

FULL-WAVEFORM MODELING AND MACHINE LEARNING-BASED TOMOGRAPHY OF VOLCANIC EDIFICES

A THESIS

submitted to

*Indian Institute of Science Education and Research Pune
in partial fulfilment of the requirements for the
BS-MS Dual Degree Programme*

BY

Rifa Medappil Pinatt



April, 2025

Supervisor:

Prof. Dr. Georg Rümpker
Institute of Geosciences
Goethe University Frankfurt
60438 Germany

© Rifa Medappil Pinatt
All rights reserved

CERTIFICATE

This is to certify that this dissertation entitled **Full-Waveform Modeling and ML-based Tomography of Volcanic Edifices** towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents work carried out by **Rifa Medappil Pinatt** at Goethe University Frankfurt under the supervision of **Dr. Georg Rumpker**, Professor, Institute of Geosciences, Goethe University Frankfurt, during the academic year 2024-2025.



Expert:
Dr. Arjun Datta
IISER Pune

Supervisor:
Prof. Dr. Georg Rumpker
Goethe University Frankfurt



This thesis is dedicated to all the moments I have cherished while traveling in Europe from June 2024 to March 2025, which will forever stay with me.
Diese Momente bedeuten mir sehr viel. Ich werde diese Zeit nie vergessen.

DECLARATION

I hereby declare that the matter embodied in the report entitled **Full-Waveform Modeling and ML-based Tomography of Volcanic Edifices** are the results of the work carried out by me at the Institute of Geosciences, Goethe University Frankfurt, under the supervision of **Dr. Georg Rümpker**, and the same has not been submitted elsewhere for any other degree. Wherever others contribute, every effort is made to indicate this clearly, with due reference to the literature and acknowledgement of collaborative research and discussions.

A handwritten signature in black ink, appearing to read 'Rifa Medappil Pinatt', with a stylized, cursive script.

Rifa Medappil Pinatt
20201229

ACKNOWLEDGEMENT

I sincerely thank Prof. Dr. Georg Rümpler for giving me the opportunity to undertake this project and for his invaluable support and guidance. I am also grateful to Dr. Arjun Datta for his long-term mentorship and for kindly agreeing to be my expert.

Additionally, I would like to express my sincere gratitude to my co-supervisors, Abolfazl Komeazi and Fabian Limberger, for their unwavering support and mentorship throughout the course of my thesis. Whether it concerned work or personal matters, they were always there for me, generously sharing their knowledge, advice, and encouragement throughout this journey. I also want to thank Sharare Zhian for her company and friendship.

I'm grateful to have been accepted into the Goethe Research Experience Program (GREP), which made my tenure smoother and more enjoyable. They organized social outings and gatherings that were enjoyable to be a part of, and the German classes were a lot of fun too. I thank the entire GREP team for making this possible, and to the friends I made through the program.

Last but not least, I would like to thank my dear (and sometimes delightfully dumb) friends — Ananthu, Anju, Joseph, Sakshi, Manav Sivaram, and Arya — for filling my college life with unforgettable memories, and good times. I don't think Devaprasad deserves my gratitude, so I'll skip that for now.

Danke schön.

Contents

Abstract	I
Acronyms	II
1 Introduction	1
1.1 Background	1
1.2 Problem Statement	1
1.3 Research Objectives	2
1.4 Scope and Limitations	2
1.5 Methodology Overview	2
1.6 Thesis Structure	3
2 Ray-based Travel-Time Tomography	4
2.1 Introduction	4
2.2 Methodology	4
2.2.1 Assumptions	4
2.2.2 Forward Modeling	5
2.2.3 Determination of Systems	5
2.2.4 Linear Inversion	5
2.2.5 ANN Inversion	6
2.3 Model Setup	8
2.3.1 Modeling Variables	8
2.3.2 2-D Domain	9
2.3.3 3-D Domain	10
2.4 Results	11
2.4.1 2-D Domain	12
2.4.2 3-D Domain	17
2.5 Discussion	18
3 Full Waveform Inversion	20
3.1 Introduction	20
3.2 Methodology and Setup	20
3.2.1 Forward Modeling	20
3.2.2 Waveform Reconstruction	24
3.2.3 Inversion	26
3.3 Results	28
3.3.1 Setup 1	29

3.3.2	Setup 2	30
3.3.3	Setup 3	32
3.4	Discussion	33
4	Discussion	35
5	Conclusion	37
	References	III
	Appendices	A
A	Computational Environments	A
A.1	Programming Languages	A
A.2	Major Libraries and Frameworks	A
A.3	Computational Resource	A
B	Forward Operator: G	B
C	Spectral Element Method	C

List of Figures

2.1	Examples for Input-Output Pairs	6
2.2	2D ANN	7
2.3	3D ANN	8
2.4	Examples for Model Meshing	9
2.5	Examples for 2D Model Setups. The lines represent ray coverage.	10
2.6	3D True Model	11
2.7	3D Model Setups ¹	11
2.8	LI: Model 1	12
2.9	LI: Model 2	12
2.10	LI: Model 3	13
2.11	LI: Model 4	13
2.12	LI: Model 5	14
2.13	LI: Model 6	14
2.14	LI vs ANN: Comparison Statistics	15
2.15	LI vs ANN: Model 6	15
2.16	LI vs ANN: Model 7	16
2.17	LI vs ANN: Model 8	16
2.18	LI vs ANN: Model 9	17

2.19	LI vs ANN: 3D Setup 1 ²	17
2.20	LI vs ANN: 3D Setup 2	18
3.1	Model Setups for Full-Waveform Modeling	21
3.2	Velocity densification example for 3×3 mesh	21
3.3	Random velocity profile with the Velocity-Y seismogram generated using Salvus.	22
3.4	Meshing and absorbing boundary condition (ABC) elements for the veloc- ity profile shown in Figure 3.3.	23
3.5	Some wavefield snapshots illustrating Velocity-Y propagation for the ve- locity profile shown in Figure 3.3.	23
3.6	Comparison of an example true waveform with its 10% downsampled ver- sion	24
3.7	Autoencoder architecture for waveform reconstruction in Setup 1 ³ . The dimensions shown represent the sizes of the convolutional layers, which are reduced by a factor of 2 after each max-pooling layer. The resulting dimensions after max-pooling are indicated below.	25
3.8	Color maps for the artificial neural network (ANN) plots	25
3.9	Autoencoder feature maps for an example case using Setup 1	26
3.10	ANN Architecture for Inversion. The dimensions shown represent the sizes of the convolutional layers, which are reduced by a factor of 2 after each max-pooling layer.	27
3.11	Encoded Waveform for Inversion. Compare with the feature maps in Fig- ure 3.9 used for waveform reconstruction.	27
3.12	ANN Architecture for Setups 1 and 2	28
3.13	ANN Architecture for Setup 3	28
3.14	Setup 1: Statistics	29
3.15	Setup 1 with a random test velocity profile: Model 1	29
3.16	Setup 1 with a random test velocity profile: Model 2	30
3.17	Setup 1 with a velocity profile containing an anomaly: Model 3	30
3.18	Setup 2: Statistics	31
3.19	Setup 2 with a random test velocity profile: Model 1	31
3.20	Setup 2 with a velocity profile containing an anomaly: Model 2	32
3.21	Setup 3: Statistics	32
3.22	Setup 3 with a velocity profile containing an anomaly: Model 1	33
B.1		B
B.2		B

Abstract

Seismic imaging is an essential method for exploring the structure of volcanic features beneath the surface, such as conduits and magma chambers. These insights are crucial for a more profound understanding of volcanic processes. However, the complex and heterogeneous nature of the subsurface, along with the limited availability of data, presents substantial challenges in accurately mapping the internal structure of volcanoes, often leading to interpretive uncertainties.

This study explores the application of machine learning, specifically artificial neural networks (ANNs), in seismic tomography and full-waveform inversion to improve volcanic imaging. The training is performed on synthetic data generated from randomized velocity structures.

In the initial phase, a feedforward neural network (FNN) is trained on travel-time data to predict velocity profiles. Tests in the 2D domain for simplified cases demonstrate that the ANN approach outperforms traditional linear inversion techniques, especially when seismic data is sparse. Furthermore, the method's efficiency in the 3D domain is demonstrated.

However, in real-world scenarios, such inversions would significantly benefit from using full-waveforms to accurately determine the velocity profiles. An ANN with an encoder followed by a fully connected stack is utilized. The encoder, composed of convolutional layers, extracts features from the waveforms, which are then used by the fully connected stack to predict velocity profiles. Initial tests show promising results, suggesting that encoders can capture key characteristics of the waveforms — beyond just arrival times — which can be leveraged for inversion. The ANN architecture can be further optimized to enhance the inversion process and better account for complex velocity profiles.

Acronyms

ABC	absorbing boundary condition 22, 23
ANN	artificial neural network 1, 2, 6, 7, 14–16, 18, 19, 24, 25, 29–34, 36, 37
FC	fully connected 26, 27
FNN	feedforward neural network 2, 4
FWI	full-waveform inversion 1, 2, 33–35, 37
L-BFGS	limited-memory Broyden–Fletcher–Goldfarb–Shanno 8
LI	linear inversion 11
ML	machine learning 1–4, 11, 34–37
MSE	mean squared error 8, 25, 26, 33
SEM	spectral element method 2, 20, 21, 34, C
SSIM	structural similarity index 14–16, 18, 19, 28, 29, 31–33, 36, 37

Chapter 1

Introduction

1.1 Background

Seismology, a sub-discipline of geophysics, is the study of seismic waves and their propagation through the Earth. It is mainly an observational science, which is fundamental in understanding subsurface structures and earthquake dynamics. In the context of a volcanic edifice, subsurface structures include but are not limited to, conduits, magma chambers, and dykes. Identifying these structures is essential for monitoring magma movement [1, 2], assessing eruption risks, and evaluating other seismic hazards. They can be identified by seismic imaging as they influence the velocity profiles [2].

Ray-based travel-time tomography [3] and full-waveform inversion (FWI) [4] are seismic imaging techniques used to reconstruct subsurface velocity profiles from ground motion data. Ray-based travel-time tomography utilizes straight ray-based travel-time data while full-waveform inversion utilizes entire waveforms of the seismic signals. Full-waveform inversion has an advantage over travel-time tomography as it incorporates amplitude and phase information in addition to arrival times. It also accounts for wave propagation effects such as reflection, refraction, diffraction etc.

1.2 Problem Statement

Obtaining data from volcanic edifices is challenging due to hazardous terrain and the limited spatial coverage of data collection. This limited availability of data, combined with the complex and heterogeneous nature of the subsurface, presents substantial challenges in accurately mapping the internal velocity profiles of volcanic edifices, often leading to ambiguity in interpretations [1, 5]. These challenges highlight the need for methods that can extract meaningful insights despite limited data coverage.

In recent years, machine learning (ML) has emerged as a powerful tool in many scientific and engineering disciplines, offering innovative solutions for complex problems. In the field of seismology, ML has been increasingly applied to seismic data analysis, wave propagation modeling [6, 7], and other related tasks [8].

Recent advancements have demonstrated the potential of ANNs in ray-based travel-time tomography. Komeazi et al. (2023) [9] introduced an ANN-based optimization ap-

proach for 2D sparse-ray travel-time tomography of volcanic edifices, showing improved performance over traditional linear inversion techniques. This study aims to extend their approach by evaluating its efficiency in 3D. Additionally, the potential of ANNs in full-waveform inversion is explored to further improve velocity profile predictions.

1.3 Research Objectives

This study aims to:

1. Implement an ANN-based ray-based travel-time tomography method for imaging volcanic interiors
2. Evaluate its performance against traditional linear inversion techniques
3. Generate synthetic seismograms using full-waveform modeling
4. Develop an ANN-based full-waveform inversion method
5. Predict velocity profiles from previously unseen seismograms using the ANN

1.4 Scope and Limitations

This study explores the application of ANNs for seismic inversion in data-sparse conditions. It is intended as an initial investigation to assess the potential of ML for seismic inversion rather than a fully optimized framework. This is particularly relevant for full-waveform inversion, where only very simplified cases are considered to test the viability of the approach. Additionally, the study uses only synthetic seismic data without noise, limiting its direct applicability to real-world datasets. Source locations are assumed to be known as well.

1.5 Methodology Overview

In this study the following methodologies are used

ANN-based Ray-based Travel-Time Tomography: A fully connected feedforward neural network (FNN) with two hidden layers is implemented in the 2D domain. In the 3D domain, an additional hidden layer is introduced to account for the complexity due to the third dimension.

Synthetic Seismogram Data Generation: Salvus [10], a Python-based framework that implements the spectral element method (SEM), is used for waveform forward modeling.

ANN-based FWI: An ANN with an encoder composed of convolutional layers for feature extraction, followed by a fully connected stack is utilized to predict velocity profiles from seismograms.

1.6 Thesis Structure

This thesis is organized into four chapters. Chapter 1 introduces the research topic, outlining the challenges and presenting the study objectives. Chapter 2 focuses on ray-based travel-time tomography, explaining the model setups and methods, and comparing a conventional inversion technique against an ML approach. Chapter 3 delves into full-waveform inversion, discussing its advantages, methods along with the model setups, and how it enhances subsurface imaging by incorporating additional information. Finally, Chapter 4 summarizes and discusses the results and suggests potential directions for future research.

Chapter 2

Ray-based Travel-Time Tomography

2.1 Introduction

Ray-based travel-time tomography is an inversion technique used to estimate subsurface velocity profile or other physical parameters from seismic travel-times. This study starts in the 2D domain, by implementing the tomography approaches taken from Komeazi et al. [9]. While they used a 2D Gaussian model, representing a vertical cross-section of a volcanic edifice, this study instead employs a circular model, representing a horizontal cross-section of the volcanic edifice. The study is then extended to 3D domain employing a 3D gaussian surface.

We begin the chapter by outlining the methodologies (Section 2.2) used for modeling. Inversion is performed using two different approaches: conventional linear inversion and a machine learning (ML) based inversion. The latter employs a fully connected feedforward neural network (FNN). We then define the model setups in Section 2.3. Finally, Section 2.4 starts with the linear inversion results for various 2D setups to highlight the motivation for using the ML-based approach. We then compare the results of both the techniques, including their performance in 3D.

2.2 Methodology

Every inverse problem begins with the formulation of the forward problem, which describes how the observable data (here, travel-times) can be computed from a given model [11]. It establishes a mathematical relation between the model parameters and the observable data, which is essential in defining the inverse relation for a conventional approach and in generating the datasets for an ML-based approach.

We begin this section by addressing the modeling assumptions, followed by defining the forward problem and the inversion techniques.

2.2.1 Assumptions

Several assumptions are considered to simplify the modeling:

1. Straight-rays assumption: reflection, refraction, and diffraction phenomena are not taken into consideration
2. The location of the sources is assumed to be known a prior
3. Noise effects are not considered

2.2.2 Forward Modeling

Forward problem [11] is given by

$$\mathbf{d} = \mathbf{G}\mathbf{m} \quad (2.1)$$

where $\mathbf{d}$ is the travel-times vector, $\mathbf{G}$ is the forward operator and $\mathbf{m}$ is the slowness (inverse of velocity) vector. $\mathbf{G}$ maps the observable data $\mathbf{d}$ to the model parameters $\mathbf{m}$. In the matrix form, equation 2.1 can be expressed as

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_r \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \dots & G_{1p} \\ G_{21} & G_{22} & \dots & G_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ G_{r1} & G_{r2} & \dots & G_{rp} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_p \end{bmatrix} \quad (2.2)$$

where r denotes the total number of rays and p denotes the total number of model parameters. Here, $\mathbf{G}$ is defined as the matrix of ray segments. Each $(i, j)^{th}$ element of $\mathbf{G}$ corresponds to the length of the i^{th} ray within the area of j^{th} model parameter. However, model parameters with no ray coverage are deleted for computations as they are redundant. Thus, the shape of $\mathbf{G}_{\text{eff}}$ that is used in computations is (r, p_{eff}) , *i.e.* it is determined by the number of rays and number of model parameters with ray coverage. For a detailed explanation on the forward operator computation, refer Appendix B.

2.2.3 Determination of Systems

A system is either underdetermined or overdetermined based on the number of equations and unknowns. In our case, for inversion, the number of equations is controlled by the number of rays r and number of unknowns by the number of model parameters p_{eff} .

- Well-determined : $r = p_{\text{eff}}$
- Overdetermined : $r > p_{\text{eff}}$
- Underdetermined : $r < p_{\text{eff}}$

2.2.4 Linear Inversion

The inverse problem for Equation 2.1 is given by

$$\mathbf{m} = \mathbf{G}_{\text{eff}}^{-1} \mathbf{d} \quad (2.3)$$

However, $\mathbf{G}_{\text{eff}}^{-1}$ exists only if $\mathbf{G}_{\text{eff}}$ is a square matrix and of full rank (no linearly dependent rows or columns and non-singular). This often is not the case in practical scenarios. Hence, if the system is overdetermined, Moore-Penrose pseudoinverse is used to obtain $\mathbf{m}$

$$\mathbf{m} = (\mathbf{G}_{\text{eff}}^T \mathbf{G}_{\text{eff}})^{-1} \mathbf{G}_{\text{eff}}^T \mathbf{d} \quad (2.4)$$

If the system is underdetermined, regularization techniques are employed to stabilize the solution and prevent overfitting the data. Regularization is also applied when $\mathbf{G}_{\text{eff}}$ is ill-conditioned or singular. In such cases, Tikhonov regularization [12] is commonly used.

$$\mathbf{m} = (\mathbf{G}_{\text{eff}}^T \mathbf{G}_{\text{eff}} + \lambda^2 \mathbf{I})^{-1} \mathbf{G}_{\text{eff}}^T \mathbf{d} \quad (2.5)$$

where λ is the regularization factor. It controls the trade-off between fitting the data and smoothing the solution. $\mathbf{I}$ is the identity matrix.

