategies on three diverse and widely-used DLI architectures: InversionNet, VelocityGan and FourierDeepONet. We also maintain a fixed computational budget, meaning that an increase in the amount of training data is balanced by a reduced training time. The results of our study demon-

strate that data augmentation can yield significant performance gains to existing Deep Learning Inversion methods, with some benefiting more than others. Even simpler strategies contribute positively in the majority of our experiments. The cost of generating the seismic data for the synthetic earth models is significant, but we believe it to be justified in situations where performance is the main goal.

6.1 Velocity Model Generation Strategies

We selected three distinct velocity model generation strategies of increasing complexity and computational cost. They are inspired by traditional data augmentation strategies from computer vision, namely Mixup (ZHANG et al., 2017) (Interpolation), Cutmix (YUN et al., 2019) (Masking) and Generative models (Diffusion). The main adaptation from the originals strategies was operating with labels (velocity models) instead of the inputs (seismic data), as this is an inversion problem, as well as not reusing the seismic data in any way, as each augmented velocity model needs to have its seismic data recalculated. Those changes are what motivated the alternative naming. Figure 17 illustrates each of the procedures.

6.1.1 Interpolation

Interpolation is a simple strategy to combine two velocity models (A and B) by using a parameter $\alpha \in [0, 1]$ to control the contribution of each one.

$$C = A\alpha + (1 - \alpha)B \tag{8}$$

It has the desirable property that velocity values in resulted synthetic model C will have the same range of values in A and B, but at the same time can introduce a transparency effect to the end result.

6.1.2 Masking

Masking composes a new velocity model from two initial ones (A and B) and a binary mask (M) of the same size.

$$D = A \odot M + B \odot \overline{M} \tag{9}$$

$\overline{M}$ denotes the bit-flipped version of M. As M is a binary matrix, its bit-flipped version consists of all ones turned to zeros, and all zeros turned to ones. $\odot$ denotes element-wise multiplication. This process essentially selects a subset of elements from A and B, with ones in M determining elements selected from A, and ones in $\overline{M}$ determining selected elements from B. This guarantees that D will only have elements that were previously in either A or B.

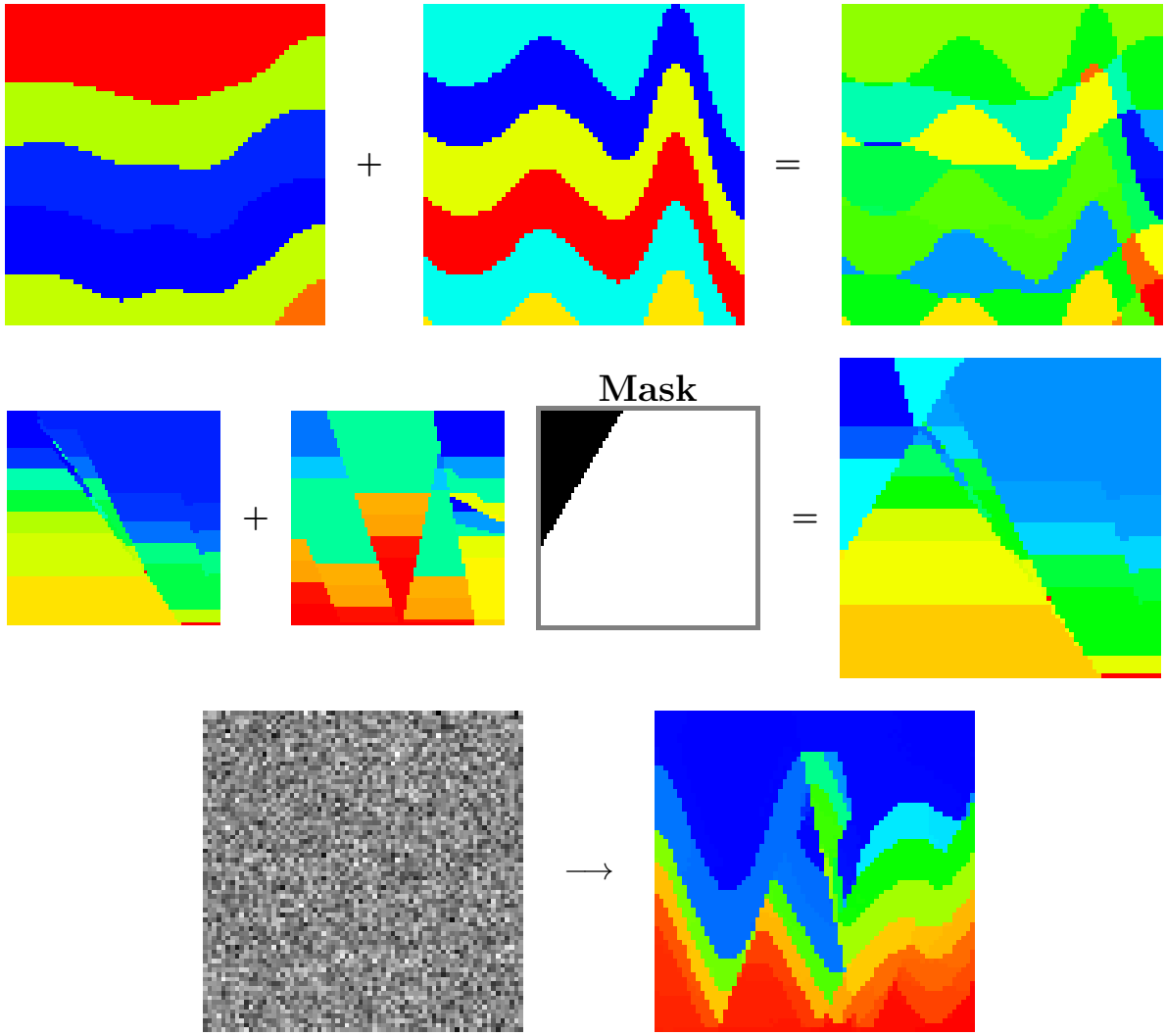

Figure 17 – Different augmentation methods. Interpolation on CurveVel-B (top), Masking on FlatFault-B (middle) and Diffusion on CurveFault-B (bottom). Samples were extracted from the datasets described in Table 3

The choice of mask has a defining influence on the final result, being able to introduce, for example, geological structures that weren't present in either the initial models, like geological faults. In this study, we only worked with masks that could be described by a line, such as the one in Figure 17.

6.1.3 Diffusion

Using Diffusion for velocity model generation consists of running inference on a previously trained diffusion model, like the ones introduced in section 2.4.3 to generate new velocity samples.

In an unconditional setting, this translates into running a fixed number of denoising steps (T) on random noise, an expensive procedure as each step requires a full forward pass on the network.

6.2 Evaluated DLI architectures

To better analyze the impact of synthetic data in deep learning based inversion, and account for a wider range of problem formulations, we decided to work with multiple networks.

A popular approach to DLI is to frame the problem as image-to-image, making heavy use of convolutional layers. InversionNet (WU; LIN, 2018) is a fully convolutional, encoder-decoder neural network capable of outperforming purely physics-driven approaches on an evaluated synthetic dataset. One variation of this approach is to introduce adversarial training in the inversion process, where a discriminator network is introduced to distinguish the velocity models generated by the data-driven inversion from the ground truth. In this setting, the inversion process can be interpreted as a conditional generative task, where the generation of subsurface velocities is conditioned by the seismic data. VelocityGAN (ZHANG; LIN, 2020) follows this framework and delivers an architecture that outperforms physics-based methods in some scenarios, while having generalization capabilities. Another work that tries to improve the generalization of DLI is FourierDeepONet (ZHU et al., 2023). The authors employ fourier blocks (WEN et al., 2022) and incorporate seismic data parameters into the network architecture to create a method that's robust to changes and interferences in the seismic data, while still being competitive with both InversionNet and VelocityGAN under normal conditions.

We selected the three aforementioned inversion architectures for our experiments due to them being different approaches that deal with 2D velocity models of comparable dimensions with open source code.

6.3 Experimental setup

To evaluate the effectiveness of the data augmentation strategies, we used as a baseline OpenFWI (DENG et al., 2023), an open collection of datasets with a large number of pairs of synthetic 2D velocity models and their respective seismic data, suitable for DLI. The full collection is subdivided into datasets representing different subsurface structures (interfaces, faults, folds) each with an easy (A) and hard (B) version. Datasets CurveVel-B (CVB), CurveFault-B (CFB), FlatFault-B (FFB), Style-A (STA) and Style-B (STB) are used in the experiments. The total number of samples and geological structures in each dataset is shown in Table 2.

