Data-Driven and Theory-Guided Pseudo-Spectral Seismic Imaging Using Deep Neural Network Architectures

Christopher Zerafa

Supervised by Prof Pauline Galea

Co-supervised by Prof Cristiana Sebu

Department of Geosciences

Faculty of Science

University of Malta

September, 2021

A thesis submitted in partial fulfilment of the requirements for the degree of Ph.D. in Geosciences.

First edition, September 30, 2021

When something is important enough,
you do it even if the odds are not in your favor.

Elon Musk

Acknowledgements

Writing a significant scientific thesis is hard work and it would not possible without the support from various people. First of all, I wish to express my greatest appreciation towards my supervisor Prof Pauline Galea and co-supervisor Prof Cristiana Sebu for the intellectual guidance, valuable advice and help that was given to me during my research. The thesis would not have been written without their supervision and support.

My special appreciation to Dr. Godwin Debono who encouraged me to pursue this work and who's support have been invaluable.

I would like to thank Dr. Carlo Giunchi at Istituto Nazionale di Geofisica e Vulcanologia for providing the necessary computational resources and Dr. Elena Cuoco at EGO for the collaborations that we have had, and ones to be held.

I would like to express immense thanks to my wife Rachel, she has been extremely supportive of me throughout this entire process and has made countless sacrifices to help me get to this point.

Last but not least, gratitude goes to to my parents, Charles and Jane, my brother Daniel, and friends for their support and love. In more ways than one, I am here because of them.

Abstract

Full Waveform Inversion seeks to achieve a high-resolution model of the subsurface through the application of multi-variate optimization to the seismic inverse problem. Although now a mature technology, FWI has limitations related to the choice of the appropriate solver for the forward problem in challenging environments requiring complex assumptions, and very wide angle and multi-azimuth data necessary for full reconstruction are often not available.

Deep Learning techniques have emerged as excellent optimization frameworks. These sit on a spectrum between data and theory-guided methods. Data-driven methods impose no physical model of wave propagation and are not exposed to modelling errors. At the opposite end of the spectrum there are deterministic models governed by the laws of physics. In between, there are theory-guided methods which have some fixed parameters able mimic physical processes. This enables more intelligibility as compared to purely data driven approach.

Application of seismic FWI has recently started to be investigated within Deep Learning. This has focussed on the time-domain approach, while the pseudo-spectral domain has not been yet explored. However, classical FWI experienced major breakthroughs when pseudo-spectral approaches were employed. This thesis addresses the lacuna that exists in incorporating the pseudo-spectral approach within Deep Learning. This has been done by re-formulating the pseudo-spectral FWI problem as a Deep Learning algorithm for both a data-driven and a theory-guided pseudo-spectral approach. A deep neural network (DNN) and recurrent neural network (RNN) framework are derived. Either was formulated theoretically, qualitatively assessed on synthetic data, applied to a two-dimensional Marmousi dataset and evaluated against deterministic and time-based approaches.

Inversion of data-driven pseudo-spectral DNN was found to outperform classical FWI for deeper and over-thrust areas. This is due to the global approximator nature of the technique and hence not bound by forward-modelling physical constraints from ray-tracing. Pseudo-spectral theory-guided FWI using RNN was shown to be more accurate with only 0.05 error tolerance and 1.45% relative percentage error. Indeed, this provides more stable convergence, able to identify faults and has more low frequency content than classical FWI. From the comparative analysis of data-driven DNN and theory-guided RNN approaches, DNN was better performing, and recovered more of the velocity contrast, whilst RNN was better at edge definition. In general, RNN was more suited in shallow and deep sections due to cleaner receiver residuals.

Besides showing the improved performance of FWI formulated as a Deep Learning approach, this thesis highlighted the significant potential of such methods in other fields which have so far not been explored. New research avenues resulting from the shift in the inversion paradigm were identified and the next steps on how to continue developing these two frameworks presented.