2.2.5 ANN Inversion

Machine learning draws inferences from patterns in data and can be trained on large datasets to make predictions on unseen data. Artificial neural network (ANN) [13] is a type of machine learning technique that tries to mimic the human brain. It consists of connected nodes called artificial neurons as we have neurons in the brain, and these neurons work together to identify patterns, evaluate choices and draw conclusions [14]. It does so by training on large datasets for repeated iterations (epochs) aimed at adjusting the weights and biases associated with the connections between neurons in the consecutive layers of an ANN.

Applying to our specific case, the ANN model would serve as the inverse operator, predicting the velocity profile when provided with travel-times. The ANN becomes capable of doing so by learning the relation between travel-times and velocity profiles from the large training datasets provided. This approach is implemented using PyTorch version 2.5.1 [15] on a single NVIDIA GeForce RTX 2080 Ti GPU.

In order to construct an ANN model, we need to broadly consider three key components:

1. Training Dataset
2. ANN Architecture
3. Loss Function and Optimization

Training Dataset

The training dataset consists of input-output pairs that the ANN learns from. It is generated by forward modeling using Equation 2.1. Our dataset consists of 15,000 random velocity profiles (in range (1,2) km/s) and their corresponding travel-times. It is split into 80% for training and 20% for testing. Training is done on a batch size of 2048.

- Inputs - Travel-Time Arrays
- Outputs - Velocity Profiles

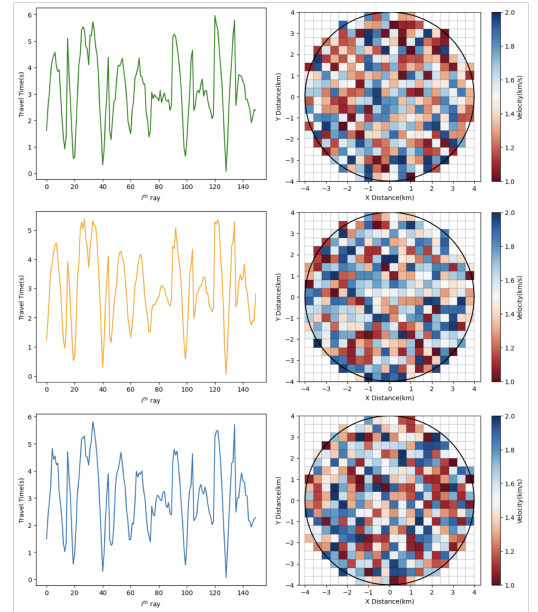


Figure 2.1: Examples for Input-Output Pairs

It is important to note that in forward modeling, the velocity profiles serve as the inputs, while the travel-times are the resulting outputs.

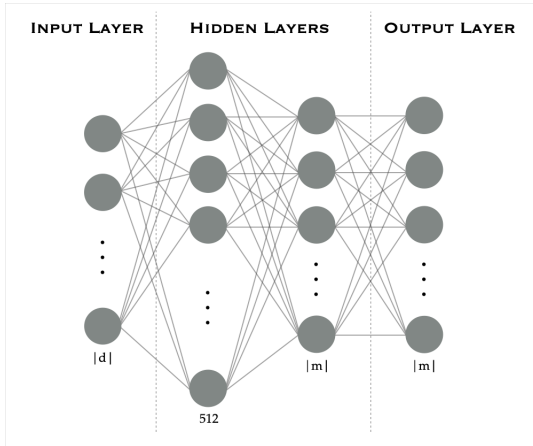
ANN Architecture

The neural network architecture consists of fully connected layers with a Tanh activation function. In the 2D domain, two hidden layers are used, whereas in the 3D domain, the architecture is slightly modified by adding an additional hidden layer. The extra hidden layer in 3D helps the network account for the complexities introduced by the third dimension.

The network begins with an input layer, where the travel-time arrays are fed into the ANN model. Each input node corresponds to the travel-time of a ray. The input layer is then followed by two or three hidden layers, depending on the dimensionality.

A hidden layer size of 512 neurons is chosen for the 2D case, providing sufficient capacity while avoiding excessive complexity, which helps to prevent overfitting. For the 3D case, an additional hidden layer of 1024 neurons is included, which enables the network to better capture the more complex features of the data. The final hidden layer is set to match the output size, helping to refine the final values and improve the accuracy of the model's predictions.

Each neuron receives inputs from all the neurons in the previous layer, computes a weighted sum, and passes it to the next layer. A Tanh activation function is applied after the first hidden layer to introduce non-linearity into the model. Even though the problem at hand has a linear relationship, non-linearity can help capture important features of the data to avoid underfitting. Finally, the output layer computes the predicted velocity profile.



ANN Architecture for 2D

1. Input Layer with $|d|$ neurons
2. Hidden Layer 1 with 512 neurons
3. Hidden Layer 2 with $|m|$ neurons
4. Output Layer with $|m|$ neurons

Figure 2.2: 2D ANN

As described above, during forward propagation, travel-time data flows from the input layer through the hidden layers to the output layer. Each layer transforms the data in some way to predict the velocity profile. The network is trained using backpropagation [16], a process where the error between the predicted velocity profile and the true velocity profile is calculated and propagated backward through the network. The weights are adjusted during this process to minimize the error using an optimization algorithm.

ANN Architecture for 3D

1. Input Layer with $|\mathbf{d}|$ neurons
2. Hidden Layer 1 with 1024 neurons
3. Hidden Layer 2 with 512 neurons
4. Hidden Layer 3 with $|\mathbf{m}|$ neurons
5. Output Layer with $|\mathbf{m}|$ neurons

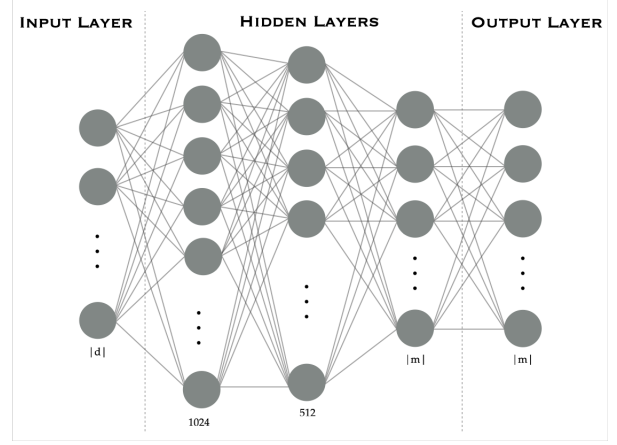


Figure 2.3: 3D ANN

Loss function and Optimization

The weights are updated in each epoch aimed at minimizing the loss function using an optimizer. While the optimizer adjusts the weights, the loss function guides the model. Mean squared error (MSE) [17] is employed as the loss function. The model tries to learn by reducing the average squared difference between the predicted and actual values.

$$MSE = L(\mathbf{m}) = \frac{1}{N} \sum_{i=1}^N (m_i - \hat{m}_i)^2 \quad (2.6)$$

where N is the number of parameters, m_i is the true value (actual value) and $\hat{m}_i$ is the predicted value.

Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [18] is used as the optimizer. L-BFGS is a quasi-Newton method that approximates the Hessian matrix to optimize the method efficiently.

$$m^{n+1} = m^n - \eta \mathbf{H}^{-1} \nabla(L(\mathbf{m})) \quad (2.7)$$

where η is the learning rate, $\mathbf{H}$ is the Hessian matrix (second derivatives of the loss function relative to the weights) and $\nabla(L(\mathbf{m}))$ is the gradient of the loss function.

2.3 Model Setup

2.3.1 Modeling Variables

The key modeling variables that influence the complexity of the model include meshing and source-receiver setup, which are detailed in the following sections.

- Meshing : controls the number of model parameters p
- Source-Receiver Setup : controls the number of rays r and ray coverage

2.3.2 2-D Domain

We consider a circular model of radius 4 km. The circular area corresponds to a horizontal cross-section of a volcanic edifice. A 4 km radius can help in capturing seismic signals relevant to volcanic activity such as magma movement, earthquakes, and volcanic tremor, while still being manageable for computational simulations. For instance, at volcanoes like Mount St. Helens [19], the summit crater spans about 1-1.5 km in radius, and the mountain base spans about 5-6 km in radius. A 4 km radius would be a good choice for monitoring at an optimal elevation. However, radius can be changed according to the elevation of choice.

For our modeling, we consider various source-receiver setups. We use a homogeneous background velocity structure with one or two low velocity anomalies indicative of magma chambers or other subsurface structure, for the comparison of different approaches.

Meshing

Meshing influences the number of actual model parameters, $\mathbf{p}$, and determines the spatial resolution of the velocity profile. A finer mesh has larger number of parameters, allowing higher-resolution velocity profile. A coarser mesh has a smaller count but is limited to coarser velocity profiles. Some examples are provided below in Figure 2.4. Dxy represents the x and y length of each smaller block.

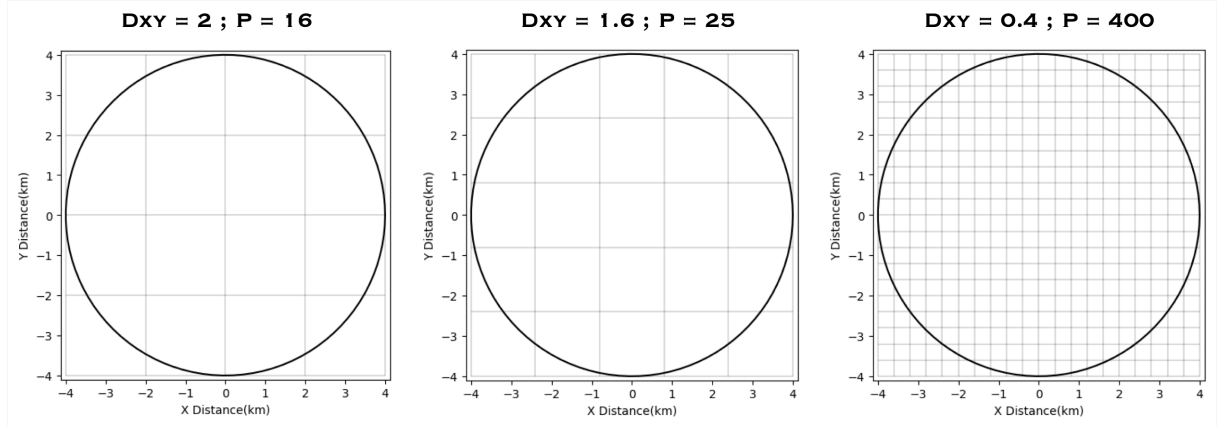


Figure 2.4: Examples for Model Meshing

Source-Receiver Setups with Ray Coverage

Sources are either placed randomly within the circle or by the boundary. Sources if placed inside the circle represent natural seismic events, such as earthquakes or volcanic tremors. In contrast, sources if placed at the boundary of the circle represent controlled seismic sources, which could be designed to simulate for experimental purposes. The receivers are restricted to the boundary. If the number of sources is $\mathbf{N}_s$ and the number of receivers is $\mathbf{N}_r$, then the total number of rays, $\mathbf{r} = \mathbf{N}_s \times \mathbf{N}_r$. The relative spatial distribution of sources and receivers controls the ray coverage. For example, in Figure 2.5, despite having the same number of sources and receivers, Setup 3 has a better ray coverage than Setup 2 due to a more optimal spatial distribution. The lines in the figures show the ray coverage.

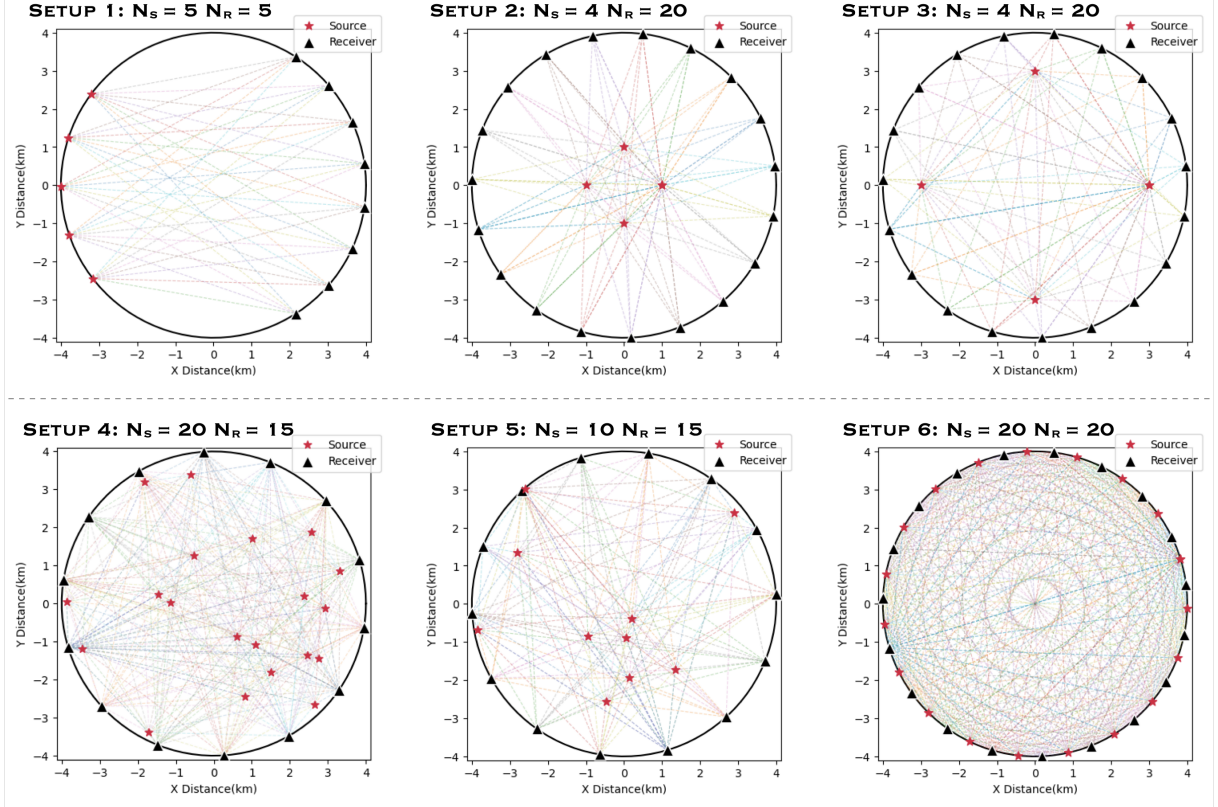


Figure 2.5: Examples for 2D Model Setups. The lines represent ray coverage.

2.3.3 3-D Domain

We consider a 3D Gaussian surface for simple modeling. While no real volcano perfectly follows a 3D Gaussian shape, some volcanic edifices approximate a Gaussian-like distribution in terms of their elevation profiles. For example, Mount Fuji in Japan is a nearly symmetric stratovolcano with a conical shape, often approximated by Gaussian-like profiles in topographic and geophysical modeling. Its smooth slopes and radial symmetry make it a suitable real-world analogy for simplified Gaussian surface representations in volcanic studies.

Our model extends 3 km laterally and ~ 1.6 km in height. Sources are placed randomly within the gaussian volume while receivers are restricted to the surface. For meshing, we consider a cubic volume containing the gaussian volume. The cubic volume is divided into smaller cubes, where each smaller cube corresponds to a model parameter.

We assume a true velocity profile with a background velocity of 1.5 km/s and a vertical low-velocity anomaly of 1.2 km/s. We use $15 \times 15 \times 15$ meshing with 3375 actual model parameters. The fully defined model is shown in Figure 2.6, illustrating the true velocity profile along with the outline of the Gaussian surface.

We model for two different source receiver settings as shown in Figure 2.7. Sources are representative of natural seismic events, such as earthquakes or volcanic tremors. In both the cases, we have 10 receivers, with 5 receivers uniformly distributed at heights 0.557 km (a radius of 1.2km in x-y plane) and 1.13 kms (a radius of 0.9km in x-y plane) each. The two setups differ in the number of sources as given below. The 3D plotting is performed in Julia [20] using Makie [21], as it provides interactivity.