We then used the generation strategies from section 6.1 to double the amount of velocity models in each dataset. This translates into each technique being used to generate 240,000 new samples.

To choose the inputs for Interpolation and Masking, we randomly selected two samples without replacement from the same dataset. For the Diffusion strategy, we trained the

Table 2 – Number of velocity model/seismic data pairs and type of geological structures in each of the OpenFWI datasets used for the experiments

Dataset	# Training	# Validation	Structures	Origin
CVB	24,000	6,000	Folds	Math modeling
CFB	48,000	6,000	Folds, Faults	Math modeling
FFB	48,000	6,000	Faults	Math modeling
STA	60,000	7,000	-	Deep learning (style transfer)
STB	60,000	7,000	-	Deep learning (style transfer)
Total	240,000	32,000		

Denoising UNet specified in chapter 4 for 100 epochs for each dataset, with number of timesteps $T = 1000$. Diffusion training used a V100 GPU and the total process of training all networks took 49h 33m. Generation for Interpolation and Masking ran on Ryzen 5900X CPU, while Diffusion generation used a V100 GPU. The time and settings to generate the augmented data is shown in Table 3. To speed up the generation process for the Diffusion strategy, we decided to reduce the number of timesteps to $T = 200$ during generation, after confirming the produced models had no noise artifacts.

Table 3 – Time to generate augmented datasets

Strategy	Time to generate	Parameters
Interpolation	10.975s	$\alpha \in [0.3, 0.7]$
Masking	22.23s	Mask area $> 10\%$
Diffusion	49h 15m	$T = 200$

To obtain seismograms from the synthetically generated velocity models, we used finite-difference forward modeling, following Deng et al. (2023), Zhu et al. (2023)'s GPU implementation using Python, Pytorch and Numpy. This step took the same amount of time for each strategy, running in 4h 30m on an RTX 3060.

6.4 Results

The augmented version of each dataset had twice the number of samples as the original baseline datasets. To ensure a fair comparison in terms of computational budget, the training on augmented datasets was conducted for half the number of epochs as the baseline training. Specifically, InversionNet and FourierDeepONet were trained for 240 epochs on baseline data and 120 epochs on augmented data, while VelocityGAN was trained for 480 and 240 epochs, respectively. Each training run of InversionNet and FourierDeepONet took a single day, while training VelocityGAN took 2 days; all on 2 H100 GPUs. The performance of all models was evaluated using three standard metrics for

DLI: Mean Absolute Error (MAE), Mean Squared Error (MSE) and Structural Similarity (SSIM). The results are displayed in Table 4.

Table 4 – Quantitative results for different models and data augmentation strategies. The best performance for each metric within a dataset and architecture is highlighted in bold. Arrows indicate whether a lower (↓) or higher (↑) value is better.

		InversionNet			**VelocityGAN**			**FourierDeepONet**		
Dataset	**Strategy**	MAE ↓	MSE ↓	SSIM ↑	MAE ↓	MSE ↓	SSIM ↑	MAE ↓	MSE ↓	SSIM ↑
CFB	Baseline	0.1414	0.0494	0.6471	0.1687	0.0642	0.6186	0.1711	0.0669	0.5954
	Interpolation	0.1275	0.0414	0.6765	**0.1496**	0.0508	**0.6537**	0.1595	0.0571	**0.6319**
	Masking	**0.1193**	**0.0374**	**0.6939**	0.1532	0.0548	0.6202	0.1885	0.0761	0.6064
	Diffusion	0.1293	0.0427	0.6763	0.1509	**0.0497**	0.6465	**0.1553**	**0.0555**	0.6291
CVB	Baseline	0.1096	0.0571	0.7450	0.1491	0.0799	0.6857	0.1401	0.0852	0.6772
	Interpolation	**0.0993**	**0.0486**	**0.7717**	0.1420	0.0704	0.6926	**0.1221**	**0.0682**	**0.7155**
	Masking	0.1051	0.0507	0.7616	**0.1376**	**0.0683**	**0.7062**	0.1276	0.0714	0.7047
	Diffusion	0.1079	0.0532	0.7615	0.1441	0.0706	0.6916	0.1259	0.0711	0.7075
FFB	Baseline	0.0831	0.0219	0.7786	**0.0987**	0.0269	**0.7545**	0.1077	0.0329	0.7398
	Interpolation	0.0732	0.0178	0.8073	0.1011	**0.0262**	0.7517	0.0882	0.0235	0.7860
	Masking	**0.0686**	**0.0163**	**0.8149**	0.1242	0.0341	0.7276	**0.0808**	**0.0208**	**0.7900**
	Diffusion	0.0742	0.0184	0.8037	0.1156	0.0317	0.7479	0.1032	0.0295	0.7666
STA	Baseline	0.0635	0.0094	0.9005	**0.0695**	**0.0122**	**0.8776**	**0.0857**	**0.0171**	0.8715
	Interpolation	0.0802	0.0149	0.8964	0.1889	0.0681	0.6837	0.1429	0.0458	0.8190
	Masking	**0.0514**	**0.0069**	**0.9200**	0.1306	0.0414	0.8301	0.1032	0.0268	0.8718
	Diffusion	0.0583	0.0079	0.9140	0.0836	0.0154	0.8725	0.0971	0.0206	**0.8811**
STB	Baseline	0.0640	0.0098	0.7586	**0.0632**	**0.0094**	**0.7638**	**0.0683**	**0.0105**	**0.7835**
	Interpolation	0.0548	0.0078	0.7840	0.1238	0.0274	0.6896	0.1036	0.0228	0.7410
	Masking	0.0589	0.0082	0.7793	0.2159	0.0794	0.5477	0.0975	0.0188	0.7763
	Diffusion	**0.0486**	**0.0062**	**0.8008**	0.0663	0.0108	0.7428	0.0998	0.0203	0.7794

Our primary finding is that data augmentation provides a substantial performance benefit for DLI across a wide range of conditions. Table 5 provides a high-level summary of the experiments, indicating the best performing strategy taking SSIM as the primary metric. In most experimental configurations, at least one augmentation strategy resulted in improved performance when compared to the same architecture trained with original data. The prevalence of augmentation strategies in the table underscore their general effectiveness. At the same time, the choice of an optimal strategy is not universal, and highly dependent on both the DLI architecture and the geological characteristics of the data. For example, for the InversionNet architecture, the Masking strategy provides the best SSIM for CFB and FFB, while Diffusion performs better on the more complex STB dataset. For FourierDeepONet, the simpler Interpolation method is often the most effective. This refutes the assumption that the computationally complex method would be best in all scenarios.

When factoring in the time to generate synthetic models from Table 3, the computationally inexpensive methods of Masking and Interpolation had the best return on investment. This suggests that the simple, fast augmentation techniques can represent a

Table 5 – Best data generation strategy for each model and dataset using SSIM as a main metric. "Baseline" indicates no augmentation surpassed the original model's performance.

Dataset	InversionNet	VelocityGAN	FourierDeepONet
CFB	Masking	Interpolation	Interpolation
CVB	Interpolation	Masking	Interpolation
FFB	Masking	Baseline	Masking
STA	Masking	Baseline	Diffusion
STB	Diffusion	Baseline	Baseline

highly practical approach for improving DLI performance, with most of the time being spent calculating the forward wave propagation of the generated data. The Diffusion strategy had great performance for InversionNet on the complex, style-transfer-generated STB dataset, improving the SSIM from 0.7586 to 0.8008. This suggests that the use of such a costly strategy may be justified in situations where the underlying data distribution is complex and difficult for simpler methods to replicate convincingly.

The DLI architectures also responded differently to the augmentation strategies, reflecting their underlying mechanisms.

InversionNet, a fully convolutional network, consistently benefited from all augmentation strategies across all datasets, indicating that DLI methods mostly reliant on learning spatial hierarchies of features benefit from introducing augmentation.

FourierDeepONet, which introduces operations in the frequency domain, was less consistently improved by augmentation, seeing significant uplifts in some situations, such as Masking on FFB boosting the SSIM score from 0.7398 to 0.7900, and significant downgrades in others, such as Interpolation on STA dropping SSIM from 0.8715 to 0.8190. Considering that, the results indicate that the Interpolation strategy should be applied with caution to DLI methods making use of fourier blocks. It provided a substantial increase in performance when the expected geological structures in the velocity models are a combination of faults, folds and flat layers, but the other augmentation strategies also provided decent gains in these circumstances, without a sharp decrease when the more irregular models from STA and STB were introduced, making them a more stable choice.