Contents

List of Figures

List of Tables

List of Abbreviations

CDP	Common Depth Point	77
CNN	Convolutional Neural Network	16
CWT	Continuous Wavelet Transform	44
DNN	Deep Neural Network	4
FD	Finite Difference	125
FFT	Fast Fourier Transform	44
FWI	Full Waveform Inversion	2
LSTM	Long Short-Term Memory	18
NN	Neural Network	4
RNN	Recurrent Neural Network	16
RPE	Relative Percentage Error	70

1

Introduction

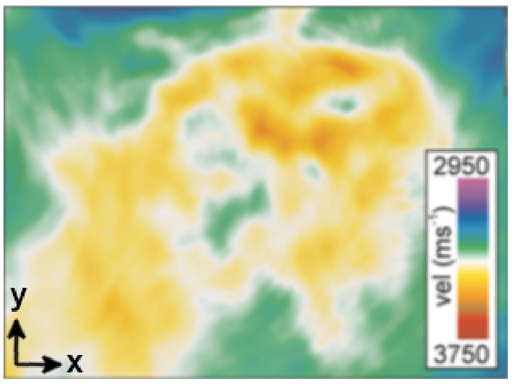
1.1 | Motivation

The seismic reflection method is by far the most widely used tool in geophysical exploration (Sheriff and Geldart, 1985). It uses artificially generated seismic waves that excite the earth and propagate through the subsurface. They are attenuated by interactions with their medium of propagation, and are partially reflected back and transmitted when coming across a high contrasting acoustic impedance. The reflected data are recorded by receivers (geophones or hydrophones) at or close to the surface. The time required for the waves to travel through the subsurface provides a measurement from which a subsurface model of acoustic media is determined. Geological significance is inferred from the data either directly from the seismic reflection method, or more commonly, through the integration of other methods such as gravitational, magnetics, refraction and other data sources such as well log data, vertical seismic profiles and geological settings of the region.

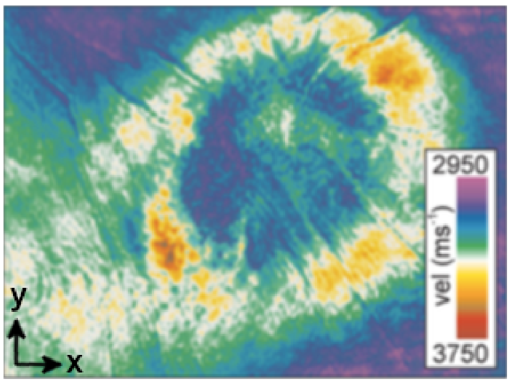
The search for new petroleum resources has pushed exploration areas of ever increasing complicated subsurface geology where the success of exploration wells depends heavily on the imaging of seismic data sets. To be able to image this geology, solvers have gradually moved from ray tracing algorithms to one-way wave equation methods and to acoustic and elastic two-way wave equation methods. These imaging techniques have until recently been largely limited to depth mapping of seismic reflections at boundaries between rock formations. The extraction of other information available in seismic data sets was compromised by prohibitive computing processing time and as a result only a fraction of the information contained in seismic datasets could be extracted.

The availability of super computers and Graphical Processing Units in recent years

has enabled seismic processing to implement previously prohibited imaging technology. One such technology is Full Waveform Inversion (FWI), which honours the physics of the wave equation for both phase and amplitude (Tarantola, 1987). The technique does not only invert for conventional depth imaging of boundaries but also rock properties such as velocity (v_p - compressional and v_s - shear), lithology and pore-fill thus providing a more comprehensive geology of the subsurface (Plessix et al., 2014). The technique has also gained popularity after it was demonstrated to produce spectacular improvements in imaging subsurface geology beneath a heterogeneous overburden - See Figure 1.1. This is a landmark in seismic exploration and has changed the way seismic data is used and interpreted.



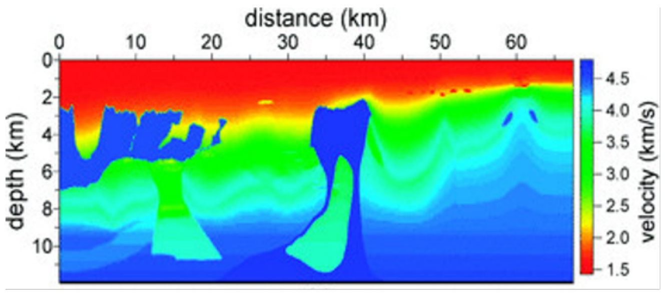
(a) Velocity model from conventional methods.



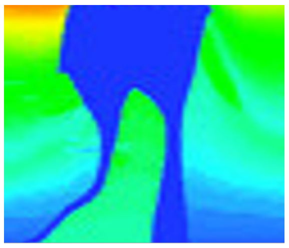
(b) <10Hz FWI velocity model result.