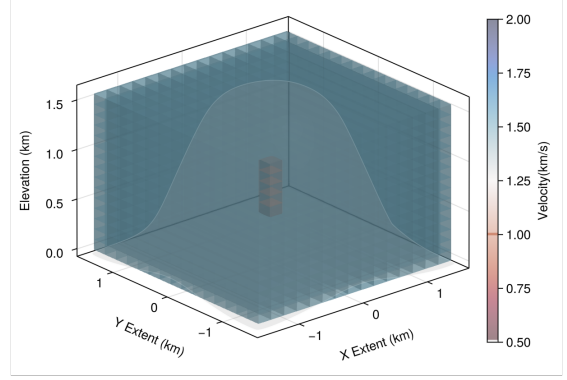


Figure 2.6: 3D True Model

- | | |
|-------------------------------------|-------------------------------------|
| • Number of Sources = 100 | • Number of Sources = 50 |
| • Number of Receivers = 10 | • Number of Receivers = 10 |
| • Number of Rays = 1000 | • Number of Rays = 500 |
| • Effective Model Parameters = 1154 | • Effective Model Parameters = 1089 |

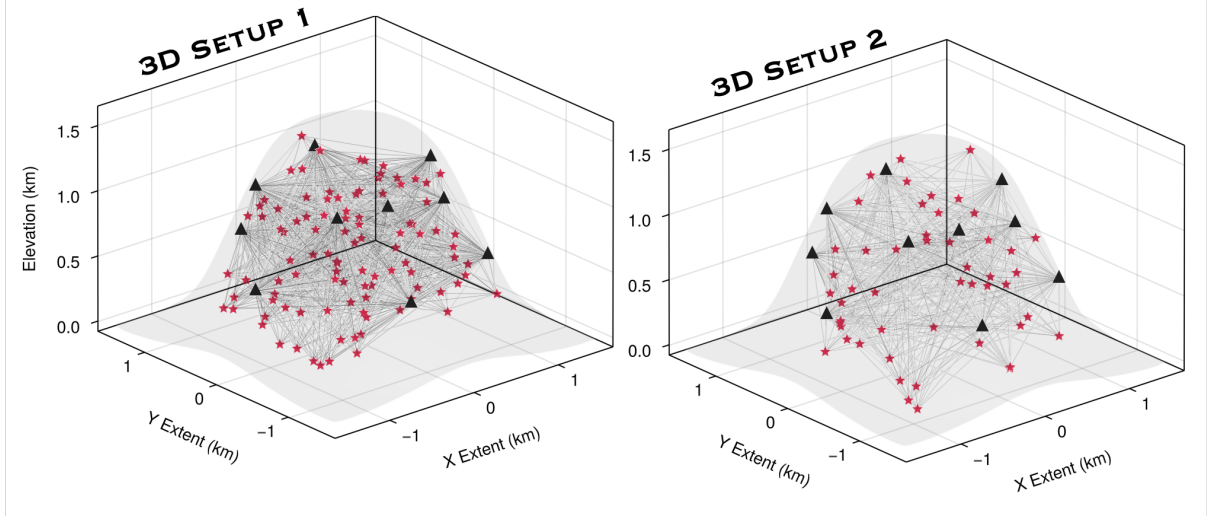


Figure 2.7: 3D Model Setups ¹

2.4 Results

We begin this section by presenting linear inversion (LI) results for various 2D setups. The setups are carefully chosen to highlight the motivation for using the ML approach. We then compare the results of both inversion methods starting from the setup where linear inversion appears to fail, *i.e.* in sparse-ray tomography (underdetermined systems). Results for 3D domain are then presented.

¹Markers: Stars-Sources, Triangles-Receivers

2.4.1 2-D Domain

Linear Inversion

For all the setups considered, $\mathbf{G}$ is ill-conditioned with condition number $> 10^{17}$. Even $\mathbf{G}_{\text{eff}}$ is ill-conditioned with condition number $> 10^{16}$. Hence, regularization is applied to all the models, *i.e.* Equation 2.5 is employed. The value of λ for modeling is chosen by trial and error method. Grey cells in the figures correspond to the parameters with no ray coverage. Star markers represent sources while the triangle markers represent the receivers. The grey lines show ray coverage in all the true models.

Figures 2.8-2.10 illustrate well-resolved cases for overdetermined systems. Model 1 (Figure 2.8) contains 16 parameters, each cell extending 2 kms in both the x and y directions. The source-receiver configuration is designed to cover all the parameters using 25 rays, ensuring full coverage of the model.

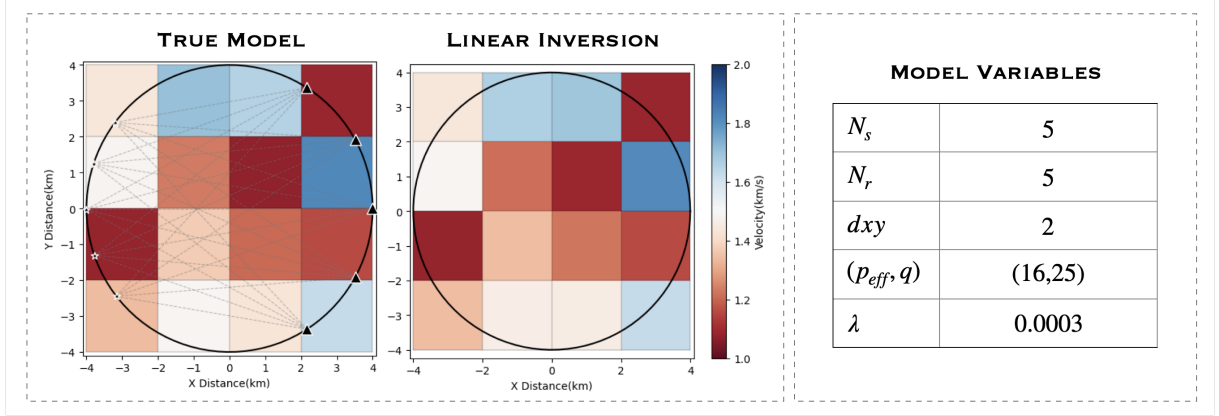


Figure 2.8: LI: Model 1

In Model 2 (Figure 2.9), the source-receiver setup remains unchanged, but the size of each parameter is reduced to 1.6 km in both the x and y directions. As a result, the total number of parameters increases to 25. However, only 23 parameters are covered, maintaining the overdetermined nature of the model.

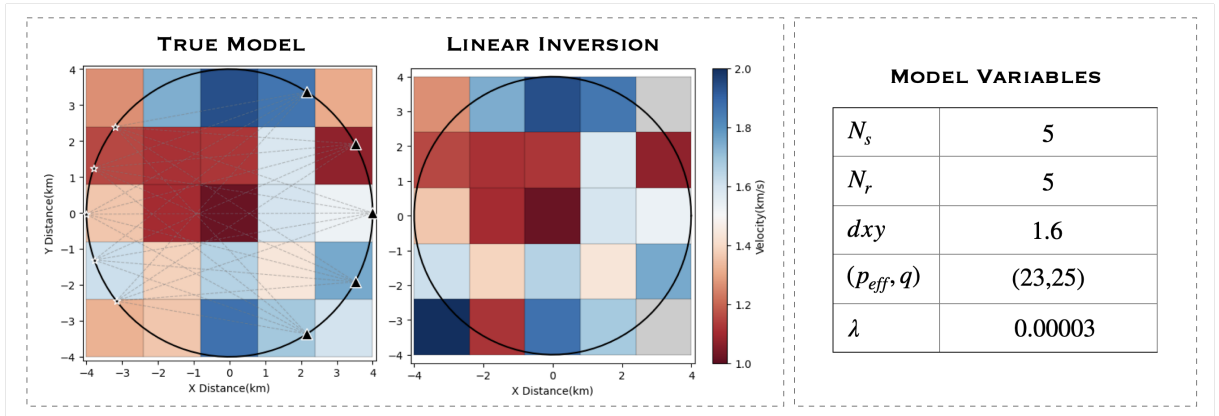


Figure 2.9: LI: Model 2

In Model 3 (Figure 2.10), the size of each parameter is further reduced to 0.4 km in both the x and y directions. The source-receiver setup is adjusted to provide denser ray coverage, as the number of model parameters increases to 400. Within the circular boundary, there are 344 parameters, all of which are fully covered by 400 rays, ensuring the model remains overdetermined.

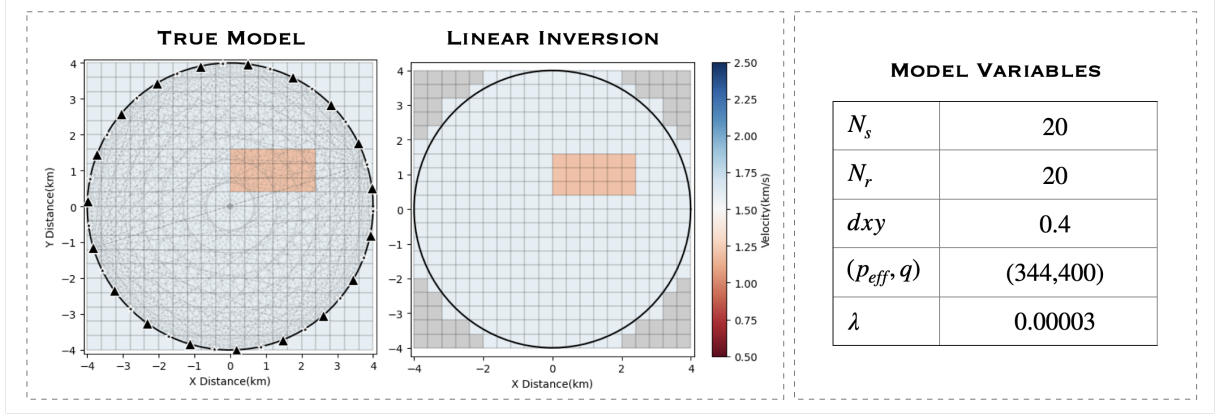


Figure 2.10: LI: Model 3

From Model 3 onwards, we focus on velocity profiles featuring a low-velocity anomaly, which is representative of low-velocity magma chambers within volcanic edifices. We consider a background velocity of 1.6km/s with an anomaly of 1.2km/s.

Figures 2.11-2.13 illustrate results for underdetermined systems. In Model 4 (Figure 2.11), compared to Model 3, the number of receivers is reduced by 5, resulting in a total of 300 rays. This reduction in ray count (q) noticeably affects the accuracy of the inversion.

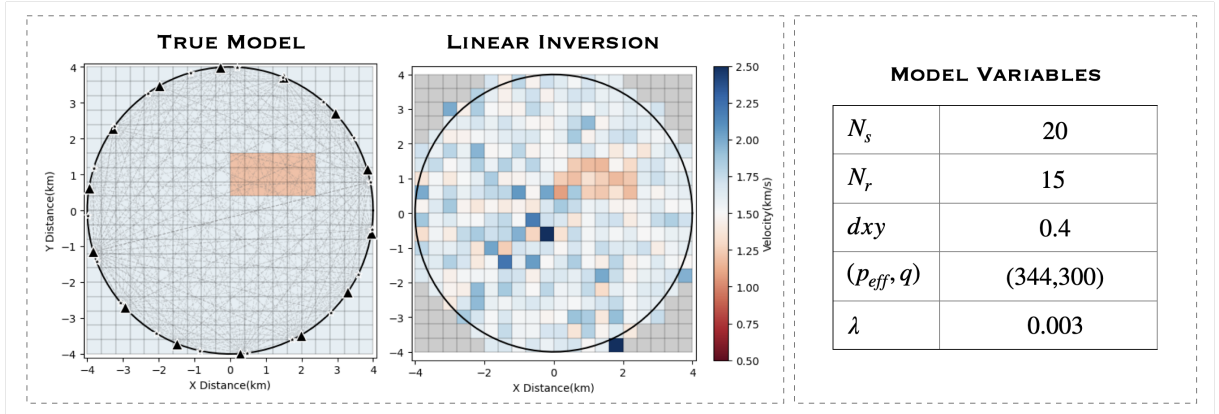


Figure 2.11: LI: Model 4

In Model 5 (Figure 2.12), a different source-receiver setup is used, with sources placed at random locations. All other settings remain the same as in Model 4. This highlights how the source-receiver configuration itself influences the resolution of the inversion.

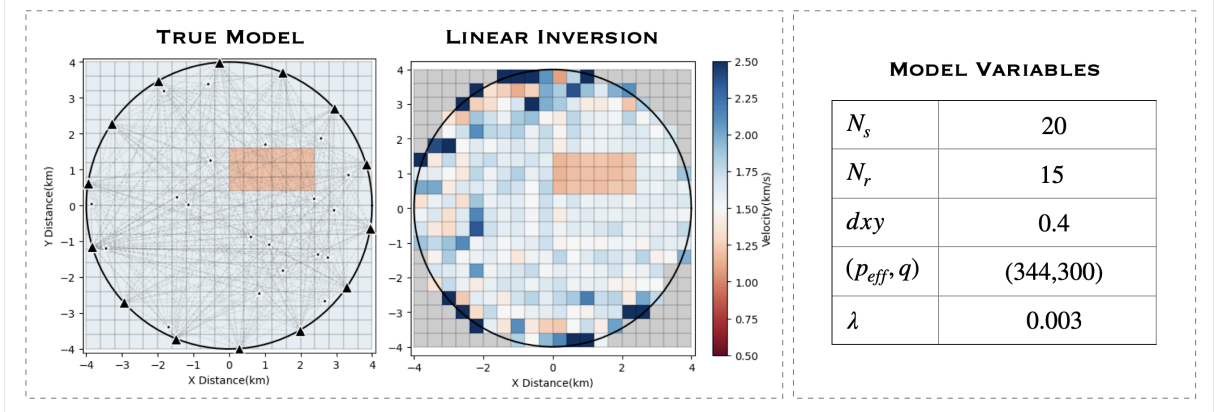


Figure 2.12: LI: Model 5

In Model 6 (Figure 2.13), the number of sources is reduced to 10, resulting in a total of 150 rays. Consequently, the number of covered parameters decreases to 318.

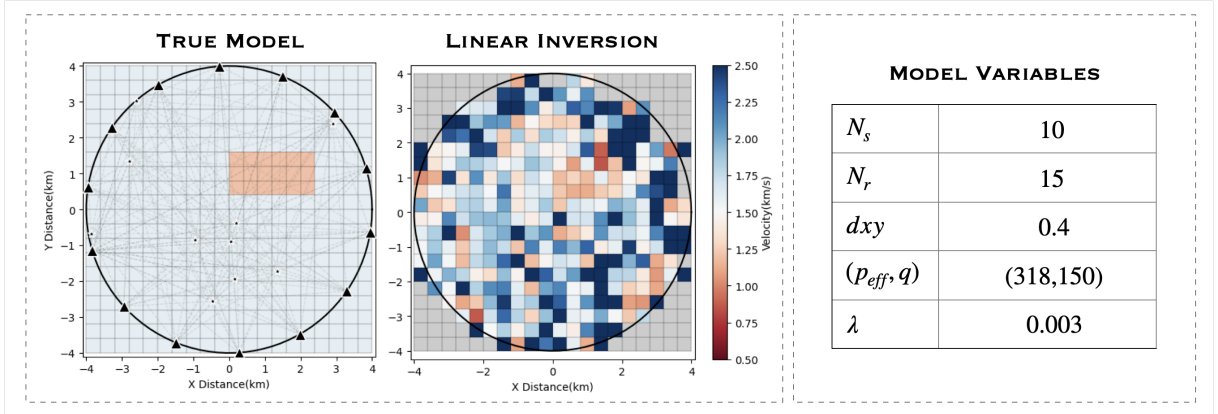
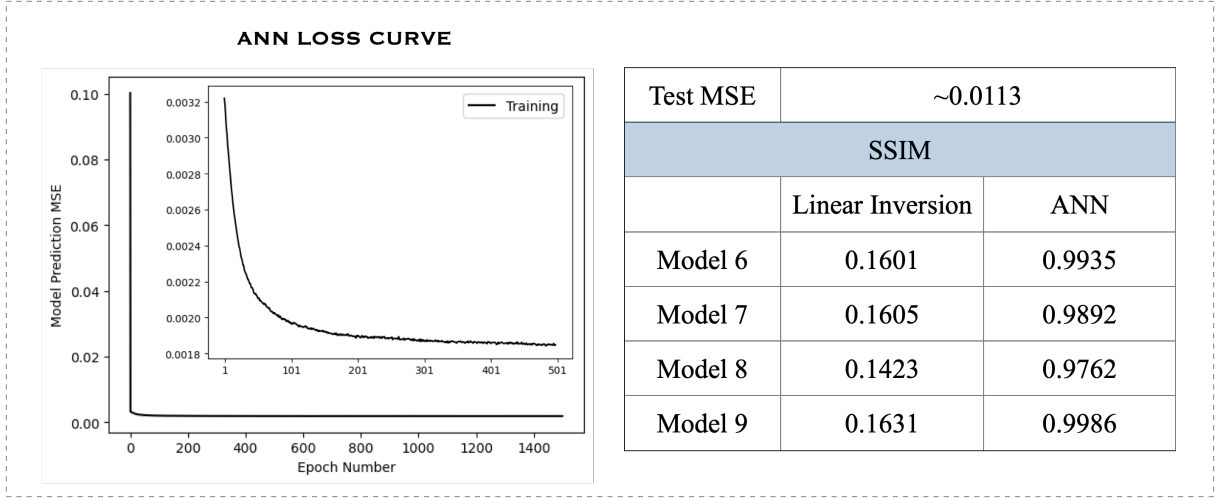


Figure 2.13: LI: Model 6

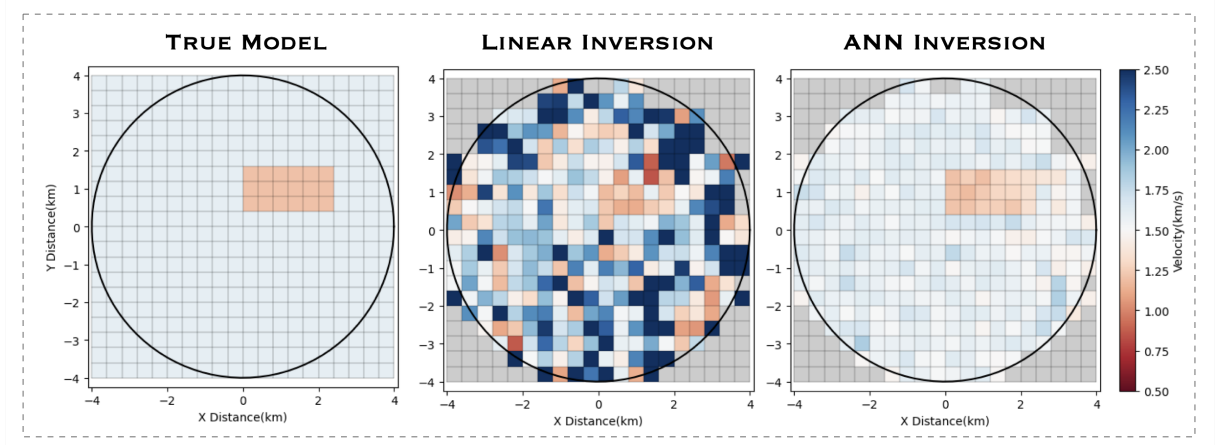
Linear Inversion vs ANN

We start off the comparison of inversion methodologies from Model 6 (Figure 2.13), where linear inversion seemingly fails. To ensure consistency, the same source-receiver setup from Model 6 is used across all comparisons in this section. As a result, all models in this section have 150 rays and 318 model parameters. The comparison is performed for different velocity profiles, and a quantitative evaluation of the results is conducted using structural similarity index (SSIM), as presented in Figure 2.14. A higher SSIM value indicates greater accuracy of the results.