VelocityGAN saw performance gains in CFB and CVB, but some strategies significantly hindered the performance in the other datasets, with none offering substantial benefit in those cases. This suggests that data augmentation is not a straightforward procedure when dealing with DLI methods that make use of the adversarial framework. As the training process of adversarial networks is notoriously unstable(METZ et al., 2016), adding more data in the form of augmented samples shifts the original data distribution and might be sufficient to require adjustments in the network's hyperparameters for optimal performance. Making hyperparameters adjustments to the worst performing strategy for VelocityGAN on STA (Interpolation), we were able to achieve a result that surpasses

our best Baseline across all evaluated metrics, as shown in Table 6. The adjustments consisted in lowering the learning rate and increasing the number of updates of the discriminator in relation to the generator. Augmentation can benefit adversarial-based DLI methods, but may require additional effort in tuning an already existing model, something that's not required for the other DLI methods in our experiments.

Table 6 – Comparison between VelocityGAN Baseline and Tuned Interpolation. Arrows indicate whether a lower (↓) or higher (↑) value is better.

Strategy	**MAE ↓**	**MSE ↓**	**SSIM ↑**
Baseline	0.0695	0.0122	0.8776
Interpolation	**0.0648**	**0.0104**	**0.8832**

Figure 18 shows inversion results for InversionNet trained with each of the augmentation strategies compared to the baseline. The augmented variations show improvements in terms of reconstructed features, as demonstrated in better recovering the dome-shaped formation in the CFB example and the central flat layers in the FFB example. The CVB example demonstrates a more precised velocity (color) reconstruction for the augmentation models when compared to the baseline. STA and STB examples also show improvements in reconstructing the main structures of the ground truth. Overall, the visuals corroborate the quantitative findings that data augmentation has a strong positive effect on the InversionNet architecture.

As a last observation, it is worth noting that with data augmentation the models were trained with half number of epochs than the baseline. This setup assumes that computing budget is the constraint. However, in a scenario where data (not computing) is the constrained resource, using augmented data with the same number of training epochs as the baseline is likely to produce better results.

6.5 Conclusion

In this Chapter, we directly address Specific Objective 3 (SO3) by evaluating how different methods for generating synthetic earth models —ranging from simple Interpolation and Masking to a more complex diffusion model —impact the performance of a variety of DLI architectures, namely InversionNet, VelocityGAN and FourierDeepONet. The results indicate a positive impact of the data augmentation strategies in the majority of the evaluated scenarios, with InversionNet having the strongest response. FourierDeepONet showed an overall positive response, with a few scenarios having the augmented data degrade performance. VelocityGan saw mixed results, with an increase in performance in only two out of the 5 evaluated datasets. However, further tuning the model to the augmented dataset proved able to realize some of the performance gains.

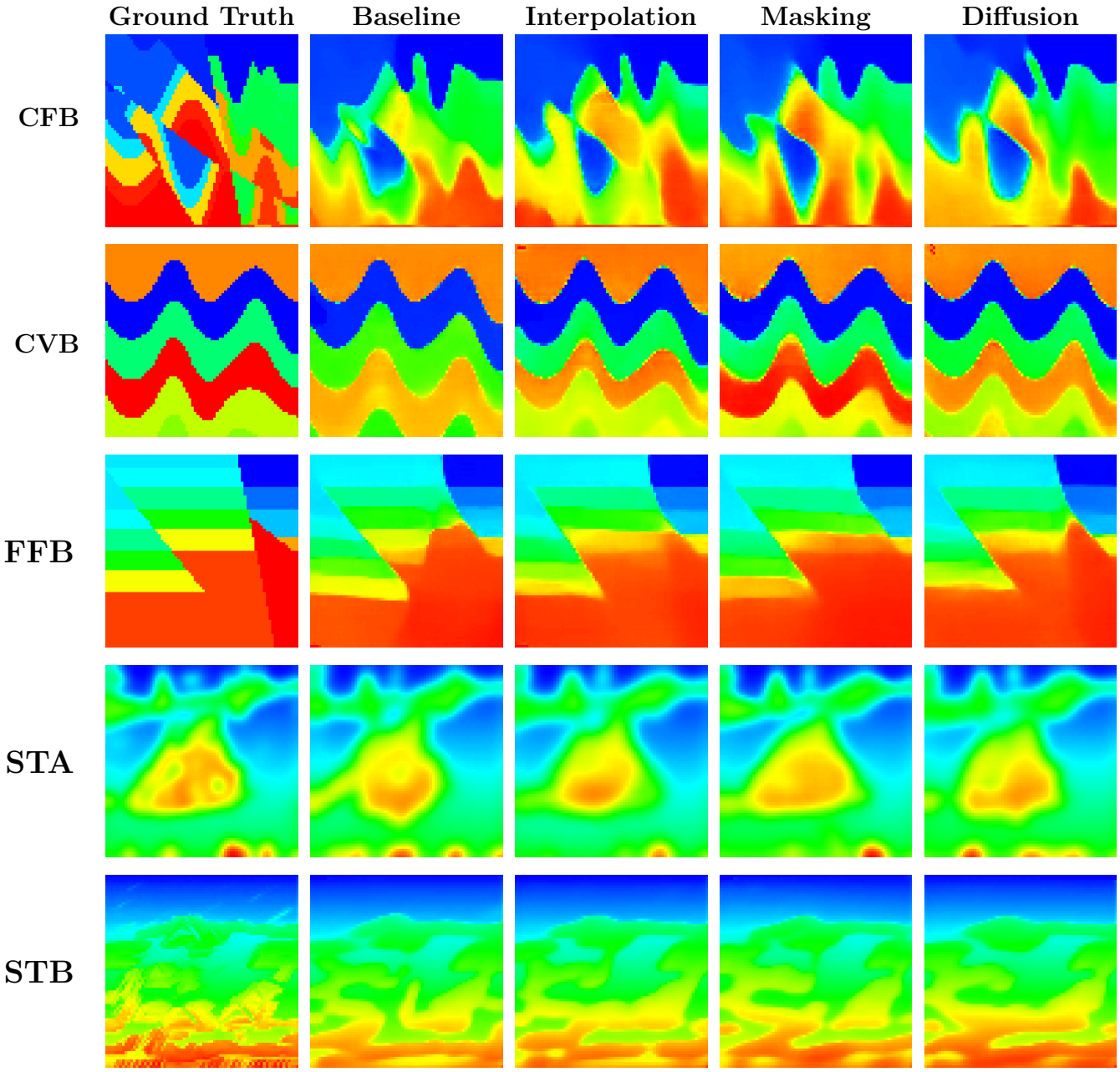

Figure 18 – Qualitative comparison of inversion results for multiple samples using the InversionNet architecture. Each row corresponds to a validation sample from one of the datasets used in the experiments.

Regarding the evaluated data-generation strategies, the computationally cheaper techniques demonstrated a great return on investment, while the more expensive diffusion proved better suited and stable for generating velocity models in situations with irregular formations and more intricate detail. Overall, the results of the experiments indicate a positive outlook for synthetic data for DLI, which is further explored in the following Chapter.

Chapter 7

Improving training strategies for DLI: A case study on the Kaggle GWI competition

In computer vision, the success achieved by deep learning solutions is not only due to changes in the architecture of neural networks, but also from advancements in training strategies (BELLO et al., 2021; WIGHTMAN; TOUVRON; JÉGOU, 2021). As one popular formulation for DLI is as an image-to-image problem (a common setting in computer vision), it stands to reason that it should also benefit from those advancements. Much of the research effort in DLI, however, is directed to finding alternative architectures to solve seismic inversion, with few works focusing on the more practical issues of training these methods. This discrepancy between fields is partially due to the lack of open seismic datasets, which hinders the process of reproducing results; in contrast, open natural image datasets are in abundance. Recent initiatives such as the OpenFWI dataset (DENG et al., 2023) and the Kaggle GWI competition (LIN et al., 2025) are a step towards reproducible research and allow for such investigation. In this chapter we detail our DLI solution to the Kaggle GWI competition, capable of achieving state-of-the-art performance on OpenFWI, heavily relying on general, modern practices to improve DL performance.