Figure 1.1: Horizontal slices though the Samson Dome in the Barent Sea at 1350m showing the uplift in imaging obtained through FWI. Axis indicated extend of the horizontal slice were not present in the original image from Morgan et al. (2013).

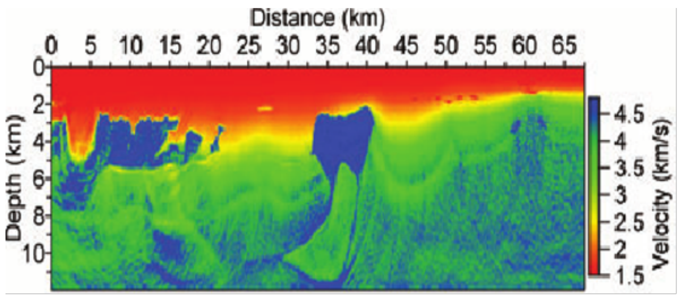
FWI seeks to achieve a high-resolution model of the subsurface through the application of multivariate optimization to the seismic inverse problem (Virieux and Operto, 2009). The inversion process begins with a best-guess initial model which is iteratively improved using a sequence of linearized local inversions to solve a fully non-linear problem. Figure 1.1 illustrates the image uplift which is achievable with FWI. In situations of more complex structures at depth with convoluted ray-paths in the overburden, the inversion becomes more difficult and more computationally expensive. Figure 1.2 illustrates an example of FWI on the 2004 BP synthetic data. The zoomed section (d) illustrates a lack of resolution of FWI.



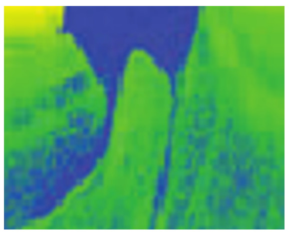
(a) 2004 BP synthetic for FWI.



(b) Zoom of complexity at depth.



(c) 2D FWI result.



(d) Zoomed section highlighting lack of resolution.

Figure 1.2: Limitations of FWI due to poor illumination. From Shin et al. (2010).

1.2 | Aims & Objectives

Optimization theory is fundamental to FWI since the parameters of the system under investigation are reconstructed from indirect observations that are subject to a forward modelling process (Tarantola, 2005). The accuracy of this forward problem depends on the validity of physical theory that links ground-truth to the measured data (Innanen, 2014). Moreover, solving for this inverse problem involves learning the inverse mapping from the measurements to the ground-truth which is based on a subset of degraded or best-estimate data (Tarantola, 2005; Tikhonov and Arsenin, 1977). Thus, two limitations within inverse theory can be identified: (i) the forward problem and (ii) the training data.

As evidenced throughout the historic development (reviewed in §2.1), choice of the forward problem will impact the accuracy of the FWI result. Challenging environments require more complex assumptions to try and better explain the physical link between data and observations, with not necessarily improved levels of accuracies (Morgan et al., 2013). Secondly, the data being used to reconstruct the mapping of measurements for the ground-truth are not optimal. Very wide angle and multi-azimuth data is required to enable full reconstruction of the inverse problem (Morgan et al., 2016); which is not always available. Furthermore, pre-conditioning of data is a necessity prior to FWI to make the inversion better posed (Kumar et al., 2012a; Mothi et al., 2013; Peng et al., 2018; Warner et al., 2013), however if done incorrectly this can degrade the inverse process (Lines, 2014).

Recently, deep learning techniques have emerged as excellent models and gained great popularity for their widespread success in pattern recognition tasks (Ciresan et al., 2011, 2012), speech recognition (Hinton et al., 2012) and computer vision (Deng and Yu, 2013; Krizhevsky et al., 2015). The use of deep neural networks to help solve inverse problems has been explored by Elshafiey (1991), Adler and Öktem (2017), Chang et al. (2017), Wei et al. (2017) and has achieved state-of-the-art performance in image reconstruction (Adler et al., 2017; Kelly et al., 2017; Petersen et al., 2017), super-resolution (Bruna et al., 2015; Galliani et al., 2017) and automatic-colourisation (Larsson et al., 2016). These deep learning based waveform inversion processes sit on a spectrum between strongly-data and strongly-theory guided methods. Data-driven methods impose no physical model of wave propagation and Neural Network (NN) weights are all trainable. These types of approaches require relatively exhaustive training datasets and tend to be less robust. Yet, due to the large number of degrees of freedom, they are not exposed to modelling errors as any conventional FWI algorithm (Li et al., 2019; Wu et al., 2018b). At the opposite end of the spectrum there are deterministic models, with bases in classical physics. In between, there are theory-guided or physics-informed methods which have some parameters fixed as non-trainable. If non-trainable parameters are built to mimic a physical process, the training model space now consists of only the parameters for that process and the number shrinks drastically. This provides robust training and enables more intelligibility as compared to purely data driven approach (Biswas et al., 2019).