The ANN training loss converges within 500 epochs (Figure 2.14). Since the source-receiver setup remains unchanged, the same trained network is used to predict all the results presented below.

**Figure 2.14:** LI vs ANN: Comparison Statistics

Model 6 (Figure 2.15) has a background velocity of 1.6 km/s and an anomaly of 1.2 km/s. The linear inversion yields an SSIM value of 0.1601, while the ANN approach achieves a much higher SSIM value of 0.9935.

**Figure 2.15:** LI vs ANN: Model 6

Model 7 (Figure 2.16) features two anomalies, each with a velocity of 1.2 km/s. The linear inversion results in an SSIM value of 0.1605, while the ANN approach significantly outperforms it with an SSIM value of 0.9892.

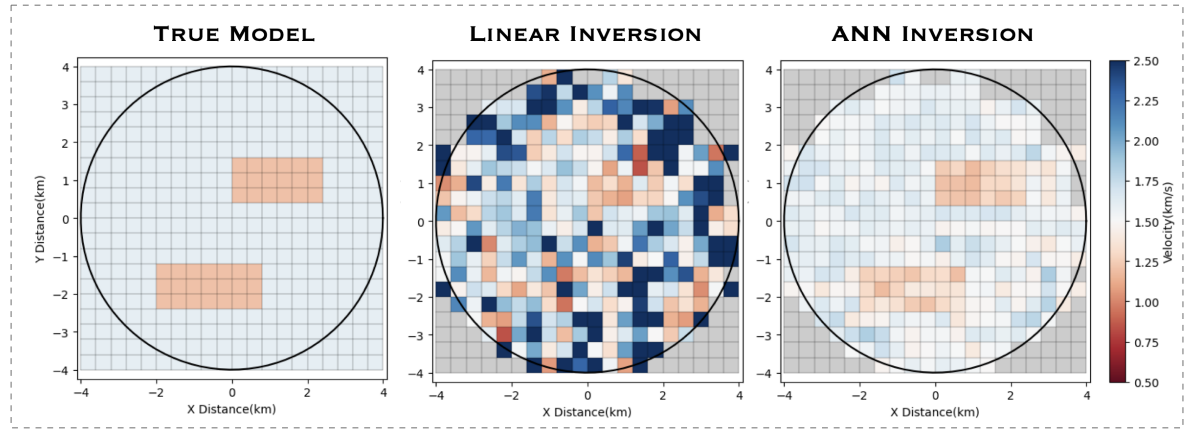


Figure 2.16: LI vs ANN: Model 7

Model 8 features a higher velocity contrast with an anomaly of 0.8 km/s. The linear inversion yields an SSIM value of 0.1423, while the ANN approach achieves an SSIM value of 0.9762.

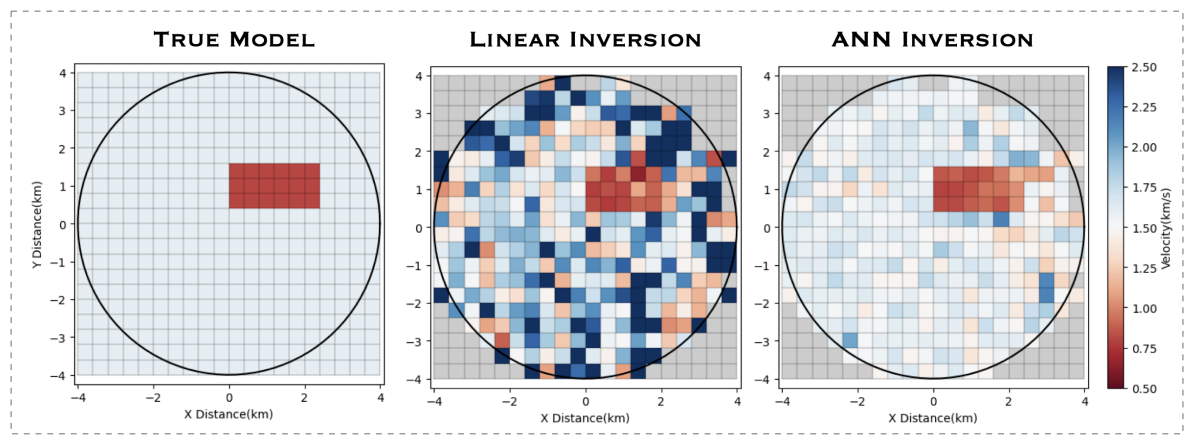


Figure 2.17: LI vs ANN: Model 8

Model 9 features a lower velocity contrast with a background velocity of 1.4 km/s and an anomaly of 1.2 km/s. The linear inversion results in an SSIM value of 0.1631, while the ANN approach significantly outperforms it, achieving an SSIM value of 0.9986.

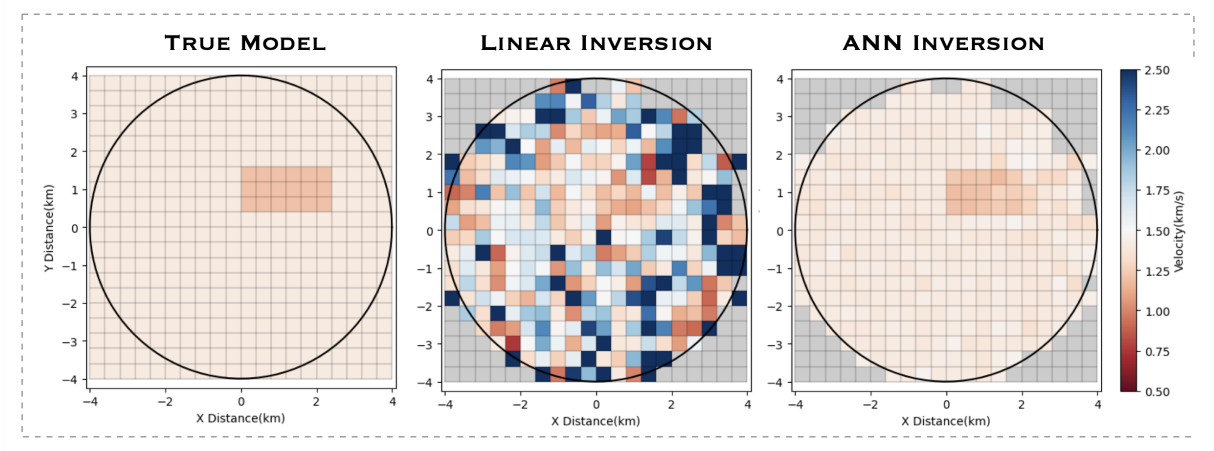
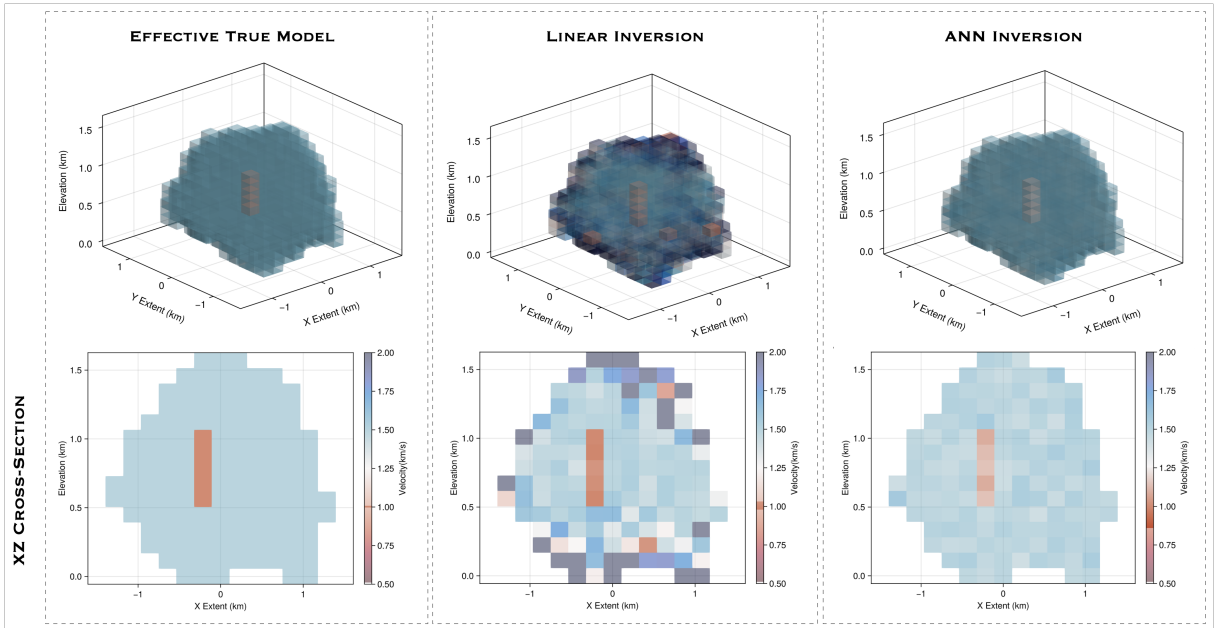


Figure 2.18: LI vs ANN: Model 9

2.4.2 3-D Domain

Results for the effective true model in the 3D domain with ray coverage are presented here. The vertical (XZ) cross-section containing the low-velocity anomaly is also shown. The effective true model varies depending on the source-receiver setup. Blocks with no ray coverage are kept transparent, and the low-velocity range is highlighted in the color map for better visualization.

Figure 2.19 illustrates the results for 3D setup 1, as shown in Figure 2.6. The model covers 1154 parameters with 1000 rays.


 Figure 2.19: LI vs ANN: 3D Setup 1 ²

²Plotted using [21]

Figure 2.20 illustrates the results for 3D setup 2, as shown in Figure 2.6. The model covers 1089 parameters with 500 rays.

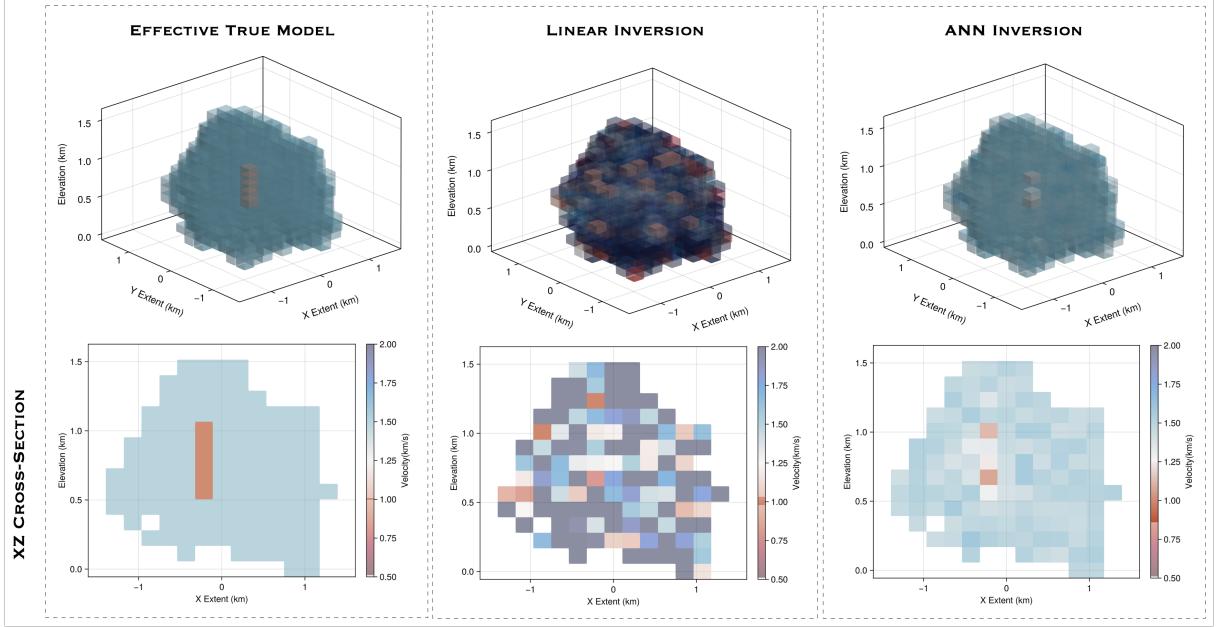


Figure 2.20: LI vs ANN: 3D Setup 2

2.5 Discussion

This chapter evaluates the efficiency of a machine learning approach compared to traditional linear inversion for sparse-ray tomography. The results presented in the previous section indicate that linear inversion struggles to accurately determine the velocity profile, whereas the ANN achieves better performance. It is important to clarify that by linear inversion, we refer to damped/regularized linear inversion. Figure 2.14 provides a quantitative comparison in 2D, showing that linear inversion yields low SSIM values, while the ANN achieves significantly higher values, highlighting its better performance.

When the model is overdetermined and well-covered by rays, linear inversion yields perfect results. Models 1 and 2 provide simple random example cases where linear inversion is well-resolved. However, our primary focus is on models incorporating a low-velocity anomaly indicative of subsurface structures, such as magma chambers. These cases are introduced from Model 3 onward. Model 3 successfully detects the anomaly with high accuracy being overdetermined.

Starting from Model 4, the linear inversion results become increasingly noisy. By Model 6, the noise level is so high that the presence of the anomaly is difficult to interpret from the figure alone. From this point onward—where linear inversion starts to struggle—we introduce a comparison with ANN inversion. We examine models with one and two anomalies, as well as variations in velocity contrast.

For Model 6, where linear inversion produces highly noisy results, ANN inversion delivers a much clearer detection of the anomaly, as indicated by the high SSIM score. In

Model 7, where linear inversion fails to provide any indication of the second anomaly, ANN inversion successfully detects both anomalies.

In Model 8, we consider a higher velocity contrast. Although the linear inversion results may appear visually improved compared to previous cases, the SSIM score is lower. This likely stems from larger error differences introduced by the increased velocity contrast in certain model parameters.

Model 9 examines a low-velocity contrast scenario, with just a 0.2 km/s reduction, representing a 14% decrease. Even under these conditions, ANN inversion performs well, whereas linear inversion fails. This is especially important in detecting anomalies with not a huge velocity difference.

In the 3D setup, both approaches successfully resolve the anomaly in setup 1, though the ANN results are less noisy. In setup 2, where the number of rays is reduced by half, linear inversion fails to detect the anomaly and produces a highly noisy result. In contrast, ANN inversion remains significantly better, successfully identifying the anomaly, though not perfectly, against the homogeneous background.

The performance of the ANN in both 2D and 3D domains suggests that it is a more effective approach for the sparse-ray tomography problem under study. While ANN training requires more computational time initially, predictions can be obtained instantly using the trained network, with no additional computational cost. Our model is trained on a diverse set of random velocity distributions without assuming a predefined velocity profile. Unlike some linear inversion techniques, such as Bayesian inversion, ANN inversion does not require an initial model. This is a significant advantage, as the choice of an initial model strongly influences traditional inversion outcomes. ANN inversion benefits from exposure to a large number of datasets, enabling it to learn patterns more effectively. Additionally, different ANN models can be trained for different source-receiver setups, optimizing performance for specific configurations.

In the following chapter, we aim to address one of the primary assumptions made in this study—the straight-ray approximation.

Chapter 3

Full Waveform Inversion

3.1 Introduction

As we move toward addressing real datasets and challenges, it becomes necessary to account for phenomena such as reflection and refraction, which were previously neglected in travel-time tomography (Chapter 2). Full-waveform modeling handles this by ensuring realistic modeling of wave physics. Full-waveform tomography utilizes entire waveforms for inversion. In this work, we test the viability of a machine learning approach to full-waveform tomography.

We forward model simplified synthetic cases to generate paired waveform-velocity datasets, which are required for training neural networks. Before getting into inversion, we first attempt to reconstruct the waveforms using an unsupervised neural network based on an autoencoder. This motivates the use of encoded waveforms for velocity prediction. To predict velocity, we implement an encoder followed by a fully connected network for velocity prediction.

This chapter begins with the methodology used for forward modeling. This is followed by waveform reconstruction, which complements the design of the neural network architecture for inversion. The model setups used are integrated within the section for better flow. Finally, the results of the inversion are presented.

3.2 Methodology and Setup

3.2.1 Forward Modeling

Forward modeling is performed using Salvus [10] 2024.1.2, a high-performance Python-based simulation framework built on the SEM. This method accurately solves wave propagation [22] problems by combining the flexibility of finite elements with the accuracy of spectral methods. For more details on SEM, refer to Appendix C.

Elastic wave propagation is simulated for 6 seconds in a medium with a constant density of 2670 kg/m^3 , using an explosive Ricker wavelet source of 5Hz. Random velocity profiles are considered for the P-wave velocity (VP). S-wave velocity (VS) is approximately 0.588 times the P-wave velocity. Three elements per wavelength with a quadratic

interpolation is used for SEM. An absorbing boundary layer with a thickness of one wavelength is implemented to prevent wave reflections from the boundaries.