7.1 Geophysical Waveform Inversion Competition

The Kaggle GWI competition (LIN et al., 2025) poses a standard seismic inversion challenge: estimating subsurface properties (velocity maps) from seismic waveforms. Pro-

posed solutions were ranked by their MAE on a closed test set, for which only seismic waveforms were available. The data follows the same generating process as the 2D synthetic part of OpenFWI, and as a result, using OpenFWI as the main data source was a common strategy among competitors.

Although Kaggle is a machine learning-focused platform, using machine learning solutions was not a requirement, allowing for physics-based approaches. The volume of test data, however, made a purely physics-based solution impractical, as it would require processing the seismic data from more than 60,000 seismic surveys, a challenging task even with simple acquisition geometry, due to the limited time frame of 90 days. At the end, every published top solution used machine learning in some form, highlighting that under favorable circumstances, that is, an abundance of training data, machine learning models are extremely competitive, even when the main consideration is inversion quality, a characteristic generally associated with traditional physics-based methods.

7.2 A standard network for inversion - InversionNet

To introduce our final solution for the competition, we gradually change a well established model for DLI: InversionNet

InversionNet is a CNN with an encoder-decoder structure. Although CNNs are commonly applied to image data, their inductive bias is significantly more general, implying locality and translation equivariance (GOODFELLOW; BENGIO; COURVILLE, 2016). This broadens their range of applicability to any domain where it is reasonable to assume that spatially close features are correlated, and the presence of a feature is more important than its exact location. This is the case when working with seismic waveforms, where the nature of wave propagation establishes a dependency between neighboring cells in a wavefield, and reflection patterns in seismograms suggest the presence of specific geological structures. In InversionNet, the encoder is responsible for learning high-level features, while the decoder translates those features to the desired velocity models. The original model proposes adding a CRF head to the decoder, but the gains in terms of MAE are marginal, so we omit it for simplicity. Figure 19 illustrates the network architecture.

One challenge that neural networks for inversion have to tackle is the extreme asymmetry of the seismic data. Hydrophones can have a high sample rate, that translates to thousands of records in a matter of seconds. This is much higher than the number of hydrophones, creating data matrices where one dimension is in the scale of dozens (number of hydrophones or receivers), where the other is in the scale of thousands (number of records). InversionNet handles this by employing convolutions with asymmetric kernels that gradually transform the input data into a "square" representation that is then fed to the decoder. Apart from the unconventional kernels, InversionNet remains fairly standard as a CNN.

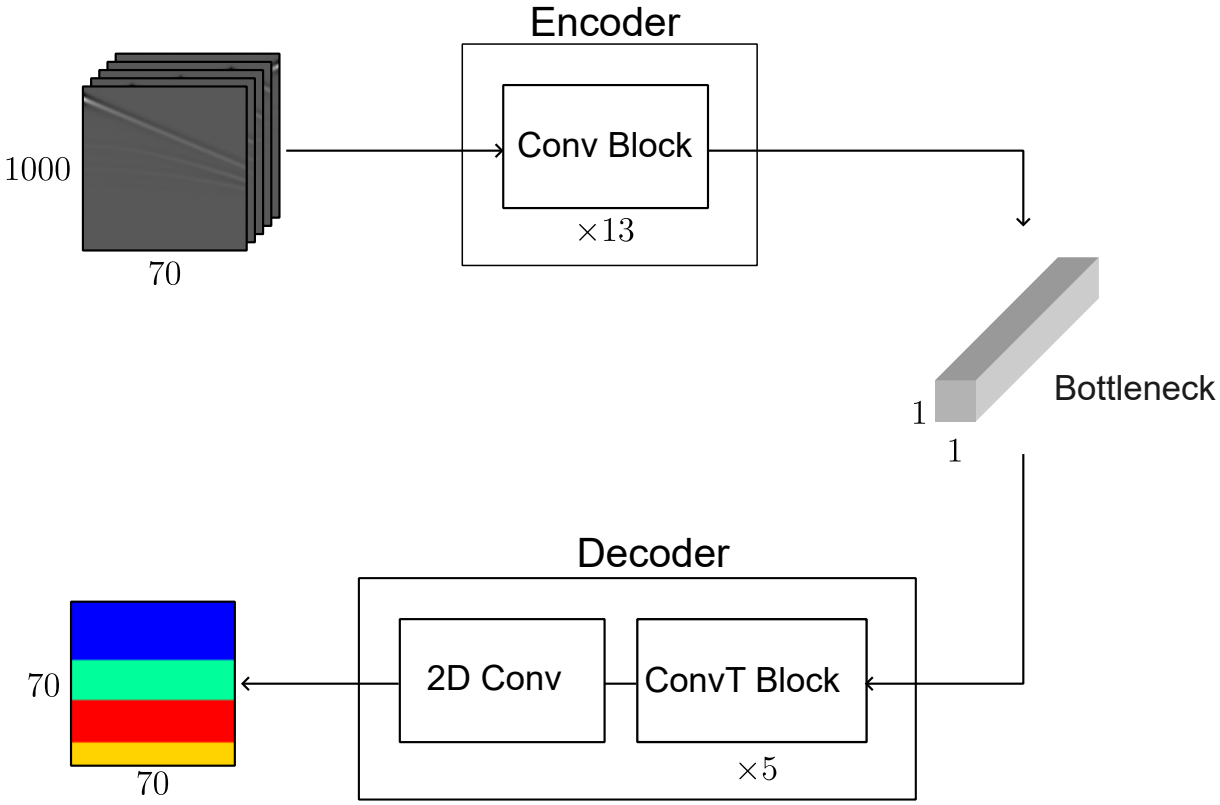

Figure 19 – Diagram illustrating the InversionNet architecture. It has two main components: an Encoder composed of 13 convolution blocks that gradually convert the input into a 1×1 representation and a Decoder composed of 5 Deconvolution blocks followed by a 2D convolution that transform the intermediate representation to a 70×70 spatial size. In the original architecture from (WU; LIN, 2018), it incorporated a CRF head to process the decoder output, however the performance improvement wasn't substantial.

7.3 Incorporating modern tricks

InversionNet is a very straightforward architecture, and even though it performs well for inversion (as seen in Chapter 6), it doesn't employ some common DL training strategies and architectural components that are generally beneficial, such as transfer learning, data augmentation, and methods to rank feature importance. This section elaborates on the changes that were incorporated when creating our solution to the GWI competition.

7.3.1 Transfer learning

Transfer learning is a cornerstone of applied DL, allowing for a previously trained model to be employed on a new task, effectively reusing computational time. However, it requires some adaptation in cases where the original task and the new task expect inputs of different sizes. If the difference is small and some distortion is acceptable, a simple interpolation to the desired size is a common solution. That being said, due to

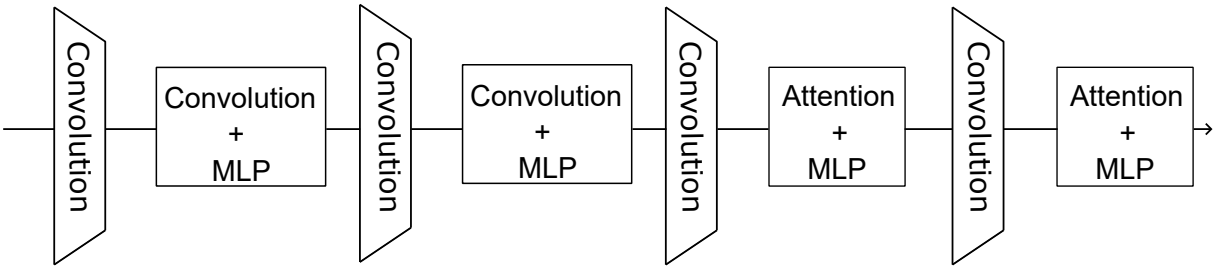

Figure 20 – CAFormer (YU et al., 2023) architecture. In the GWI solution, it was used with some modifications as the new Encoder. MLP stands for Multilayer Perceptron, a simple fully connected component. The trapezoidal blocks correspond to downsampling operations implemented with convolutions that reduce the spatial size of the features by half.

the high asymmetry of the input data for seismic inversion, simply resizing would incur a considerable distortion and loss of information: transforming a 1000×70 input into 224×224 (a common size for image models). Altering the structure of the previously trained model to handle different inputs is an option, but one can end in a situation where most of the transferred weights have to be replaced by newly initialized ones for compatibility reasons. Therefore, to make the most out of transfer learning, the adaptation process should favor changes that preserve the transferred weights as much as possible. As an example, to deal with different input sizes in convolutional layers, prefer changes to the stride and padding over changes to the kernel size, as the former only changes how the weights are operated with the input while the latter changes the size of the parameter matrix, requiring reinitialization.