As with most techniques, after decades of development, FWI algorithms now only being improved incrementally, with slight modification to the underlying algorithms. The technique hence requires a fresh injection of ideas and academic pursuits to elevate it to the next phase of development. Application of seismic FWI has recently started to be investigated within the deep learning field and has so far been focused only on the time-domain approach. However, classical FWI experienced pivotal breakthroughs via pseudo-spectral approaches which enabled the technique to go beyond academic experiments and be employed on real datasets (See § 2.1.3). The main aim of the research work presented in this thesis was to investigate whether the same advantages apply when pseudo-spectral FWI is developed within DNN. To current knowledge, there is no prior work investigating the pseudo-spectral inversion within Deep Neural Network (DNN) frameworks. Specifically, my study was motivated by the following research questions: Are pseudo-spectral approaches possible within a DNN framework? Is the shift to a data-driven inversion detrimental? How good can theory-guided inversion be? How do pseudo-spectral DNN compare to deterministic conventional FWI? Are there any particular benefits of data-driven or physics-guided pseudo-spectral DNN approaches? How do pseudo-spectral DNN compare to time-based DNN approaches? Are there any cross-over techniques between Deep Learning and geophysics which would benefit either field? Could this cause the new wave of evolution of the FWI framework?

Hence, the main aim of this work was to develop these two novel approaches in the form of data-driven and theory-guided pseudo-spectral FWI, compare them to traditional approaches and investigate their limitations. To this end, the following steps were followed:

1. Building a comprehensive literature review of DNN applications within geophysics. This will be focussed particularly on pseudo-spectral FWI approaches and highlights key examples for data-driven and theory-guided approaches.

2. Re-casting FWI within a DNN framework for both a data-driven direct learned inversion and a theory-guided temporal based DNN formulation. Both approaches were derived theoretically and assessed on synthetic data. The assessment built up from simple 1D experiments, extended to 2D and evaluated on the standard Marmousi model. The results were validated against classical FWI.

3. Analysing the limitations of both approaches and discussing future potential developments.

1.3 | Document Structure

This dissertation is composed of five chapters, each of them dealing with different aspects of a pseudo-spectral FWI within a Deep Learning framework.

- Chapter 1 is introductory and deals with introducing the topic and sets the aims and objectives for this work.

- Chapter 2 gives a literature review of work which highlights trends in FWI, Deep Learning and their combined application respectively.

- Chapter 3 is sub-divided into 3 parts and provides all relevant theory for FWI via DNNs. Part 1 illustrates the key elements within classical FWI. Part 2 presents

FWI as a data-driven DNN and Part 3 derives RNN as a physics informed framework for FWI. Each of these is compared to classical FWI, with commonalities and differences highlighted.

- Chapter 4 builds on the derived theory and evaluates each framework on synthetic data.
- Chapter 5 critically analyses the numerical results obtained in Chapter 4 and outlines their limitations. Future work is presented and conclusions addressed.

2

Literature Review

In this chapter the main literature concerning FWI is discussed in the context of pseudo-spectral approaches, with focus on pointing out the general research trends in this area. NNs are then presented with respect to inverse problems, and their development history discussed. Both these components are put together to identify applications within geophysics, with emphasis on velocity inversion approaches.

2.1 | Full Waveform Inversion

FWI tries to derive the best velocity model and other lithologic properties (as density, anelastic absorption and anisotropy) of the Earth's subsurface to be consistent with recorded data. An exhaustive search for this ideal model is almost impossible and methods for finding an optimal one describing the data space are necessary. There are two main categories for dealing with this problem: (i) global optimization methods, and (ii) direct solving through linearisation.