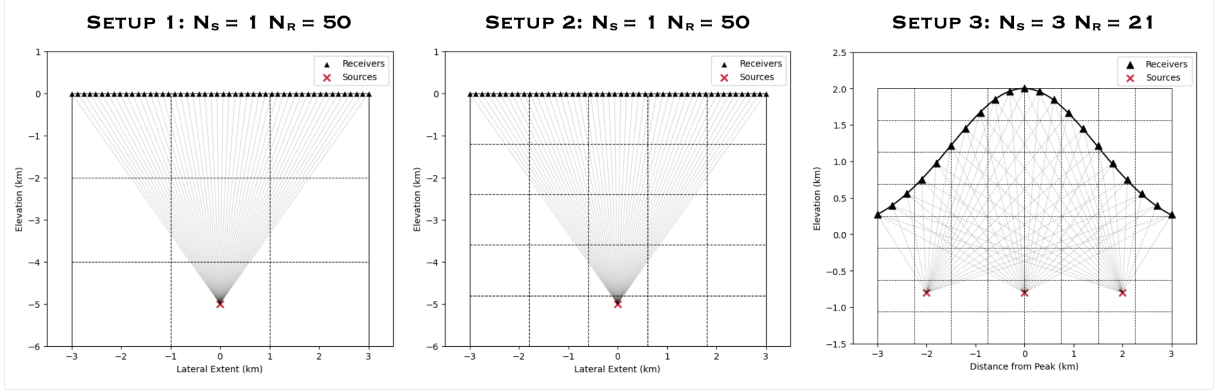


Figure 3.1: Model Setups for Full-Waveform Modeling

We consider three different source-receiver setups for the modeling, as illustrated in Figure 3.1. Here, N_s represents the number of sources and N_r the number of receivers.

All setups span 6 km laterally, from -3 km to +3 km. In setups 1 and 2, we use flat-topography models with a depth of 6 km. A single source is located at (0, -5) km, and 50 receivers are evenly distributed along the flat free surface. Setup 1 employs a 3×3 meshing, resulting in 9 velocity blocks, whereas Setup 2 uses a finer 5×5 meshing with 25 velocity blocks. Setup 3 introduces a 2D Gaussian topography with an elevation of 2 km and a depth of 1.5 km. Three sources are positioned at (-2, -0.8) km, (0, -0.8) km, and (+2, -0.8) km, while 21 receivers are evenly distributed along the topography. This setup adopts an 8×8 meshing, dividing the model into 64 velocity blocks. However, the blocks above the topography are removed, resulting in an actual number of 52 velocity blocks.

For each setup, 10,000 simulations with random velocity profiles in range 1-2 km/s are carried out to generate the seismograms, from which the Y-component velocity seismogram is retrieved. This yields 2D seismogram-velocity pairs for ANN training.

Velocity Array - Data Densification and Gaussian Smoothing

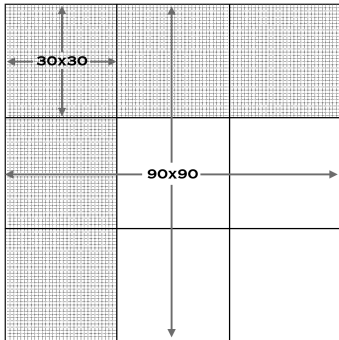


Figure 3.2: Velocity densification example for 3×3 mesh

Since we use coarse velocity profiles for testing purposes, the velocity grid resolution must be increased to ensure smoother wave simulations in Salvus. Each velocity block value is repeated 90 times within its spatial constraints, with 30 repetitions along both the X and Y axes. A Gaussian smoothing filter with a sigma value of 5 is then applied to further refine the model.

For instance, in the case of a 3×3 meshing, as illustrated in Figure 3.2, the original (3,3) velocity array is transformed into a (90,90) array with smoothing before being passed to Salvus.

It is important to note that for ANN training, the original array—without densification or smoothing—is used

Salvus

As described above, a densified and smoothed velocity array is passed to Salvus. Let's consider an example of a random velocity profile using setup 1, as shown in Figure 3.3, for a clearer understanding.

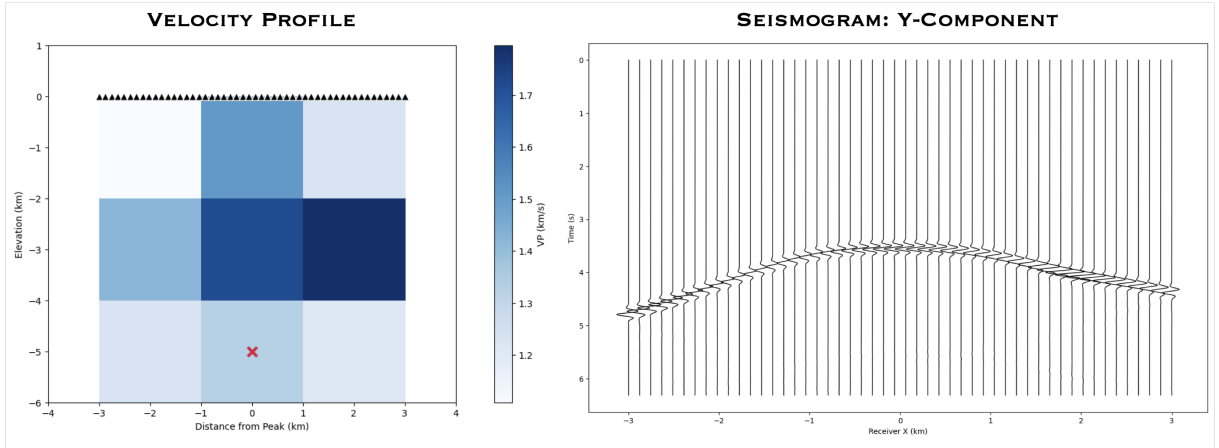


Figure 3.3: Random velocity profile with the Velocity-Y seismogram generated using Salvus.

Salvus processes the finite element meshing and ABC elements, as illustrated in Figure 3.4. The basic mesh information is provided below. The number of points refers to the discrete nodes in the mesh, while the number of elements refers to the individual finite elements (quadrilaterals in this case) formed by connecting those points. The model is extended by approximately 0.3 km at each boundary to implement the absorbing layers.

- Shape mapping order: 2
- Number of points: 129479
- Number of elements: 32190
- Spatial extent in X dimension: min = -3.336 km, max = 3.336 km
- Spatial extent in Y dimension: min = -6.300 km, max = 0 km

Salvus initiates the simulation at -0.312s to ensure that the source is developed by the zeroth time. The wavefield for the specific velocity profile propagates as shown in Figure 3.5. The snapshots illustrate how the y-component of the velocity seismogram propagates through the velocity profile. Moving from left to right, the time gap between snapshots is 0.3s, while moving from bottom to top, the time gap is 1.5s. From the snapshots in the second row from the bottom (from 1.5s to 2.7s), it is evident that the absorbing boundary does not cause reflections. At 3.0s, upon reaching the free surface boundary, reflections begin to appear, and we can observe their propagation until the 6th second. The final output, the velocity-Y seismogram for the corresponding velocity profile, is shown in Figure 3.3.

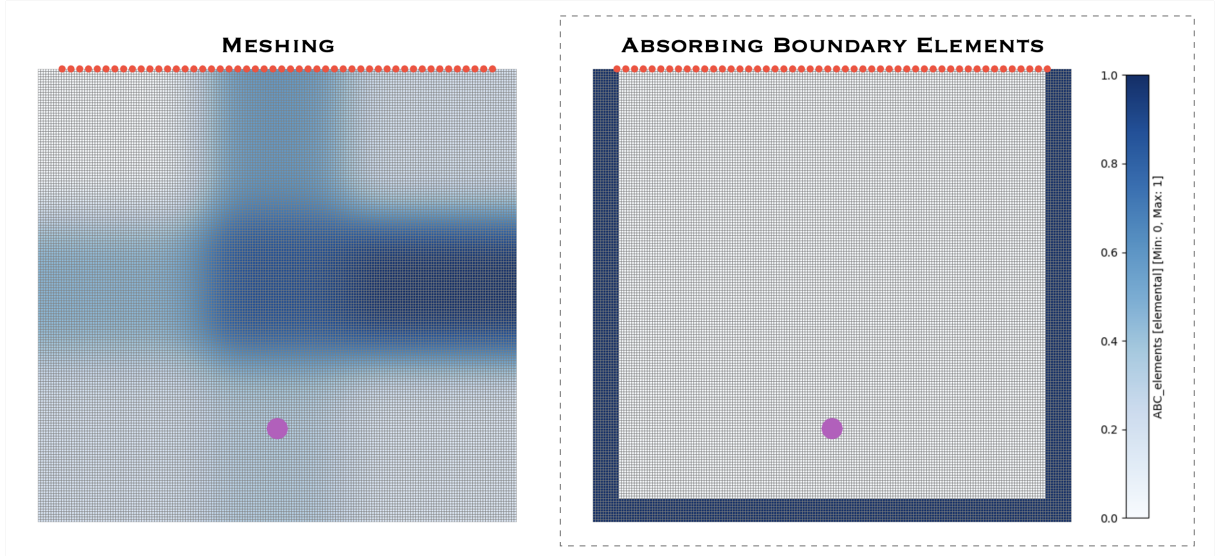


Figure 3.4: Meshing and ABC elements for the velocity profile shown in Figure 3.3.

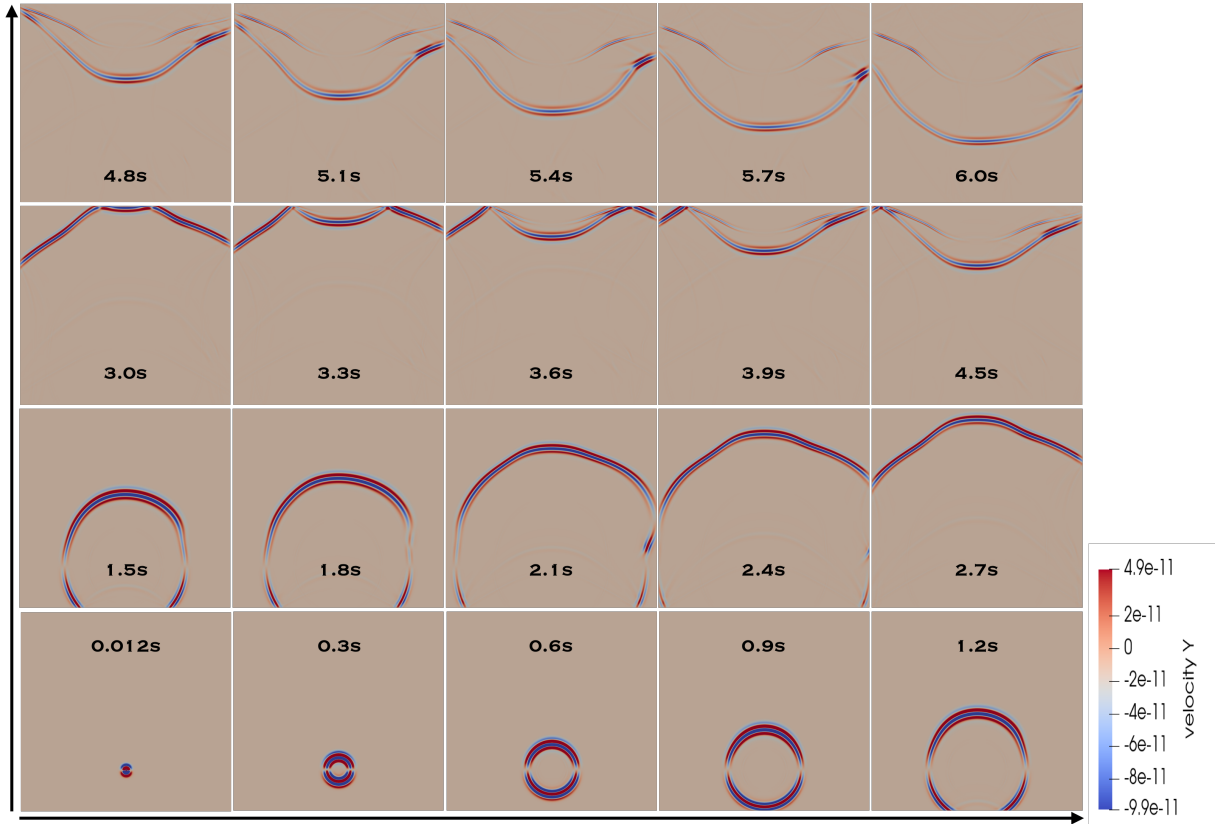


Figure 3.5: Some wavefield snapshots illustrating Velocity-Y propagation for the velocity profile shown in Figure 3.3.

Waveform Preprocessing: Downsampling and Slicing

The wavefield is simulated for 6 seconds with a timestep of 0.002 seconds, resulting in 3157 time steps per trace. To reduce the data, the waveform is downsampled to 10%, leaving only 316 elements per trace, and the timestep is increased to 0.02s. The Nyquist frequency is calculated as:

$$f_N = \frac{1}{2 * \Delta t} = \frac{1}{2 * 0.02} = \frac{50}{2} = 25 \text{ Hz}$$

Since the source frequency of 5Hz is below the Nyquist frequency of 25Hz, key information remains intact despite the 90% reduction. This downsampling process ultimately enhances the effectiveness of ANN training. An example is shown in Figure 3.6, where the true and downsampled waveforms overlap almost perfectly.

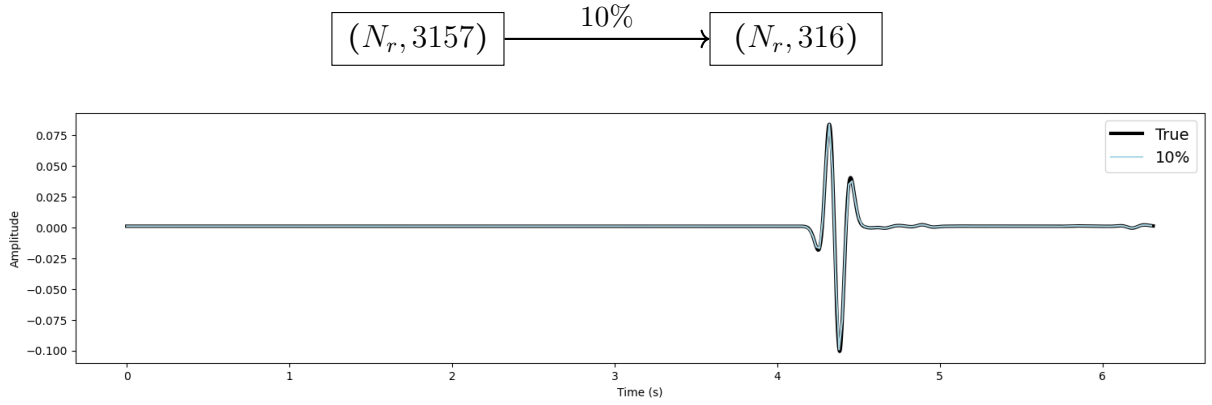


Figure 3.6: Comparison of an example true waveform with its 10% downsampled version

After downsampling, the waveforms are sliced along the time axis to preserve only the essential signal information, which is then used as input to the ANN. Slicing differs for different model setups.

3.2.2 Waveform Reconstruction

Before delving into inversion for velocity, let's first explore waveform reconstruction. The motivation behind this step is to assess whether encoded waveforms can be leveraged for velocity prediction. Waveform reconstruction offers insight into the capability of neural networks to capture key characteristics of seismic waveforms. By encoding waveforms into a latent space, we aim to create a compact representation that retains only the sufficient information required to predict velocity profiles.

Waveform reconstruction is implemented using autoencoders, an unsupervised learning technique. As shown in Figure 3.7, we employ an autoencoder [23] consisting of an encoder and a decoder, both constructed with 2D convolutional layers using a kernel size of 3×3 . The encoder returns a latent representation of the original waveform by extracting its essential features. The encoder extracts essential waveform features, returning a latent representation that progressively reduces dimensionality through 2D

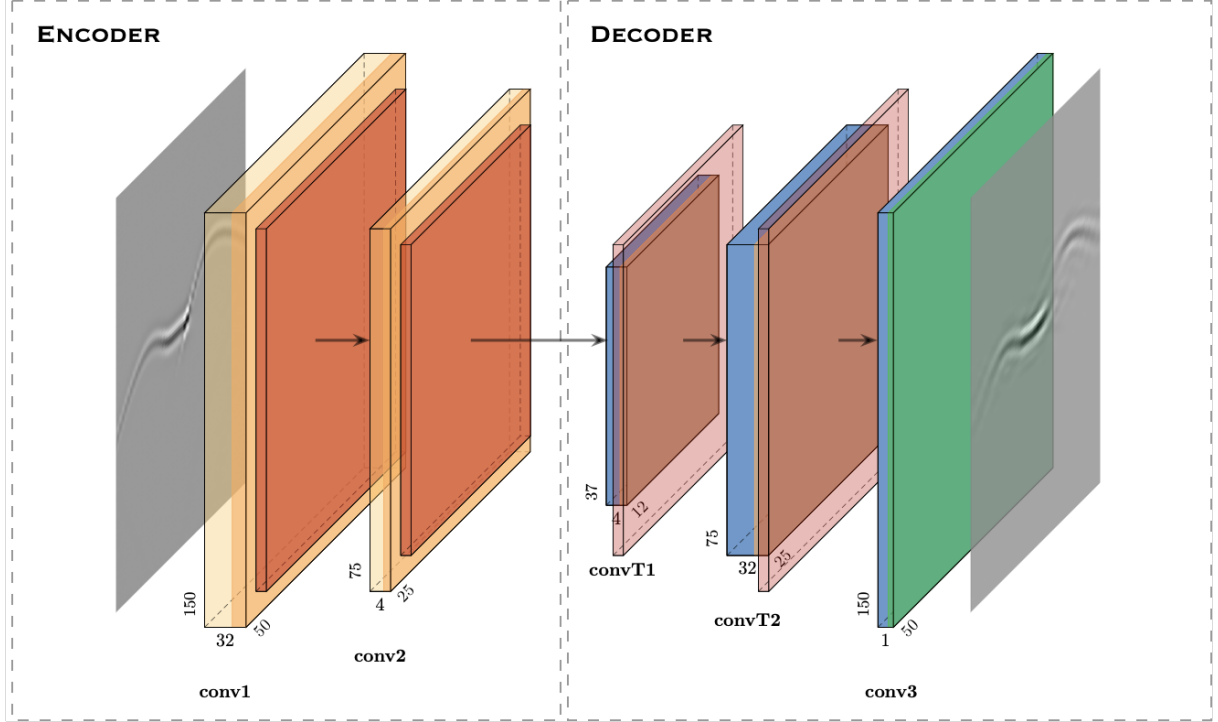


Figure 3.7: Autoencoder architecture for waveform reconstruction in Setup 1¹. The dimensions shown represent the sizes of the convolutional layers, which are reduced by a factor of 2 after each max-pooling layer. The resulting dimensions after max-pooling are indicated below.