For the GWI challenge, we replaced the InversionNet encoder with a CAFormer (YU et al., 2023) model trained on ImageNet with an asymmetric stride and extra input padding to handle input asymmetry. CAFormer is a modern CV architecture that combines separable convolutions and attention blocks, and works well as a feature extractor for natural images, as evidenced by its score on the ImageNet benchmark. Figure 20 illustrates at high level the CAFormer architecture.

7.3.2 UNet-like structure

UNet(RONNEBERGER; FISCHER; BROX, 2015) is a well-known architecture for solving image-to-image problems. Its defining characteristic is an almost symmetric encoder-decoder structure with skip connections between the two parts. Introduced by (HE et al., 2016), skip connections allowed neural networks to become much deeper by providing better gradient flow, avoiding the vanishing gradients problem. It changed the learning paradigm from direct mappings to residual mappings, and UNet heavily uses them to create a direct path between lower and high level features.

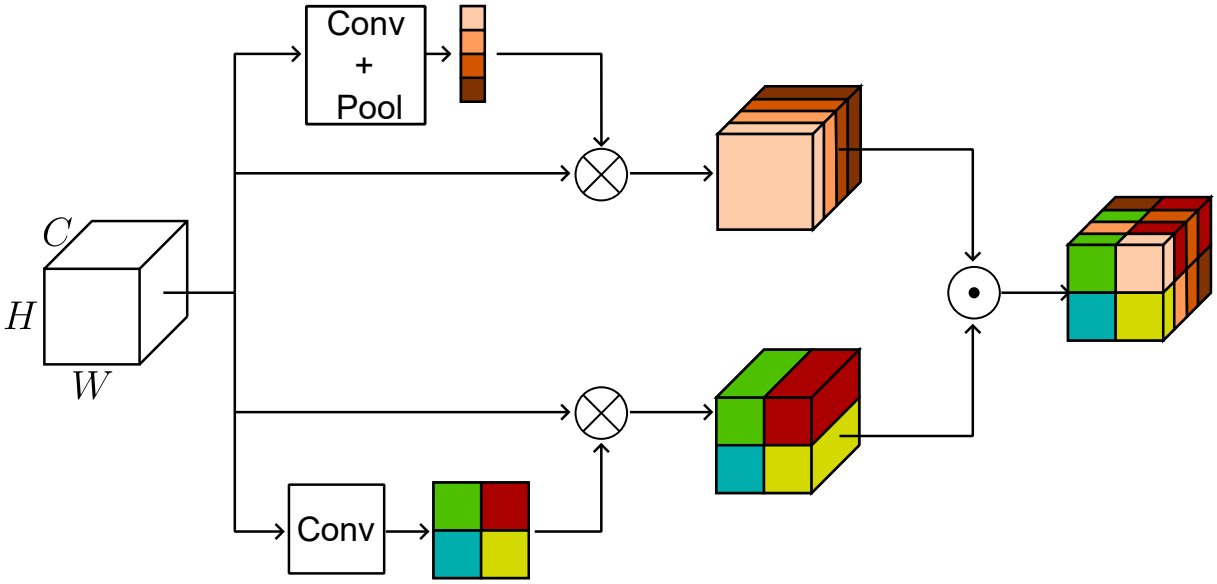

Figure 21 – SCSE block (ROY; NAVAB; WACHINGER, 2018). The top path corresponds to the channel component, while the bottom path corresponds to the spatial component. $\odot$ indicates the element-wise maximum, while $\otimes$ indicates the recalibration of the input. The spatial component squeezes the input by applying an average pool operation followed by convolutions, this is then recombined to the original input to form the component's output. The channel component squeezes the input by applying a 1×1 convolution that is then recombined to the original input to form the component's output.

Another modification to the InversionNet architecture was based on this idea to create shortcut paths from the intermediate representations of the encoder to the decoder. In our case, however, the shortcut paths were not identity maps, but two convolutional blocks. To concatenate the features channel-wise, it required modifications to the decoder, so that it matches the spatial dimensions of the encoder. The decoder was then changed so that its transposed convolutions matched the spatial size of CAFormer's convolutions, enforcing a symmetric structure.

7.3.3 SCSE blocks

SCSE blocks (ROY; NAVAB; WACHINGER, 2018) are small, efficient components that allow the network to weight the contribution of channels and spatial sections. This enables the model to ignore less important features and emphasize important ones. Each SCSE block has a spatial and a channel component, with the spatial component squeezing spatial features and recalibrating (exciting) the channels, and the channel component squeezing along the channels to recalibrate spatial features. The output of the SCSE block is the element-wise maximum of the output of each component. Figure 21 illustrates an SCSE block.

The mechanisms of feature importance from SCSE blocks add functionality similar to that of transformers, and the cost of incorporating them into an existing architecture is relatively inexpensive, involving only a few convolutional layers, that are generally efficient in terms of parameters. The fact that they can be configured to conserve spatial dimensions and channels of the input facilitates their integration. In the final architecture, they were incorporated in the decoder layers.

7.3.4 Data Augmentation

A central piece of the final solution was the heavy use of data augmentation. As the GWI competition encompassed all 2D datasets from OpenFWI, and initial experiments revealed that they closely followed the proportions in the original dataset, we trained a single diffusion model on the entire OpenFWI 2D data, and used it in conjunction with the forward wave propagation method from Zhu et al. (2023) to generate "velocity model"/"seismogram" pairs. We repeated this process until we quintupled the amount of data available, and used this enhanced dataset to train our solution.

After a qualitative analysis of inversion results, we trained and used specialized diffusion models as in Chapter 4 to generate a second set of synthetic data containing only samples from the datasets that had lower MAE score, that is CurveVel-B, CurveFault-B, FlatFault-B, Style-A, and Style-B. This second set had approximately the same amount of samples as the original OpenFWI training data, and was used for fine tuning the final model with a smaller learning rate. As validation, a subset of the original OpenFWI validation split was used.

7.4 Final architecture results and discussion

The final architecture incorporated all the changes mentioned in the previous sections, and is illustrated in Figure 22. Additional training strategies to increase stability and performance for longer runs, such as Exponential Moving Average (EMA) on the weights, input normalization and learning rate scheduling were also employed. Training the final architecture took a total of 30 days, distributed on a cluster with 4 NVIDIA V100, and the model has 1.23×10^8 parameters. As full training runs are prohibitively expensive, evaluating the effect of the individual changes to the initial architecture isn't feasible. We instead present in Table 7 the evolution of MAE on the competition's test set during model training, highlighting changes that caused a significant decrease in MAE. The MAE score for submissions in the GWI competition is calculated using non-normalized velocity values, that is, between 1500 and 4500. This differs from the OpenFWI benchmark that uses values in the range $[0, 1]$, causing competition MAE values to seem much larger when compared, for example, with the results from Chapter 6.