Global optimization methods try to find global minimum of the misfit function. These are stochastic in nature and use global information about the misfit surface to perform model updates (Sen and Stoffa, 1995; Törn and Žilinskas, 1989). Three most well-known cases of global methods are Monte Carlo methods (Biswas and Sen, 2017; Press, 1968), genetic algorithm (Gerstoft, 1994; Parker, 1999; Tran and Hiltunen, 2012) and simulated annealing (Pullammanappallil and Louie, 1994; Rothman, 1985; Tran and Hiltunen, 2011). Global optimization methods are all very dependent on a fast forward modelling algorithm as they require large amounts of forward modelling calculations. As computers are getting faster and better, the necessity of keeping the parametrization simple might decline. However, current solutions to the seismic inverse problem have to resort to local optimization. The next section reviews direct solving through linearisation for FWI.

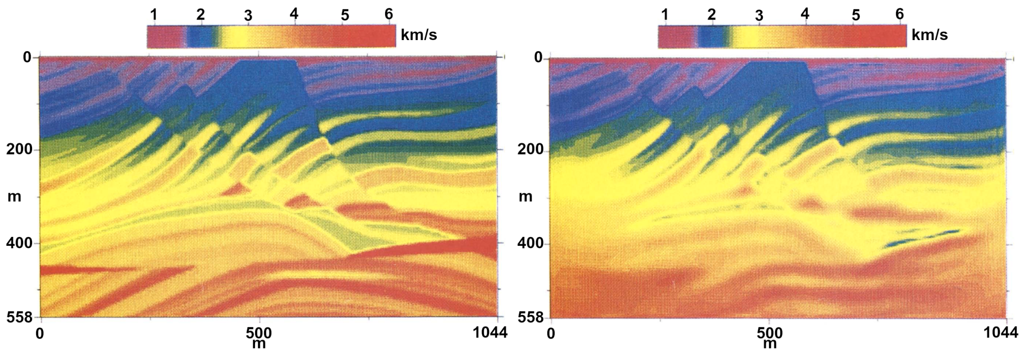
2.1.1 | Formulation of FWI as Local Optimization

The concept of local optimization FWI was introduced in the 1980's. Lailly and Bednar (1983) and Tarantola (1984a) cast the exploding-reflector concept of Claerbout (1971, 1976) as a local optimization problem which aims to minimise in a least-squares sense the misfit between recorded and modelled data. The problem is set in the time-domain as follows: set a forward propagation field to model the observed data, back propagate the residuals between the modelled and observed data, cross-correlate both fields at each point in space to derive a correction, and do least squares minimization of the residuals iteratively. This outline forms the basis of this technique to this day.

Gauthier et al. (1986) numerically demonstrate a local optimization FWI approach using two-dimensional synthetic data examples. A single diffracting point on a homogeneous model was used to illustrate the importance of proper sampling of the subsurface. Furthermore, this model was used to show that the free surface adds an extra complexity to the problem and increases the non-linearity of the inversion. FWI with or without free-surface multiple modelling is an active area of research to this day (Bergen et al., 2019; Komatitsch and Tromp, 2002).

2.1.2 | Applications in Time and Frequency

One of the pioneering applications of FWI was presented by Bunks et al. (1995) and is represented in Figure 2.1. They showed better imaging using a hierarchical multi-scale approach on the Marmousi synthetic model. This strategy initially inverts for low-frequency components where there are fewer local minima and those that remain are further apart from each other than for higher frequencies. However, decomposing by scale did not resolve issues of source estimation, source bandwidth and noise. The frequency-domain approach was proposed in the 1990s by Pratt and collaborators (Pratt, 1990; Pratt and Goulty, 1991; Pratt and Worthington, 1990). The first application was to cross-hole data utilizing a finite difference approach and an elastic wave propagator to facilitate the modelling of multi-source data. This was extended to wide-aperture seismic data by Pratt et al. (1996). Analytically, the time- and frequency-domain problems are equivalent, see Virieux and Operto (2009). Early examples of pseudo-spectral FWI include application to the Marmousi model (Sirgue and Pratt, 2004) and a wide aperture land seismic dataset by (Operto et al., 2004).