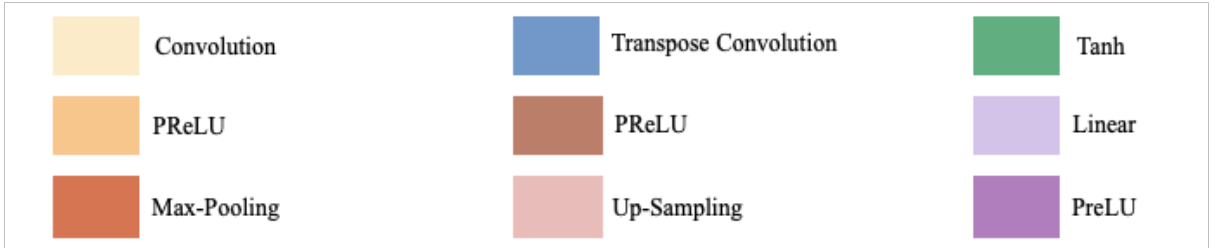


Figure 3.8: Color maps for the ANN plots

convolutional layers with PReLU activation and max-pooling. Figure 3.7 shows the dimensions of the convolutional layers, which are halved each time they pass through a max-pooling layer. Max-pooling layers with a kernel size of 2×2 is used for this purpose. The decoder reconstructs the original waveform from this latent representation using 2D transpose convolutional layers with PReLU activation and upsampling, followed by a final convolutional layer with Tanh activation for fine-tuning.

The model is trained on a dataset of 10000 seismograms, with validation and test sets containing 1500 samples each. Training is optimized using the Adam optimizer [24], with the mean squared error (MSE) as the loss function.

A test example from Setup 1, along with its latent representation produced by the encoder, is shown in Figure 3.9. The latent space consists of four channels, resulting in

¹Plotted using [25]

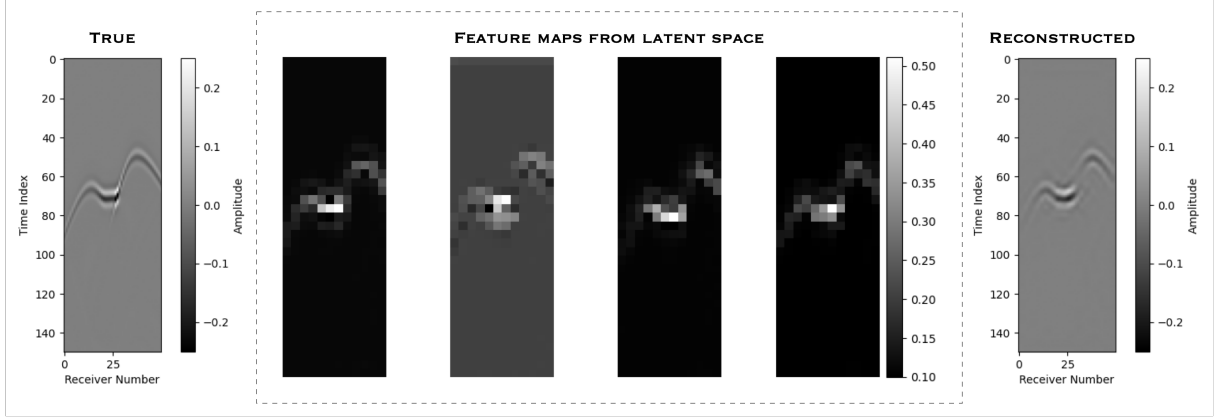
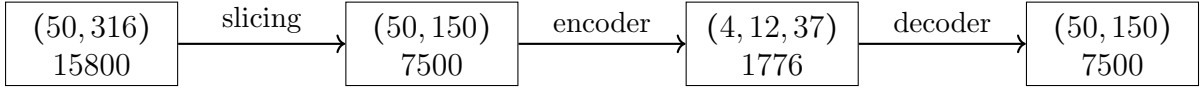


Figure 3.9: Autoencoder feature maps for an example case using Setup 1

four feature maps with distinct activations. The reconstruction confirms that the encoder can successfully extract the key features of the waveforms.



3.2.3 Inversion

As discussed in Section 3.2.2, an encoder composed of 2D convolutional layers effectively extracts the necessary features for waveform reconstruction. This motivates the use of a similar encoder for waveform encoding, followed by a fully connected (FC) stack for velocity prediction. The encoder consists of three convolutional layers with PReLU activation functions, while the FC stack includes two hidden layers. The model is trained using the Adam optimizer with MSE as the loss function. Since preserving additional spatial information is not critical for velocity prediction, the dimensionality of the latent space can be further reduced. To achieve this, an extra convolutional layer is added to the encoder, further compressing the latent representation. As a result, the encoder reduces the original waveform by a factor of 16. The same example from the previous section, now used for velocity prediction instead of reconstruction, is illustrated in Figure 3.11.

The size of the FC input layer depends on the degree of waveform compression. Different slicing approaches are used for Setups 1 and 2 compared to Setup 3, leading to variations in the dimensionality of the convolutional layers. These differences are outlined below.

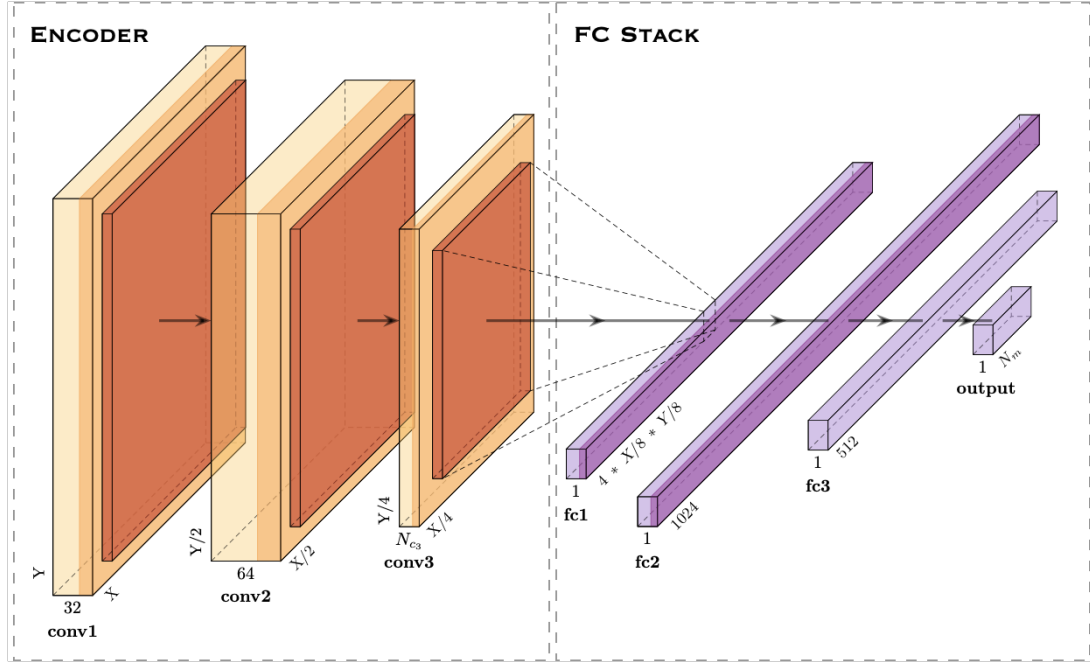


Figure 3.10: ANN Architecture for Inversion. The dimensions shown represent the sizes of the convolutional layers, which are reduced by a factor of 2 after each max-pooling layer.

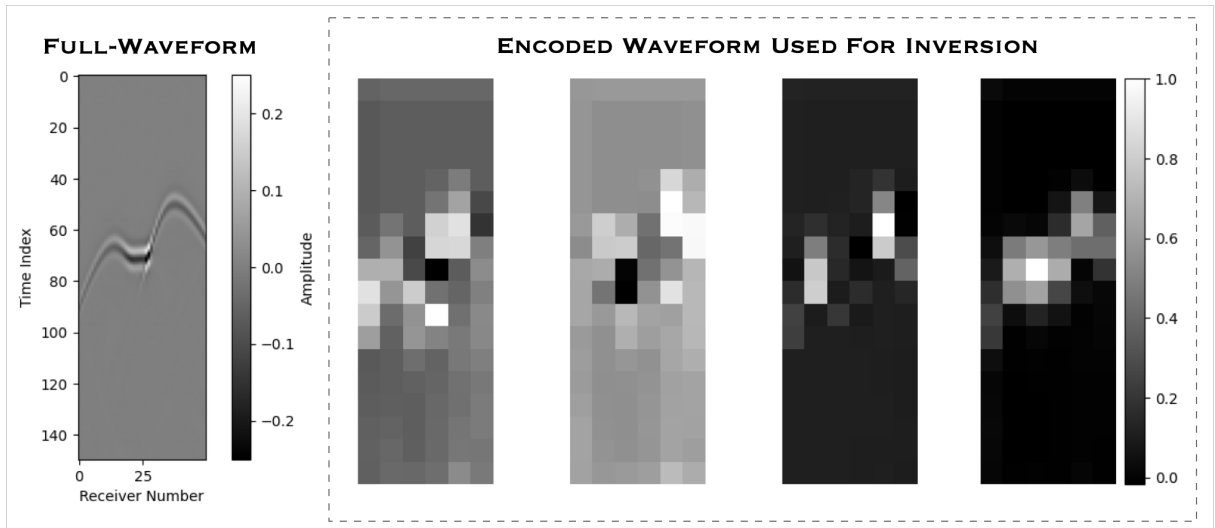


Figure 3.11: Encoded Waveform for Inversion. Compare with the feature maps in Figure 3.9 used for waveform reconstruction.

For Setups 1 and 2, the flattened encoded waveform has a size of 432. The waveform is sliced from 2.5s to 5.5s, as this interval contains the essential information for all waveforms. The input channel has a size of 1 since there is only a single source. The third convolutional layer produces 4 distinct output features, which are used for velocity prediction.

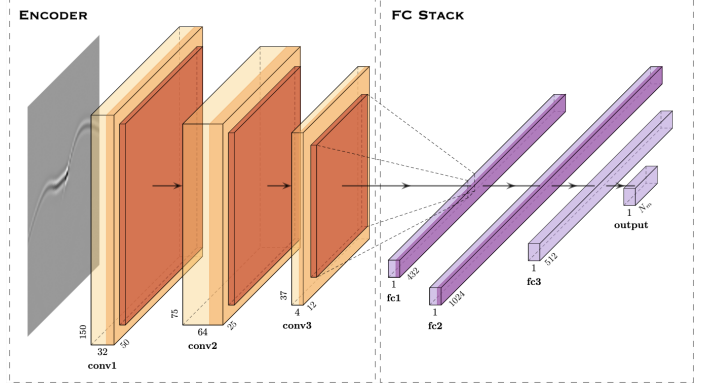
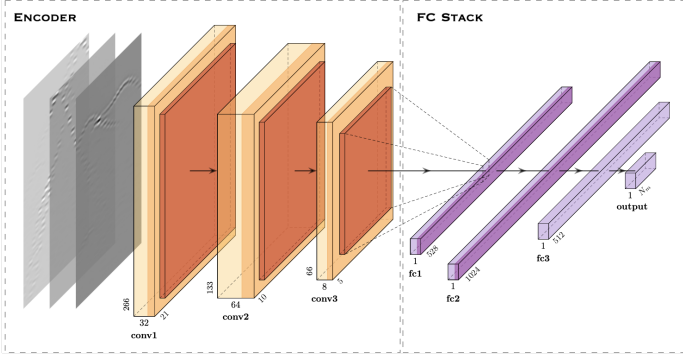
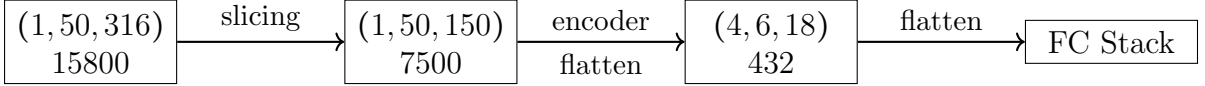
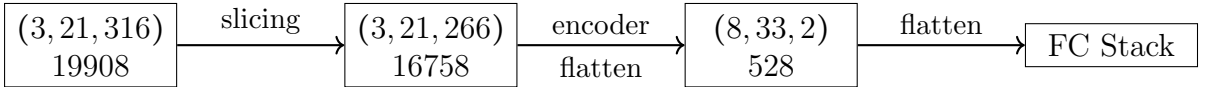


Figure 3.12: ANN Architecture for Setups 1 and 2



For Setup 3, the flattened encoded waveforms has a size of 528. The waveform is sliced from 1s to 6s. The input channel is of size 3 as there are 3 source. The third convolutional layer produces 8 distinct output features, accounting for the added complexities.

Figure 3.13: ANN Architecture for Setup 3



In Figures 3.12 and 3.13, the dimensions represent the sizes of the convolutional layers, which are reduced by a factor of 2 after each max-pooling layer. The flow chart illustrates the dimensions after propagation through all layers.

3.3 Results

Some inversion results for the three setups, as shown in Figure 3.1, are presented in this section. We compute SSIM values for quantitative interpretation, both for the entire model and for the model with redundant velocity blocks removed (specifically, the last row).

3.3.1 Setup 1

As illustrated in Figure 3.1, setup 1 consists of 1 source, 50 receivers and 9 velocity parameters. We present results for two random test velocity profiles and a velocity profile containing an anomaly.

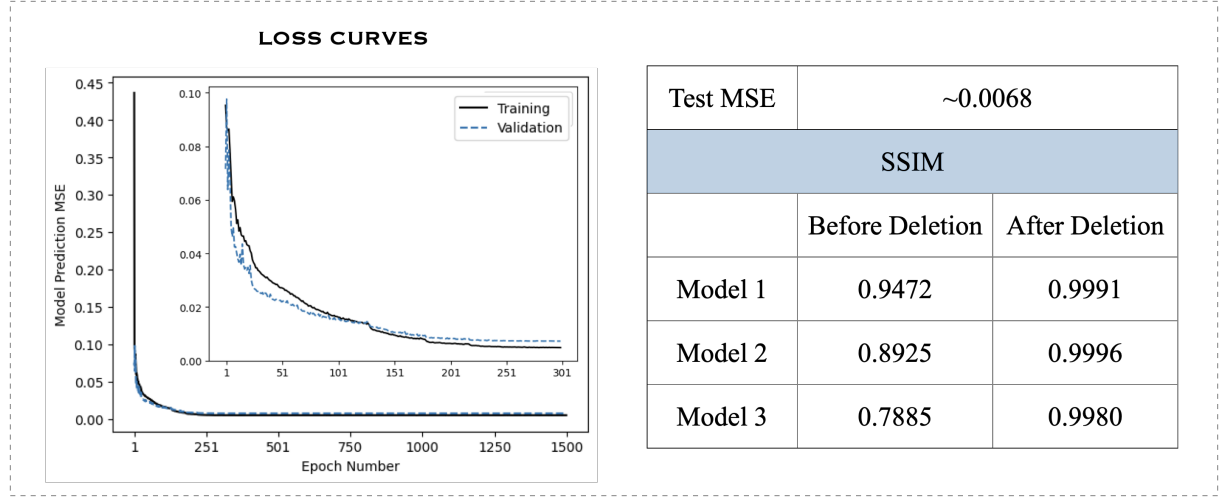


Figure 3.14: Setup 1: Statistics

Models 1 and 2, shown in Figures 3.15 and 3.16, are test dataset examples. The velocity values ranges from 1 to 2 km/s. The ANN yields an ssim value of 0.9472 before deletion and 0.9991 after deletion for model 1. For model 2, it achieves an ssim value of 0.8925 before deletion and 0.9996 after deletion.