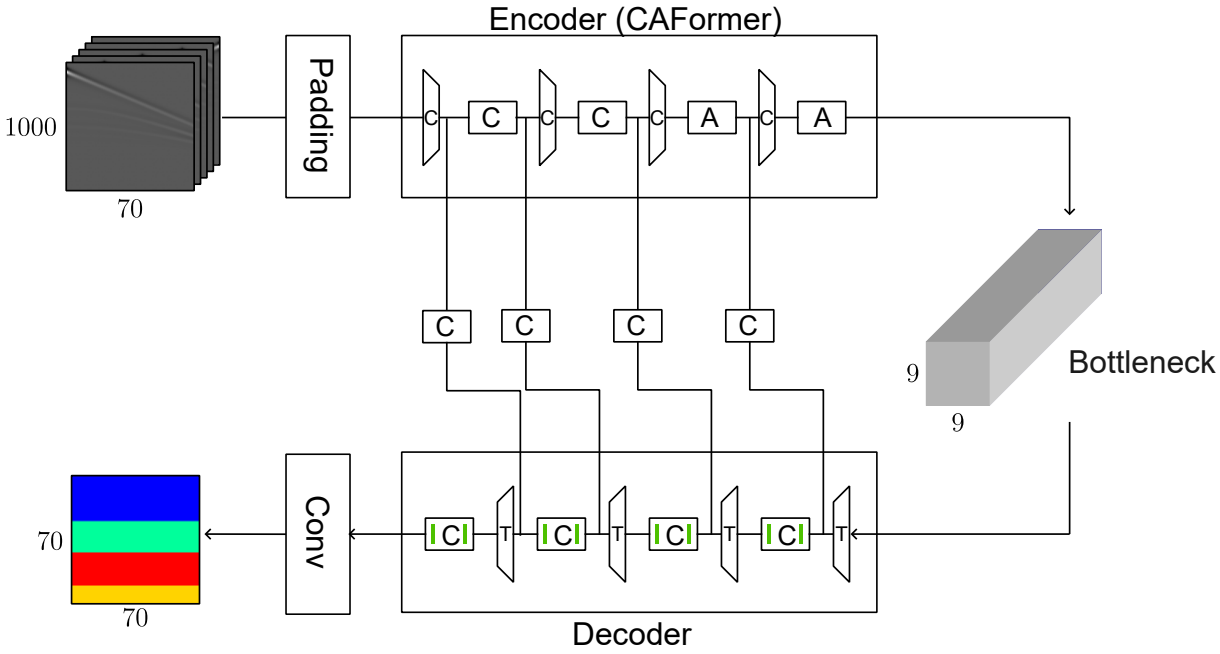

Figure 22 – Final network architecture for the Kaggle GWI competition. Blocks marked with a letter "C" indicate convolutional blocks, "A" attention and "T" transposed convolutions. Trapezoidal blocks indicate downsampling operations in the encoder and upsampling in the decoder. The green marks indicate the presence of an SCSE block

Table 7 – MAE on the Kaggle GWI competition of the final architecture at various stages of training. A baseline using InversionNet is included for comparison. The InversionNet models have 2.4×10^7 parameters and each trained for about a single day on an NVIDIA H100. Synth4 refers to the first batch of synthetic data created using diffusion models, and Hard1 refers to the second batch including only samples from the "hard" classes of OpenFWI.

DLI Model	**Data**	**Epochs**	**MAE ↓**	**Observation**
InversionNet	OpenFWI	200	107.78	Estimated score if an appropriate model was chosen for each test sample
Final Arch	OpenFWI	150	28.81	
Final Arch	OpenFWI + Synth4	40	24.54	
Final Arch	OpenFWI + Synth4	65	20.70	Learning rate decrease
Final Arch	OpenFWI + Synth4 + Hard1	80	**19.40**	Fine tuning on hard samples

To obtain a baseline of comparison with existing models seen in literature for OpenFWI, we estimate the competition MAE of an ensemble of dataset-specific InversionNets by assuming that the competition test data follows the same proportions as OpenFWI, that is, those stated in Table 1. We can only work with an estimation because informa-

tion concerning which family a test sample belongs to was never disclosed, so we can't guarantee that the correct InversionNet model was selected for a given test sample.

Further improvements on the 19.40 MAE model were applied, such as incorporating samples derived from the predictions of the test set in the training data. These techniques resulted in significant improvement, reducing MAE to a low of 12.4; however, as they are specific to the circumstances of a competition, meaning they undermine a fair evaluation of the model performance, we won't discuss them in detail here.

For a qualitative evaluation of results, we present Figure 23, that shows a comparison between the final architecture and the ensemble of InversionNets. It is evident that the quality of reconstruction of subsurface shapes is much higher for our final architecture, with it being able to accurately recover small details and flat layers, even in the presence of geological faults (see CFB and FFB). The style datasets still prove challenging, with most of the intricate structure of STB being lost. The general structure of the deeper layers, however, is better represented, and the shallower details that are missing in InversionNet's result are identified. Despite the improvement, there are still some mistakes, such as it failing to accurately represent the velocity values of the deepest layer of the CVB sample.

Including all optimizations, the final score of the model was 12.40, placing it in the top 1%. Out of the published top solutions, all but one included synthetic data strategies, reinforcing their importance and effectiveness for DLI methods.

7.5 Conclusion

This Chapter detailed the development of a state-of-the-art DLI network, resulting in a top 1% finish in the international Kaggle GWI competition, and successfully addressing Specific Objective 4. The process served as a practical case study, demonstrating that the principles of modern deep learning, particularly those from computer vision, can be successfully adapted to solve complex geophysical inverse problems. It also showcases the ability of data methods to further improve existing solutions, with the addition of synthetic data reducing the Mean Absolute Error (MAE) by approximately 33%, from 28.81 to 19.40. We conclude this section on a note of gratitude to all participants of the Kaggle GWI competition for the helpful discussions on the forums, where many of the ideas presented here were first introduced, and hope that it serves as an inspiration for similar initiatives in the future.

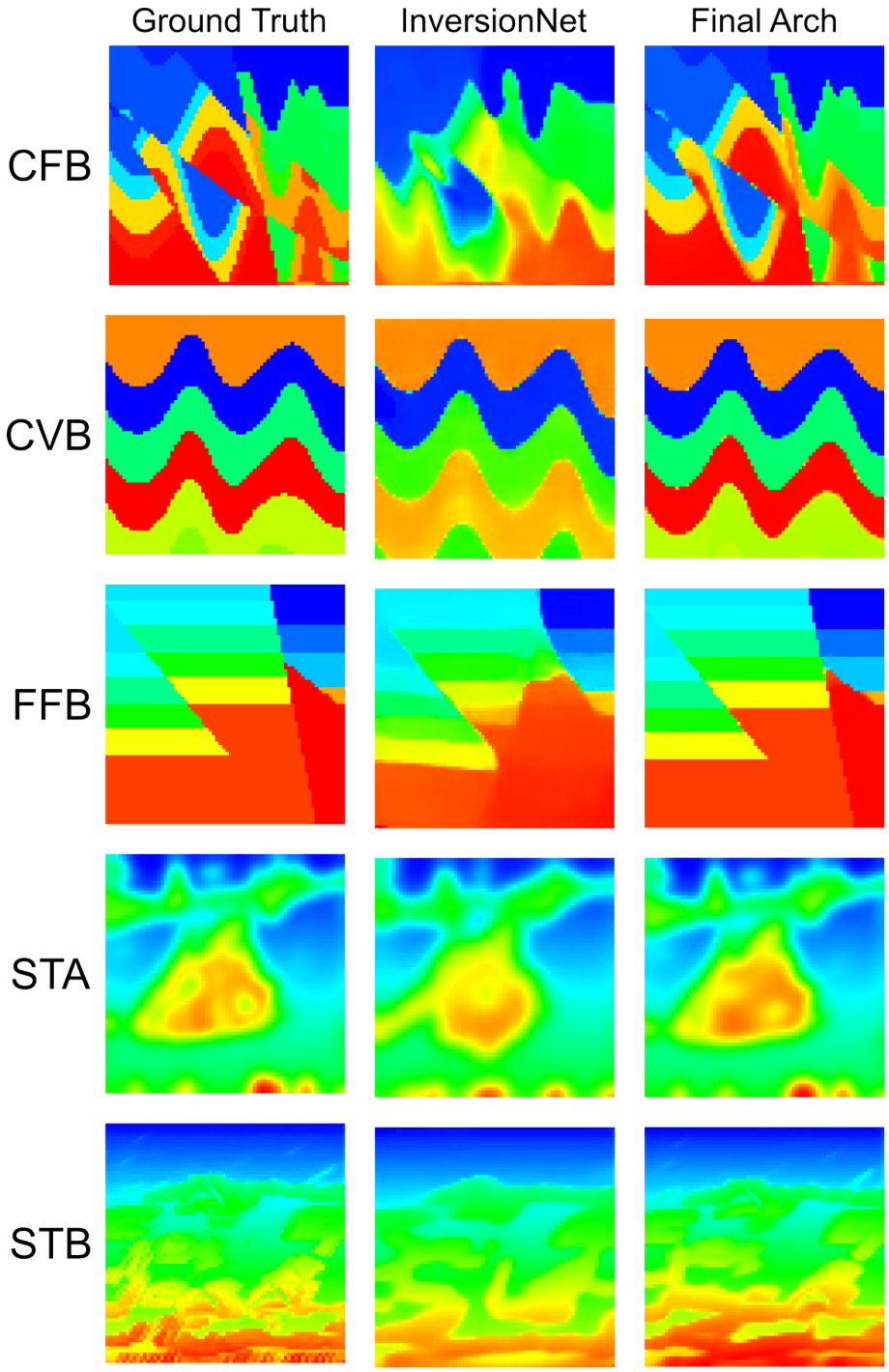

Figure 23 – Qualitative comparison between ensemble of InversionNets and final architecture on OpenFWI data. The dataset of origin of each velocity model is indicated on the left side. From top to bottom: CurveFault-B, CurveVel-B, FlatFault-B, Style-A, Style-B

Chapter 8

Conclusion

This work provides a study and evaluation of the effectiveness of synthetic data strategies to improve Deep Learning Inversion (DLI) methods, an important area of research in seismic inversion, that is often underexplored in favor of studies focusing on new methods or architectures. We demonstrate that unconditional diffusion models are able to successfully generate velocity models that capture the structural properties of existing datasets with a minor presence of artifacts. Following, we propose a novel approach using Variational Autoencoders (VAEs) to enable direct manipulation of desired geological characteristics. While perfect disentangled semantic control was not achieved, the method demonstrated a viable path toward manipulating features of interest, given enough labeled data.