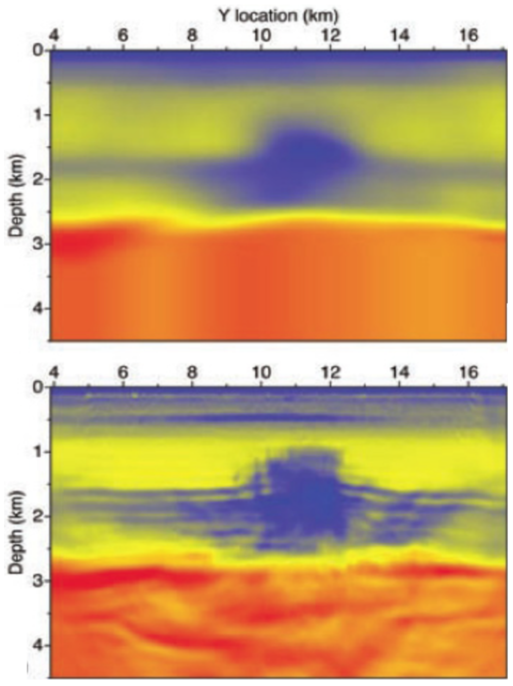
(a) Section from real Marmousi velocity model. (b) Best estimate using multiscale FWI.

Figure 2.1: First practical application of FWI using the Marmousi model. This shows significant improvements for the FWI results as presented by Bunks et al. (1995).

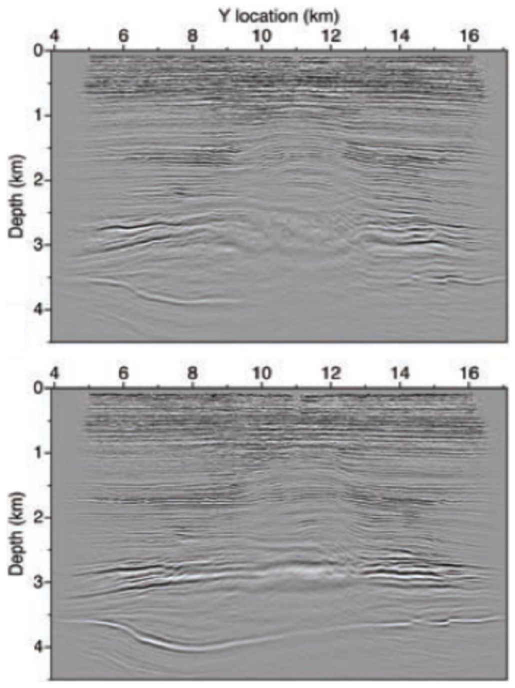
2.1.3 | Beyond Academic Experiments

Two-dimensional inversion was able to explain out-of-plane events by mapping them into in-plane artefacts (Morgan et al., 2009). The technique was restricted to purely academic pursuits (Sirgue et al., 2009) and full potential could only be realized if extended to 3D. The first 3D frequency-domain algorithms where developed by Warner et al. (2007) on synthetic datasets, however these used low initial frequencies that are not normally present in real data (Morgan et al., 2013). Examples of this application are demonstrated by Sirgue et al. (2007), Ali et al. (2007) and Operto et al. (2007). Warner et al. (2008) presented the first 3D real data application to a shallow North Sea survey. This improved the resolution of shallow high-velocity channels that resulted in uplifts upon migration. In Figure 2.2, Sirgue et al. (2009) demonstrated successful FWI results for a 3D dataset of the Valhall field, Norway. They inverted wide-azimuth ocean-bottom cable data using a sequence of low frequency bands to generate high-resolution velocity models. The updated velocity model demonstrated a network of shallow high-velocity channels and a gas-filled fracture extension from a gas cloud which was not previously identifiable (Sirgue et al., 2010).

Plessix and Perkins (2010) show results from the application of full waveform inversion to an ocean bottom seismometer dataset recorded in the Gulf of Mexico with near-ideal long-offset and wide-azimuth. Their approach was anisotropic and assumed vertical transversely isotropic media with fixed Thomsen's parameters. The model had better imaging of dips and produced flatter common image gathers in the deep part of the model (Plessix and Perkins, 2010). Wang and Tsvankin (2016) developed 3D waveform inversion for orthorhombic media in the acoustic approximation using pseudo-spectral



(a) Velocitiy model. Top: Conventional, Bottom: FWI.