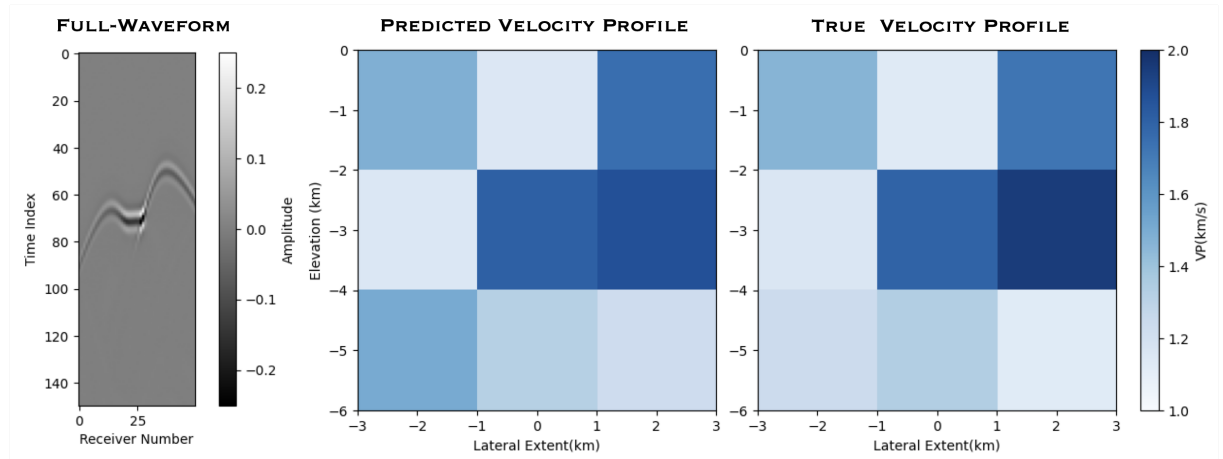


Figure 3.15: Setup 1 with a random test velocity profile: Model 1

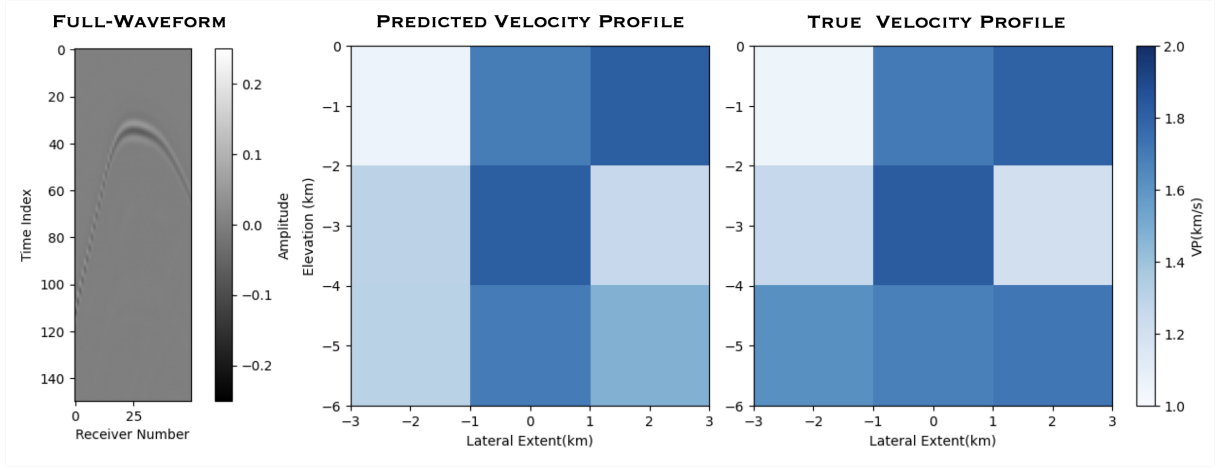


Figure 3.16: Setup 1 with a random test velocity profile: Model 2

Model 3, shown in Figure 3.17, features a homogeneous background velocity of 1.6 km/s, with a low-velocity anomaly of 1.2 km/s. The ANN yields an SSIM value of 0.7885 before deletion and 0.9980 after deletion.

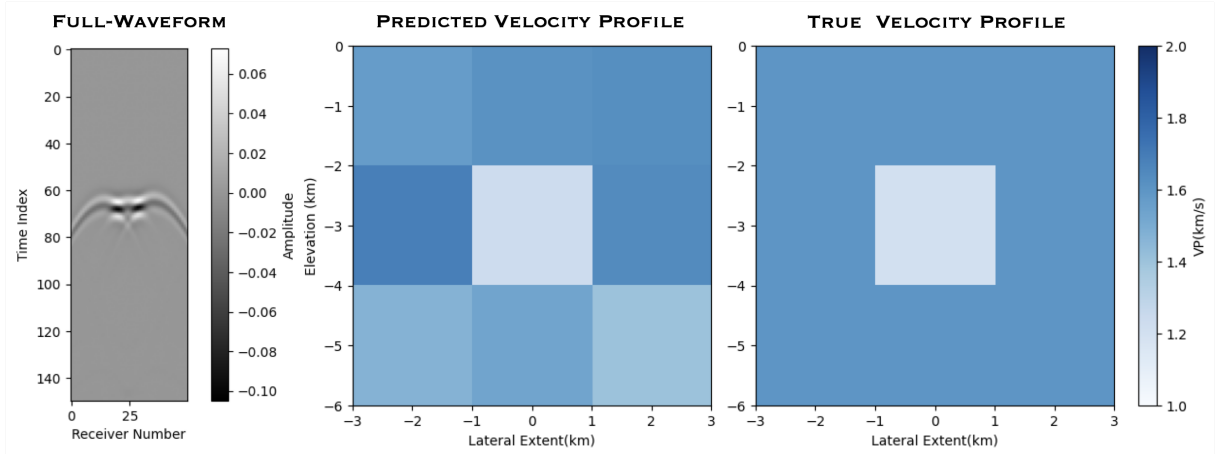
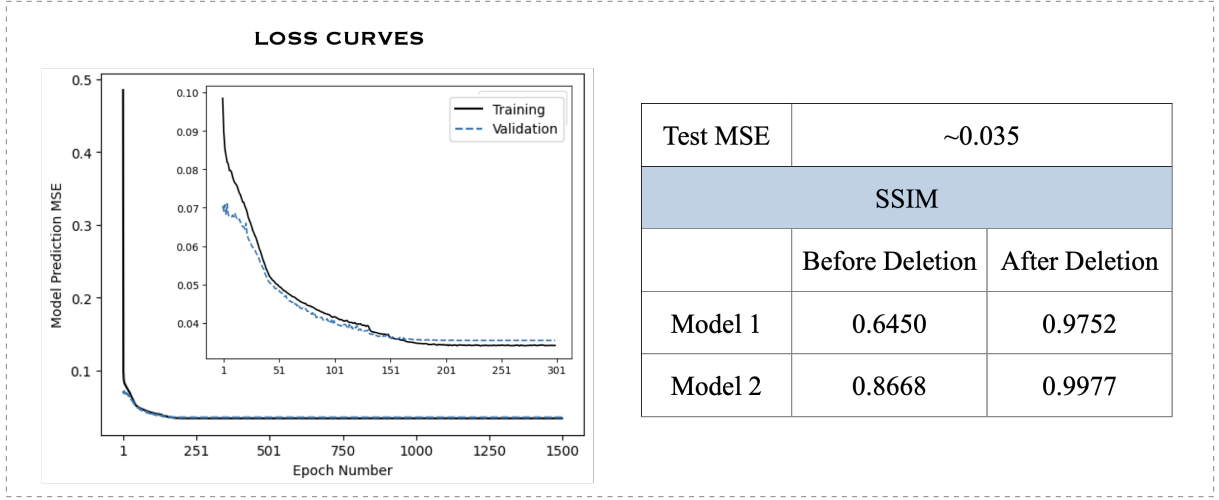


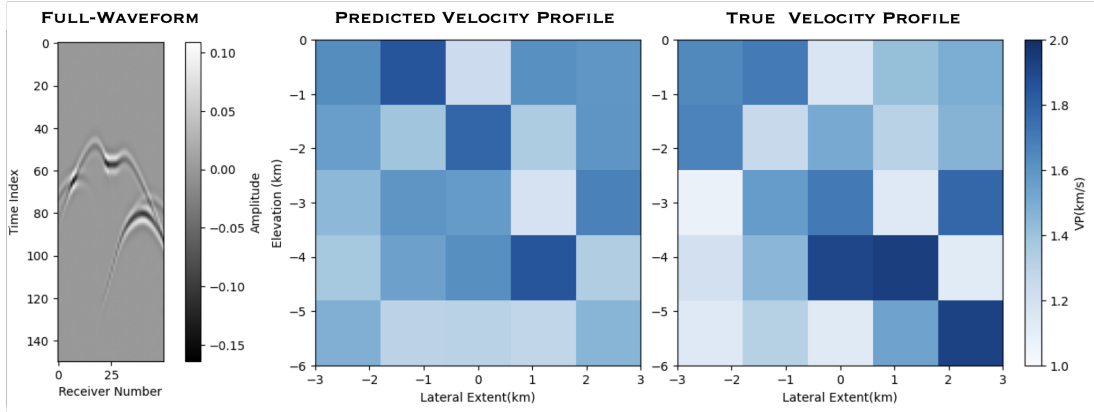
Figure 3.17: Setup 1 with a velocity profile containing an anomaly: Model 3

3.3.2 Setup 2

Setup 2 consists of 1 source, 50 receivers and 25 velocity parameters. We present results for one random test velocity profiles and a velocity profile containing an anomaly.

**Figure 3.18:** Setup 2: Statistics

Model 1, shown in Figure 3.19, is a test dataset example. The ANN obtains an ssim value of 0.6450 before deletion and 0.9752 after deletion.

**Figure 3.19:** Setup 2 with a random test velocity profile: Model 1

Model 2, shown in Figure 3.20, features a homogeneous background velocity of 1.6 km/s, with a low-velocity anomaly of 1.2 km/s. The ANN yields an SSIM value of 0.8668 before deletion and 0.9977 after deletion.

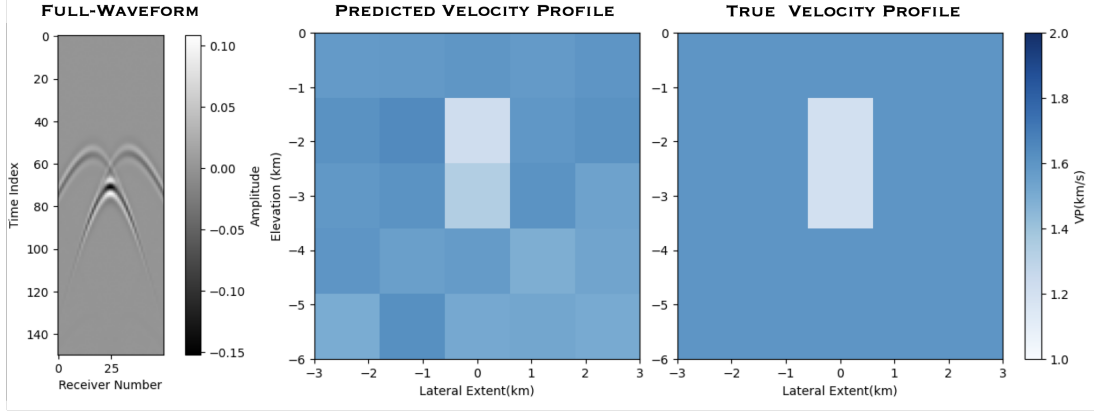


Figure 3.20: Setup 2 with a velocity profile containing an anomaly: Model 2

3.3.3 Setup 3

Setup 3 consists of 3 sources, 21 receivers and 52 velocity parameters. We present the result for a velocity profile containing an anomaly.

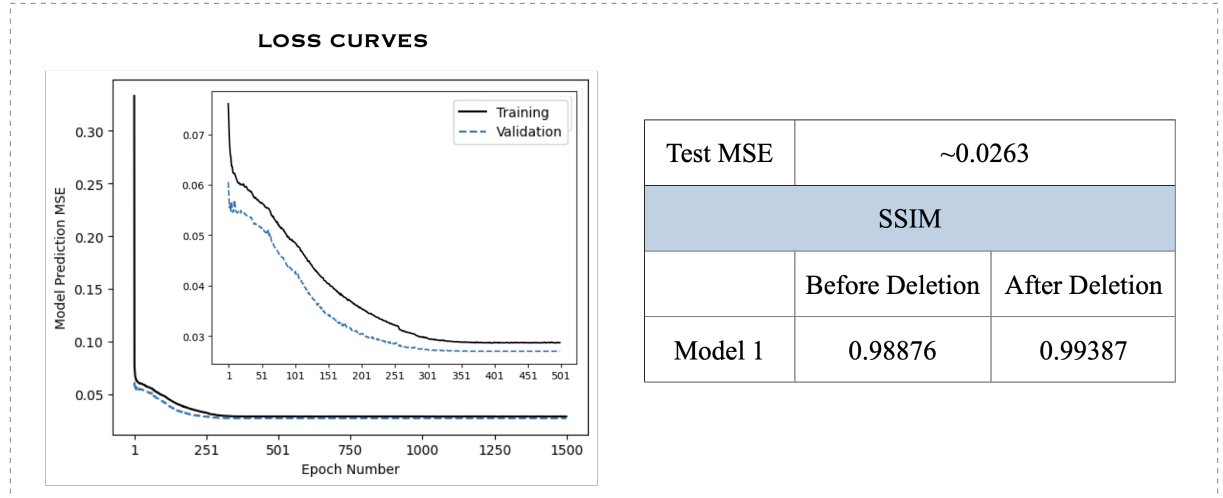


Figure 3.21: Setup 3: Statistics

Model 1, shown in Figure 3.22, features a homogeneous background velocity of 1.6 km/s, with a low-velocity anomaly of 1.2 km/s. The ANN yields an SSIM value of 0.98876 before deletion and 0.99387 after deletion.

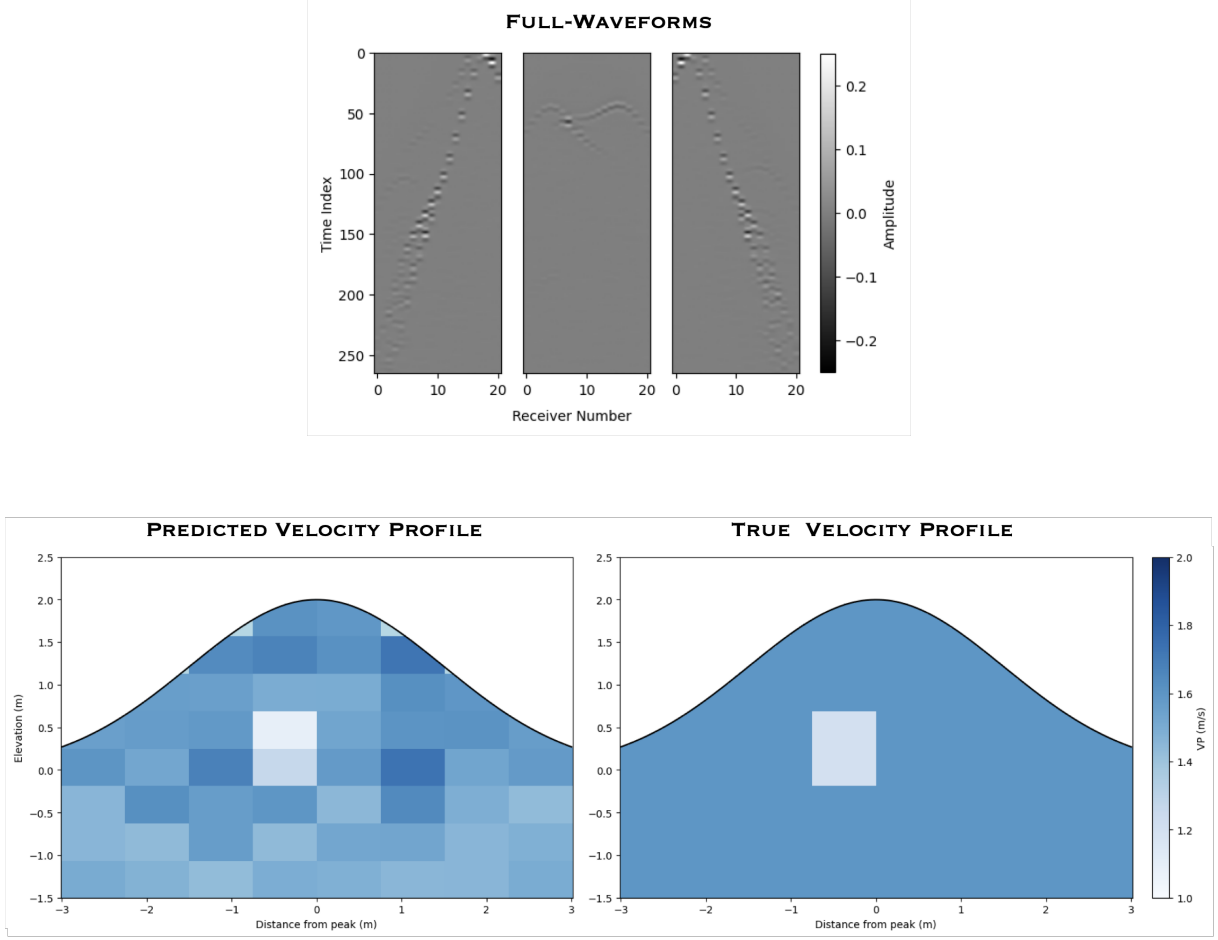


Figure 3.22: Setup 3 with a velocity profile containing an anomaly: Model 1

3.4 Discussion

This chapter evaluates the potential of a machine learning approach to FWI. The results presented in the previous section demonstrate promising potential for further advancements.

We begin the modeling with coarse velocity profiles, as in Setup 1, which consists of 9 velocity blocks. In Setup 1, we use a coarse velocity profile with 9 velocity blocks. The ANN training yields a test MSE of 0.0068, indicating a relatively low error between the predicted and actual velocity profiles. The SSIM value for this setup is notably high, showing that the model is able to capture the structural features of the waveform accurately. When the blocks with no ray coverage are removed, the SSIM value increases, suggesting that these regions do not contribute to the model’s ability to accurately predict velocity. This indicates that removing unnecessary blocks improves the model’s performance by focusing only on regions with relevant information.

For Setup 2, the velocity profile is refined by increasing the number of velocity blocks to 25. This results in a higher test MSE of 0.035, which reflects a greater challenge in learning the more complex velocity structure. However, the SSIM value remains significantly high, indicating that the structural similarity between the predicted and true velocity profiles

is still maintained. The increase in MSE is expected when refining the profile, as more detailed information is needed to accurately model the velocity structure.