With the capability to create velocity models containing complex structures established, we evaluate their utility across a range of different architectures for DLI. We showed that the best choice of augmentation method is highly dependent on the chosen architecture, with convolutional networks showing great response to all augmentations evaluated, while only partially beneficial for methods with Fourier blocks and adversarial training regimes. Even simpler augmentations were able to provide substantial improvements in the evaluated metrics, with the main benefit of more complex strategies being their ability to convincingly replicate irregular structures from the more complex Style datasets. These findings are then used to craft a gold medal solution to the Kaggle Geophysical Waveform Inversion (GWI) competition, showcasing the power of data strategies combined with modern neural network building techniques in an environment without computational constraints.

We argue for the importance of data and data generating methods for DLI workflows, but to ensure a balanced perspective, it is also important to recognize the limitations

of our study. For starters, all our experiments were conducted using 2D synthetic data from the OpenFWI (DENG et al., 2023) benchmark. While suitable for DLI due to the vast number of available samples, it may not be representative of the real-world, leaving grounds for exploration on how effective data strategies are with field data. Future work could therefore investigate domain adaptation techniques to bridge this "sim-to-real" gap and create models robust to the complexities of field data. Furthermore, the focus on 2D problems highlights a possible next step: scaling these generative strategies to the 3D domain. This is a non-trivial challenge that will require exploring memory-efficient architectures to make working with 3D velocity models computationally tractable. The high computational cost of the diffusion-based methods used in this work could also be an avenue of future research by investigating more efficient diffusion architectures, enabling the cheap generation of complex subsurface structures.

In conclusion, this work has demonstrated that the strategic synthesis of data is a powerful asset for advancing deep learning-based seismic inversion. The evidence presented, culminating in state-of-the-art performance in a competitive setting, affirms that by treating data not as a static resource but as a dynamic and optimizable component of the workflow, it is possible to unlock the full potential of deep learning for solving geophysical challenges.

8.1 Contributions

During the course of this Masters program I contributed to the following works, directly related to the application of deep neural networks for seismic inversion, with contributions on both the ideation and implementation steps:

- LIN, Y. et al. Yale/UNC-CH - Geophysical Waveform Inversion. 2025. Kaggle. (**Gold medal solution 12$^{\text{th}}$ place** - Team "working hard" - Final standings available at: <https://www.kaggle.com/competitions/waveform-inversion/leaderboard >)

- SOUZA, L. C. et al. PDARTS: Projected Differentiable Architecture Search for Seismic Inversion. 2025. Machine Learning. (**Published** DOI: 10.1007/s10994-025-06883-1)

- JUNIOR, C. G. G; SENGER, H.. Comparison of synthetic data generation strategies for data augmentation: A Case Study on Deep Learning-based Seismic Inversion. (In preparation)

Additionally, I also contributed to other works outside of the seismic domain, related to the development of Computer Vision solutions for defect detection in Wind Turbine Blades. My contributions were also on ideation and implementation:

- OLIVEIRA, A. R. F. et al. Effect of Images' Size and Noise on Machine Learning Classifiers to Detect Erosive Wear in Wind Turbine Blades. Wind Energy Science Conference - WESC, 2025, Nantes - France.(**Published**)
- GÓES, S. H. R et al. Visual detection of surface defects in wind turbine blades using two step image segmentation. Ibero-Latin American Congress on Computational Methods in Engineering - CILAMCE, 2025. (**Accepted**)

References

ADLER, A.; ARAYA-POLO, M.; POGGIO, T. Deep recurrent architectures for seismic tomography. In: EUROPEAN ASSOCIATION OF GEOSCIENTISTS & ENGINEERS. **81st EAGE conference and exhibition 2019**. [S.l.], 2019. v. 2019, n. 1, p. 1–5.

_____. Deep learning for seismic inverse problems: Toward the acceleration of geophysical analysis workflows. **IEEE Signal Processing Magazine**, IEEE, v. 38, n. 2, p. 89–119, 2021.

AMINZADEH, F.; BRAC, J.; KUNZ, T. (Ed.). **3-D Salt and Overthrust Models**. s.l.: Society of Exploration Geophysicists, 19. (SEG/EAGE 3-D Salt and Overthrust Models, no. 1). ISBN 978-1-56080-077-4.

Araya-Polo, M.; FARRIS, S.; FLOREZ, M. Deep learning-driven velocity model building workflow. **The Leading Edge**, v. 38, n. 11, p. 872a1–872a9, Nov. 2019. ISSN 1070-485X, 1938-3789.

ARAYA-POLO, M. et al. Deep-learning tomography. **The Leading Edge**, Society of Exploration Geophysicists, v. 37, n. 1, p. 58–66, 2018.

ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. Wasserstein generative adversarial networks. In: PMLR. **International conference on machine learning**. [S.l.], 2017. p. 214–223.

BAEVSKI, A. et al. wav2vec 2.0: A framework for self-supervised learning of speech representations. In: LAROCHELLE, H. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2020. v. 33, p. 12449–12460. Available at: <<https://proceedings.neurips.cc/paper_files/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf>>.

BELLO, I. et al. Revisiting resnets: Improved training and scaling strategies. **Advances in Neural Information Processing Systems**, v. 34, p. 22614–22627, 2021.

BILLETTE, F.; BRANDSBERG-DAHL, S. The 2004 BP velocity benchmark. In: **67th Annual Internat. Mtg., EAGE, Expanded Abstracts**. [S.l.]: EAGE, 2005. p. B035.

CAMPOS, L. R.; NOGUEIRA, P.; NASCIMENTO, E. Estimating initial velocity models for the fwi using deep learning. In: **Proceedings of the 16th International Congress of the Brazilian Geophysical Society. Brazilian Geophysical Society**. [S.l.: s.n.], 2019.

CHEN, P.; LEE, E.-J. Introduction. In: _____. **Full-3D Seismic Waveform Inversion: Theory, Software and Practice**. Cham: Springer International Publishing, 2015. p. 1–14. ISBN 978-3-319-16604-9. Available at: <<https://doi.org/10.1007/978-3-319-16604-9_1>>.

CHEN, Y. et al. Efficient and accurate mri super-resolution using a generative adversarial network and 3d multi-level densely connected network. In: FRANGI, A. F. et al. (Ed.). **Medical Image Computing and Computer Assisted Intervention – MICCAI 2018**. Cham: Springer International Publishing, 2018. p. 91–99. ISBN 978-3-030-00928-1.

CHUNG, Y.-A. et al. **W2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training**. 2021.

DENG, C. et al. **OpenFWI: Large-Scale Multi-Structural Benchmark Datasets for Seismic Full Waveform Inversion**. [S.l.]: arXiv, Jun. 2023.

DHARA, A.; SEN, M. K. Physics-guided deep autoencoder to overcome the need for a starting model in full-waveform inversion. **The Leading Edge**, v. 41, n. 6, p. 375–381, 06 2022. ISSN 1070-485X. Available at: <<https://doi.org/10.1190/tle41060375.1>>.

DHARIWAL, P.; NICHOL, A. Diffusion models beat gans on image synthesis. **Advances in neural information processing systems**, v. 34, p. 8780–8794, 2021.

FANG, Y. et al. Eva: Exploring the limits of masked visual representation learning at scale. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2023. p. 19358–19369.

GATYS, L. A.; ECKER, A. S.; BETHGE, M. A neural algorithm of artistic style. **arXiv preprint arXiv:1508.06576**, 2015.