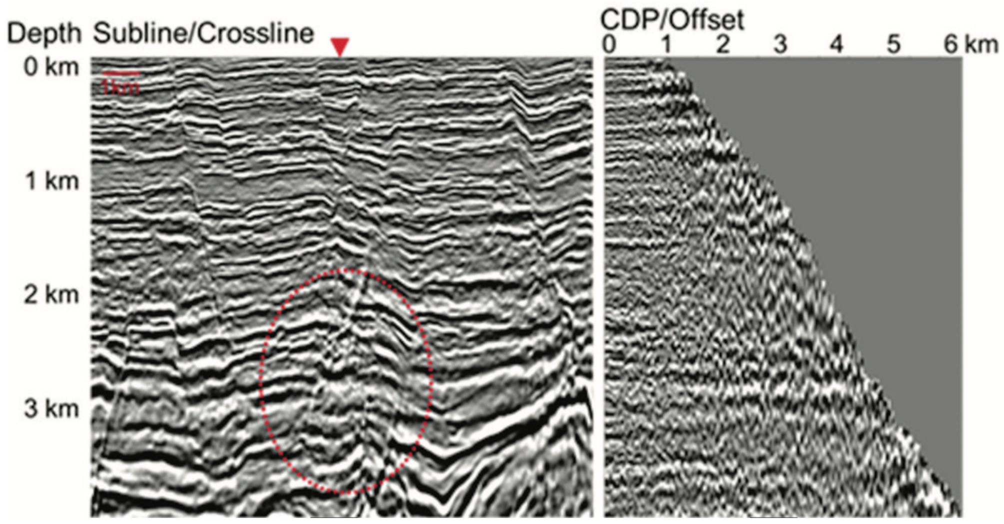
(b) Pre-stack depth migrated section. Top: Conventional, Bottom: FWI.

Figure 2.2: Improvements in velocity model and pre-stack depth migrated images obtained through FWI over the Valhall field. The FWI updated velocity model demonstrated a network of shallow high-velocity channels and a gas-filled fracture extension from a gas cloud which was not previously identifiable in conventional tomography. The impact is evident in the migrated sections, which show more continues events in otherwise poorly illuminated area. Adapted from Sirgue et al. (2009) and Sirgue et al. (2010).

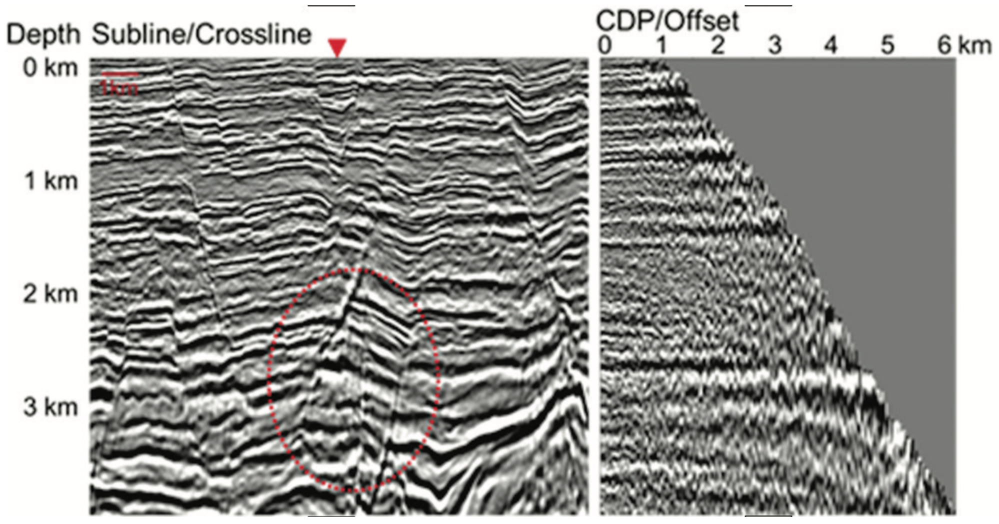
methods. This was found to be stable and produced kinematic accurate pure-mode primary wavefields with an acceptable computational cost. Xie et al. (2017) applied orthorhombic full-waveform inversion for imaging wide-azimuth ocean-bottom-cable data. The results had better azimuthal and polar direction-dependent wave imaging which significantly improved fault imaging - See Figure 2.3.