In Setup 3, the velocity profile is further refined to 52 blocks, and a topography is added. The test MSE increases to 0.0263, which is higher than in Setup 2, but still within an acceptable range. The SSIM value remains high, suggesting that the model is still able to preserve the essential features of the waveform despite the increased complexity. The encoder now has 3 input channels corresponding to 3 sources, which helps the model capture more detailed features from the waveforms. This suggests that the ANN approach is scalable and capable of handling more complex structures.

The general trend shows that the model performs well across all setups, but as the complexity of the velocity profile and the model setup increases, the MSE also increases. This suggests that while the model is able to handle more intricate structures, the challenge of capturing finer details in the velocity profile increases. Nevertheless, the high SSIM values across setups indicate that the model effectively learns the key structural features of the waveforms, making it a promising approach for FWI.

Chapter 4

Discussion

As presented and discussed in detail in Chapter 2, an ML approach using ANN outperforms damped/regularized linear inversion in sparse ray tomography. However, for the modeling, we considered several simplifying assumptions such as straight-ray-based travel-times, known source location and noise-free conditions. These assumptions limit our ability to handle real-world datasets.

To move closer to more realistic scenarios, we addressed one of the simplifying assumptions—the straight-ray assumption—by incorporating physics-based wave equation modeling. In Chapter 3, we transitioned to FWI, where we modeled and studied various simple setups. The use of an encoder-based ANN yielded promising results, highlighting the potential of ML in FWI. If fully developed and successful, machine learning could offer computational speed advantages, as traditional FWI is known to be computationally intensive. Although machine learning requires training time, once the model is trained, predictions can be made almost instantly.

ML requires large training datasets, which may not always be available for real-world scenarios. In the case of synthetic testing, the dataset generation took considerable time due to the complexity of the task. Salvus solves the wave equation with high precision using SEM. The computation time was improved by parallel computing, processing 5 simulations at a time, which reduced the overall time to about 20-30 hours to generate 10,000 datasets for each setup.

We used random velocity profiles for ANN training with no biases, which ensures that the model is not influenced by preconceived assumptions about the velocity structure. Even with this approach, the models are capable of detecting anomalies effectively in both travel-time tomography and FWI. This is an advantage over traditional methods, where the choice of an initial model plays a critical role in determining the outcomes of the inversion. In traditional inversion, an incorrect or poorly chosen initial model can lead to inaccurate results or convergence to local minima. In contrast, the flexibility of the machine learning approach allows it to generalize better and detect anomalies in data without being overly dependent on the starting model. This capability of machine learning to work with random or less informative initial models aligns well with real-world seismic applications, where prior knowledge is often limited or uncertain.

The use of full waveforms brings us closer to handling more complex, real-world seismic datasets by incorporating comprehensive wavefield information. However, the applicabil-

ity of this approach to real-world data remains limited. A significant assumption in this work is that the source location is known a priori, which is often not the case in practice. This limitation could be addressed by jointly inverting for the source location along with the velocity parameters, providing a more flexible and realistic framework for seismic inversion.

Additionally, real-world seismic data often contains noise, which can severely impact inversion results. To make the approach more robust, improvements are needed to incorporate noise handling mechanisms, such as regularization techniques or data pre-processing steps, that can mitigate its effects. The challenge lies in developing models that can not only handle noisy data but also recover accurate velocity structures even when noise is present at varying levels.

Furthermore, the model’s ability to generalize to different geological settings with varying complexities should be explored. This could include heterogeneous media, anisotropic layers, or complex topography, all of which are commonly encountered in real-world seismic scenarios. Advancing the model to handle these factors would further improve its practical applicability and effectiveness in real-world seismic imaging and inversion tasks.

Through this study, we tested the applicability of ML in FWI. With promising results at hand, if further developed, an ML-based FWI could provide a computationally more efficient alternative to conventional techniques. Future work should focus on refining the model’s robustness, handling noise, improving generalization to different geological settings, and addressing practical challenges such as source localization. If successfully adapted, ML-based FWI could significantly transform the field of seismic inversion, providing both enhanced computational efficiency and the scalability required to handle more complex datasets.

Chapter 5

Conclusion

Seismic tomography inverts for velocity profile from travel-times. One commonly used approach is linear inversion, which involves solving a system of linear equations to determine the unknown velocity parameters. This method is well-regarded for its robustness and can yield accurate solutions if the system is well-conditioned. However, in many practical scenarios - including the focus of this study - the system may be ill-conditioned or underdetermined due to sparse or uneven ray coverage. Regularization techniques can help stabilize the inversion to produce meaningful solution, but what happens when even these measures fall short and linear inversion begins to fail?

Artificial neural network (ANN), a machine learning (ML) technique, is a data-driven approach that can leverage large datasets to learn patterns in the data and draw inferences. And for the problem under study, ANNs provide a powerful alternative to linear inversion, as first demonstrated by Komeazi et al. (2023) [9].

While Komeazi et al. implemented the approach for a vertical cross-section representation of a volcanic edifice, this study used a circular model, representing a horizontal cross-section. The implementation is explained in detail in Chapter 2. Section 2.4 begins with the results for well-resolved linear inversion cases, followed by cases where it appears to fail. The focus then shifts from the limitations of linear inversion to demonstrating the results of the ANN approach. The quantitative comparison of both approaches using SSIM, as presented in Figure 2.14, shows that the ANN approach achieves higher effectiveness. The results in the 3D domain also demonstrate the reliability of the ANN approach.

However, for seismic tomography modeling, we considered straight-ray-based travel-times, which neglect physical phenomena such as reflection, refraction, diffraction and others. To better address real-world dataset inversion, physics-based wave equation modeling is essential. We transition to full waveform modeling in Chapter 3. Forward modeling is done for elastic wave propagation using Salvus [10], a Python framework. While a simple fully-connected neural network was sufficient for seismic tomography, it is not adequate for velocity inversion using full waveforms. In this case, we no longer work with 1D travel time arrays, but with full waveforms, which are better represented as 2D arrays.

We implemented an autoencoder to check whether it is capable of reconstructing the waveforms from a lower dimensional representation, *i.e.* a latent space representation. It was observed that the autoencoder, explained in detail in Section 3.2.2, was indeed capable of reconstructing the waveform with the key characteristics - mainly travel-time

curves and amplitude. For example, an input waveform with 7,500 array elements was encoded into just 1,776 elements before being decoded back to its original form. This task was carried out with the motivation to see if encoded waveforms could be used for velocity inversion, rather than using full-sized waveforms, which might contain redundant information.

With good findings from the waveform reconstruction, as shown in Figure 3.9, we proceeded by using an encoder to encode the waveforms, followed by a fully connected stack to invert for the velocity parameters. One interesting thing to note here is that the waveforms could be compressed even further. Since our goal is not reconstruction but extracting key features for inversion, retaining the full spatial structure is unnecessary. but rather to use the important features for inversion. Consequently, a waveform that was previously compressed to 1,776 elements can be further reduced to just 432 elements for velocity inversion. This approach yields promising results as shown in Section 3.3. The results are compared with true models using SSIM. In case of homogeneous velocity profiles with a low-velocity anomaly, the ANN model is capable of detecting it.

While the ANN model performs well for the simplified setups considered in this study, its applicability to real-world data remains limited. Further improvements are needed to address more complex scenarios, such as the presence of noise. Additionally, a key assumption in this study is that the source location is known a priori, which may not be the case in practical applications. This limitation could be addressed by jointly inverting for both source location and velocity parameters, providing a more robust solution.

This study tested the applicability of ML-based methods in FWI, and the results show promising potential. With further development, ML-based FWI could offer a computationally efficient alternative to traditional methods, providing enhanced scalability and performance for handling complex datasets.

Bibliography

- [1] J.M. Lees. Seismic tomography of magmatic systems. *Journal of Volcanology and Geothermal Research*, 167:37–56, 2007.
- [2] C. Magee, C.T. Stevenson, S.K. Ebmeier, D. Keir, J.O. Hammond, J.H. Gottsmann, K.A. Whaler, N. Schofield, C.A. Jackson, M.S. Petronis, and B. O’Driscoll. Magma plumbing systems: A geophysical perspective. *Geological Society, London, Special Publications*, 457:53–82, 2018.
- [3] Guust Nolet. *A Breviary of Seismic Tomography: Imaging the Interior of the Earth and Sun*. Cambridge University Press, Cambridge, UK, 2008.
- [4] Jean Virieux and Stéphane Operto. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26, 2009.
- [5] D. Zandomenighi, R. Aster, P. Kyle, A. Barclay, J. Chaput, and H. Knox. Internal structure of erebus volcano, antarctica imaged by high-resolution active-source seismic tomography and coda interferometry. *Journal of Geophysical Research*, 118(12):6219–6233, 2013.
- [6] Z. Wang, H. Zhang, and T. Xu. Machine learning for seismic wave propagation modeling: A review. *Geophysical Journal International*, 221(3):1643–1659, 2020.
- [7] W. Feng, Y. Luo, Y. Zhang, and H. Zhang. Seismic wave propagation modeling using deep learning. *Geophysical Journal International*, 225(1):75–93, 2021.
- [8] S. Mostafa Mousavi and Gregory C. Beroza. Deep-learning seismology. *Science*, 377(6607):eabm4470, 2022.
- [9] Abolfazl Komeazi, Georg Rumpker, Johannes Faber, Fabian Limberger, and Nishtha Srivastava. When linear inversion fails: Neural-network optimization for sparse-ray travel-time tomography of a volcanic edifice. *Artificial Intelligence in Geosciences*, 5:100086, 08 2024.
- [10] Michael Afanasiev, Christian Boehm, Martin van Driel, Lion Krischer, Max Rietmann, Dave A May, Matthew G Knepley, and Andreas Fichtner. Modular and flexible spectral-element waveform modelling in two and three dimensions. *Geophysical Journal International*, 216(3):1675–1692, 11 2018.
- [11] R. C. Aster, B. Borchers, and C. H. Thurber. *Inverse Theory for Applied Geophysics*. Springer, 3rd edition, 2018.

-
- [12] L.R. Line and S. Treitel. A review of least-squares inversion and its application to geophysical problems. *Geophysical Prospecting*, 32(2):159–186, 1984.
 - [13] S. Haykin. *Neural Networks and Learning Machines*. Pearson, 3rd edition, 2008.
 - [14] IBM. Neural networks, 2025.
 - [15] Adam Paszke, Sam Gross, Francesco Massa, Robert Lerer, James Bradbury, Gautham Chanan, Tzu-Ming Killeen, Zeming Lin, Nikhil Gimelshein, Luca Antiga, Adrien Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Matthieu Raison, Amjad Tejani, Trishul Chilimbi, Jonathan Gray, and Kurt Keutzer. Pytorch: An imperative style, high-performance deep learning library, 2019.
 - [16] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
 - [17] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
 - [18] D. C. Liu and J. Nocedal. On the limited memory broyden-fletcher-goldfarb-shanno method. *Mathematical Programming*, 45(3):503–528, 1989.
 - [19] R. P. Hoblitt and B. Sturtevant. The 1980 eruption of mount st. helens: A summary of events. Technical Report USGS Professional Paper 1330, United States Geological Survey, 1987.
 - [20] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.
 - [21] T. Mitchell, A. Havas, and et al. Makie.jl: A high-performance, extendable plotting library for julia, 2021. Accessed: 2023-03-12.
 - [22] Fabian Limberger and Georg Rumpfker. Numerical modeling predicts seismic resonances in the magma chamber-conduit system due to wavefield capturing. *Volcanica*, 7(2):461–470, 07 2024.
 - [23] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
 - [24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
 - [25] PlotNeuralNet Contributors. Plotneuralnet: Neural network architecture visualization, 2021. Accessed: 2023-03-12.

Appendices

Appendix A

Computational Environments

A.1 Programming Languages

Python 3.10 - 3.12

Julia 1.11.1

Shell Scripting (Bash)

A.2 Major Libraries and Frameworks

Machine Learning: PyTorch (2.5.1)

Seismic Modeling: Salvus (2024.1.2)

Visualization: Matplotlib, Makie, PlotNeuralNet

A.3 Computational Resource

NVIDIA GeForce RTX 2080 Ti GPU

Appendix B

Forward Operator: $\mathbf{G}$

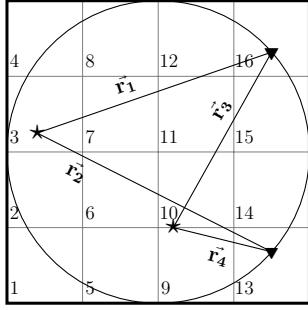


Figure B.1

As implemented before in Komeazi et al., in our modeling, $\mathbf{G}$ is the matrix of ray segments. Each $(i,j)^{th}$ element of $\mathbf{G}$ corresponds to the length of the i^{th} ray in the j^{th} block of the grid. $\mathbf{G}$ depends on the source-receiver and gridding settings. In order to understand the construction of $\mathbf{G}$, let's consider a very simple scenario as shown in Figure B.1. We have **4** rays and **16** model parameters in this case, implying $\mathbf{G}$ will be of shape (4,16). Now, ray $\vec{r}_1$ passes through blocks 3, 7, 11, 12, 16. Thus, only the elements $G_{1,3}$, $G_{1,7}$, $G_{1,11}$, $G_{1,12}$ and $G_{1,16}$ of the first row will be having non-zero values. If we consider the fourth row corresponding to ray $\vec{r}_4$, only the elements $G_{4,9}$ and $G_{4,13}$ will be having non-zero values. The entire $\mathbf{G}$ matrix in this case will be as given below.

$$\begin{pmatrix} 0 & 0 & G_{1,3} & 0 & 0 & 0 & G_{1,7} & 0 & 0 & 0 & G_{1,11} & G_{1,12} & 0 & 0 & 0 & G_{1,16} \\ 0 & G_{2,2} & G_{2,3} & 0 & 0 & G_{2,6} & 0 & 0 & G_{2,9} & G_{2,10} & 0 & 0 & G_{2,13} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G_{3,10} & G_{3,11} & 0 & 0 & 0 & G_{3,15} & G_{3,16} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G_{4,9} & 0 & 0 & 0 & G_{4,13} & 0 & 0 & 0 \end{pmatrix}$$

Notice that blocks 1, 4, 5, 8 and 14 are redundant since they have no ray coverage. Removing these blocks, *i.e.*, deleting the corresponding columns from $\mathbf{G}$ and elements from $\mathbf{m}$ can make our computations more efficient. This is especially true in the case of neural networks. Hence, the $\mathbf{G}_{\text{eff}}$ that is used for computations of this example case is

$$\begin{pmatrix} 0 & G_{1,3} & 0 & G_{1,7} & 0 & 0 & G_{1,11} & G_{1,12} & 0 & 0 & G_{1,16} \\ G_{2,2} & G_{2,3} & G_{2,6} & 0 & G_{2,9} & G_{2,10} & 0 & 0 & G_{2,13} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & G_{3,10} & G_{3,11} & 0 & 0 & G_{3,15} & G_{3,16} \\ 0 & 0 & 0 & 0 & G_{4,9} & 0 & 0 & 0 & G_{4,13} & 0 & 0 \end{pmatrix}$$

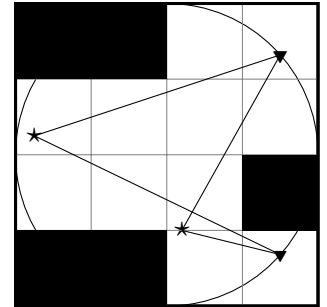


Figure B.2

Appendix C

Spectral Element Method

The equations of motion in an elastic medium can be solved by either a weak or a strong formulation, each suited for different numerical modeling approaches. The strong formulation employs the differential form of the equations with a point-wise discretization (*e.g.* finite differences), while the weak formulation relies on integral forms and a mesh-based discretization (*e.g.* finite elements).

The equations of motion in an elastic medium are given by:

$$\rho \ddot{\mathbf{u}} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f} \quad \text{in } \Omega, \quad (\text{C.1})$$

where ρ is the density, $\mathbf{u}$ is the displacement field, $\boldsymbol{\sigma}$ is the stress tensor, and $\mathbf{f}$ is the body force.

To obtain the weak formulation, we multiply by a test function $\mathbf{v}$ and integrate over the domain. The final equation is given by:

$$\int_{\Omega} \rho \ddot{\mathbf{u}} \cdot \mathbf{v} \, d\Omega + \int_{\Omega} \boldsymbol{\sigma} : \nabla \mathbf{v} \, d\Omega = \int_{\Omega} \mathbf{f} \cdot \mathbf{v} \, d\Omega + \int_{\partial\Omega} (\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot \mathbf{v} \, d\Gamma. \quad (\text{C.2})$$

where $\mathbf{n}$ is the outward unit normal on the boundary Γ .

The spectral element method (SEM) is a type of finite element method where the discretization is restricted to hexahedral volume elements (quadrilateral surfaces in 2D). This method combines the geometric flexibility of finite elements with the high accuracy of spectral methods, making it well-suited for seismic wave propagation simulations.

SEM achieves high-order accuracy using high-degree polynomial basis functions, such as Lagrange polynomials, defined at Gauss-Lobatto-Legendre quadrature points. These basis functions ensure efficient numerical integration while preserving spectral convergence. The equations of motion are reformulated in weak form by multiplying them with test functions and integrating over each element, naturally incorporating boundary conditions and enhancing numerical stability. This process results in element-wise mass and stiffness matrices that govern interactions between elements. Unlike finite difference methods, SEM produces a diagonal mass matrix, which significantly simplifies time integration. Finally, the local elemental equations are assembled into a global system that represents the entire computational domain.