GOODFELLOW, I. **NIPS 2016 Tutorial: Generative Adversarial Networks**. 2017. Available at: <<https://arxiv.org/abs/1701.00160>>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, Massachusetts: The MIT Press, 2016. (Adaptive Computation and Machine Learning). ISBN 978-0-262-03561-3.

GOODFELLOW, I. et al. Generative adversarial nets. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2014. v. 27.

HALEVY, A.; NORVIG, P.; PEREIRA, F. The Unreasonable Effectiveness of Data. **IEEE Intelligent Systems**, v. 24, n. 2, p. 8–12, Mar. 2009. ISSN 1541-1672.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HIGGINS, I. et al. beta-vae: Learning basic visual concepts with a constrained variational framework. In: **International conference on learning representations**. [S.l.: s.n.], 2017.

HO, J.; JAIN, A.; ABBEEL, P. **Denoising Diffusion Probabilistic Models**. [S.l.]: arXiv, Dec. 2020.

KARL, F.; SCHERP, A. **Transformers are Short Text Classifiers: A Study of Inductive Short Text Classifiers on Benchmarks and Real-world Datasets**. 2023.

KARRAS, T. et al. Training generative adversarial networks with limited data. **Advances in neural information processing systems**, v. 33, p. 12104–12114, 2020.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. **arXiv preprint arXiv:1312.6114**, 2013.

KODALI, N. et al. **On Convergence and Stability of GANs**. 2017. Available at: <<https://arxiv.org/abs/1705.07215>>.

LARSEN, A. B. L. et al. Autoencoding beyond pixels using a learned similarity metric. In: PMLR. **International conference on machine learning**. [S.l.], 2016. p. 1558–1566.

LIN, L. et al. Marine 3d seismic volumes from 2d seismic survey with large streamer feathering. **Marine Geophysical Research**, v. 40, p. 619–633, 2019.

LIN, T. et al. Microsoft COCO: common objects in context. **CoRR**, abs/1405.0312, 2014. Available at: <<http://arxiv.org/abs/1405.0312>>.

LIN, Y. et al. **Yale/UNC-CH - Geophysical Waveform Inversion**. 2025. <https://kaggle.com/competitions/waveform-inversion>. Kaggle.

LU, L. et al. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. **Nature machine intelligence**, Nature Publishing Group UK London, v. 3, n. 3, p. 218–229, 2021.

LUO, C. **Understanding Diffusion Models: A Unified Perspective**. [S.l.]: arXiv, Aug. 2022.

MARTIN, G. S.; WILEY, R.; MARFURT, K. J. Marmousi2: An elastic upgrade for Marmousi. **The Leading Edge**, v. 25, n. 2, p. 156–166, Feb. 2006. ISSN 1070-485X, 1938-3789.

METZ, L. et al. Unrolled generative adversarial networks. **arXiv preprint arXiv:1611.02163**, 2016.

OVCHARENKO, O. et al. Style transfer for generation of realistically textured subsurface models. In: **SEG Technical Program Expanded Abstracts 2019**. San Antonio, Texas: Society of Exploration Geophysicists, 2019. p. 2393–2397.

PARASYRIS, A.; STANKOVIC, L.; STANKOVIC, V. Synthetic Data Generation for Deep Learning-Based Inversion for Velocity Model Building. **Remote Sensing**, v. 15, n. 11, p. 2901, Jun. 2023. ISSN 2072-4292.

PUZYREV, V. et al. Geophysical model generation with generative adversarial networks. **Geoscience Letters**, v. 9, n. 1, p. 32, Aug. 2022. ISSN 2196-4092.

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. **arXiv preprint arXiv:1511.06434**, 2015.

REN, Y. et al. Building Complex Seismic Velocity Models for Deep Learning Inversion. **IEEE Access**, v. 9, p. 63767–63778, 2021. ISSN 2169-3536.

ROMBACH, R. et al. High-resolution image synthesis with latent diffusion models. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2022. p. 10684–10695.

RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. 2015. Available at: <<https://arxiv.org/abs/1505.04597>>.

ROY, A. G.; NAVAB, N.; WACHINGER, C. Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. **IEEE transactions on medical imaging**, IEEE, v. 38, n. 2, p. 540–549, 2018.

SAATCHI, Y.; WILSON, A. G. **Bayesian GAN**. 2017. Available at: <<https://arxiv.org/abs/1705.09558>>.

SALLES, T. et al. A unified framework for modelling sediment fate from source to sink and its interactions with reef systems over geological times. **Scientific Reports**, v. 8, n. 1, p. 5252, Mar. 2018. ISSN 2045-2322.

Sohl-Dickstein, J. et al. **Deep Unsupervised Learning Using Nonequilibrium Thermodynamics**. [S.l.]: arXiv, Nov. 2015.

SUN, C. et al. Revisiting unreasonable effectiveness of data in deep learning era. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2017. p. 843–852.

TARANTOLA, A. Inversion of seismic reflection data in the acoustic approximation. **GEOPHYSICS**, v. 49, n. 8, p. 1259–1266, Aug. 1984. ISSN 0016-8033, 1942-2156.

TRINH, L. T.; HAMAGAMI, T. Latent denoising diffusion gan: Faster sampling, higher image quality. **IEEE Access**, v. 12, p. 78161–78172, 2024.

VAHDAT, A.; KAUTZ, J. NVAE: A deep hierarchical variational autoencoder. In: **Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2020.

VERSTEEG, R. The Marmousi experience: Velocity model determination on a synthetic complex data set. **The Leading Edge**, v. 13, n. 9, p. 927–936, Sep. 1994. ISSN 1070-485X, 1938-3789.

VIRIEUX, J.; OPERTO, S. An overview of full-waveform inversion in exploration geophysics. **GEOPHYSICS**, v. 74, n. 6, p. WCC1–WCC26, Nov. 2009. ISSN 0016-8033, 1942-2156.

WANG, F.; HUANG, X.; ALKHALIFAH, T. **Controllable Seismic Velocity Synthesis Using Generative Diffusion Models**. [S.l.]: arXiv, Feb. 2024.

WANG, W.; MA, J. Velocity model building in a crosswell acquisition geometry with image-trained artificial neural networks. **GEOPHYSICS**, v. 85, n. 2, p. U31–U46, Mar. 2020. ISSN 0016-8033, 1942-2156.

WEN, G. et al. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. **Advances in Water Resources**, v. 163, p. 104180, 2022. ISSN 0309-1708. Available at: <<https://www.sciencedirect.com/science/article/pii/S0309170822000562>>.

WIGHTMAN, R.; TOUVRON, H.; JÉGOU, H. Resnet strikes back: An improved training procedure in timm. **arXiv preprint arXiv:2110.00476**, 2021.

WILLSE, T. **Expanding a linearly independent set to a basis**. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/1701901 (version: 2016-03-17). Available at: <<https://math.stackexchange.com/q/1701901>>.

WU, Y.; LIN, Y. **InversionNet: An Efficient and Accurate Data-Driven Full Waveform Inversion**. 2018. 419-433 p.

YANG, F.; MA, J. Deep-learning inversion: A next-generation seismic velocity model building method. **Geophysics**, Society of Exploration Geophysicists, v. 84, n. 4, p. R583–R599, 2019.

YANG, Z. et al. Xlnet: Generalized autoregressive pretraining for language understanding. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2019. v. 32. Available at: <<https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf>>.

YU, W. et al. Metaformer baselines for vision. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 46, n. 2, p. 896–912, 2023.

YUN, S. et al. Cutmix: Regularization strategy to train strong classifiers with localizable features. In: **Proceedings of the IEEE/CVF international conference on computer vision**. [S.l.: s.n.], 2019. p. 6023–6032.

ZHANG, H. et al. mixup: Beyond empirical risk minimization. **arXiv preprint arXiv:1710.09412**, 2017.

ZHANG, Z.; LIN, Y. Data-driven seismic waveform inversion: A study on the robustness and generalization. **IEEE Transactions on Geoscience and Remote Sensing**, v. 58, n. 10, p. 6900–6913, 2020.

ZHU, M. et al. Fourier-DeepONet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness. **Computer Methods in Applied Mechanics and Engineering**, v. 416, p. 116300, 2023.

ZONG, Z.; SONG, G.; LIU, Y. Detrs with collaborative hybrid assignments training. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2023. p. 6748–6758.