A re-occurring theme within this section is the creation of a better approximation to the wavefield propagation within the subsurface; 1D to 2D to 3D discretization, acoustic to anisotropic to elastic to orthorhombic wavefield modelling, with each additional dimension of information resulting in more numerical and computer intensive algorithms (Kumar et al., 2012b). Even though computing power has increased dramatically, making FWI more productive, the underlying algorithms are only improving incrementally.

Indeed, the next generation of experiments will require changes to acquisition geometry to allow for full-bandwidth and multi-azimuth reconstruction of the wavefield (Morgan et al., 2016).



(a) PSDM stack and CDP gather with original orthorhombic model from tomography.



(b) PSDM stack and CDP gather with orthorhombi FWI update.

Figure 2.3: Imaging improvements obtained through orthorhombic imaging. This produces sharp truncations and clearer faults as highlighted by the red dashed ovals, as well as better focussed gathers. Adapted from Xie et al. (2017).

2.2 | Deep Neural Networks

2.2.1 | Neural Networks for Inverse Problems

The mathematical formulation of FWI falls under the more general class of variational inverse problems (Tanaka, 2003). The aim is to find a function which is the minimal or the maximal value of a specified functional (Dadvand et al., 2006). Indeed, inverse problems attempt to reconstruct an image $x \in X \subseteq \mathbb{R}^d$ from a set of measurements $y \in Y \subseteq \mathbb{R}^m$ of the form

$$y = \Gamma(x) + \epsilon \tag{2.1}$$

where $\Gamma : X \mapsto Y, \Gamma \in \mathbb{R}^{m \times d}$ is the discrete operator and $\epsilon \in Y \subseteq \mathbb{R}^m$ is the noise. NN within Machine Learning can be considered to be a set of algorithms of non-linear functional approximations under weak assumptions (Öktem and Adler, 2018). Namely, when applied to inverse problems, Equation 2.1 can be re-phrased as the problem of reconstructing a non-linear mapping $\Gamma_\theta^\dagger : Y \mapsto X$ satisfying the pseudo-inverse property

$$\Gamma_\theta^{-1}(y) \approx x \tag{2.2}$$

where observations $y \in Y$ are related to $x \in X$ as in Equation 2.1, and θ represents the parametrization of pseudo-inverse by the NN learning (Adler and Öktem, 2017). The loss function defined in Equation 2.2 is dependent on the type of training data, which is dependent on the learning approach (Adler et al., 2017). There are two main classes of learning in Machine Learning: (i) Supervised, and (ii) Unsupervised.

In supervised learning, training data are independent distributed random pairs with input $x \in X$ and labelled output $y \in Y$ (Vito et al., 2005). Estimating θ for Equation 2.2 can be formulated as minimizing a loss function $\mathcal{J}(\theta)$ which has the following structure (Adler and Öktem, 2017):

$$\mathcal{J}(\theta) := \mathcal{D}(\Gamma_\theta^{-1}(\boldsymbol{y}), \boldsymbol{x}) \tag{2.3}$$

where $\mathcal{D}$ is a distance function quantifying the quality of the reconstruction and $\Gamma_\theta^{-1} : Y \mapsto X$ is the pseudo-inverse to be learned (Adler and Öktem, 2017). A common metric for the distance function is the sum of squared distances, resulting in:

$$\mathcal{J}(\theta) := \left|\left|\Gamma_\theta^{-1}(y) - x\right|\right|_X^2 \tag{2.4}$$

Approaching the inverse problem directly via this approach amounts to learn $\Gamma_\theta^{-1} : Y \mapsto X$ from data such that it approximates an inverse of Γ. In particular, this has successful applications in medical imaging (Lucas et al., 2018; Xu et al., 2012), signal processing

(Dokmanić et al., 2016; Rusu and Thompson, 2017) and regularization theory (De los Reyes et al., 2017; Meinhardt et al., 2017; Romano et al., 2016).

In unsupervised learning, there exist no input-output labelled pairs and the training data is solely elements of $y \in Y$. The NN is required to learn both the forward problem and inverse problem (Andrychowicz et al., 2016). The loss functional for unsupervised learning is given as:

$$\mathcal{J}(\theta) := \mathcal{L}\left(\Gamma\left(\Gamma_\theta^{-1}(y)\right), x\right) + \mathcal{S}\left(\Gamma_\theta^{-1}(g)\right) \tag{2.5}$$

where $\mathcal{L} : Y \times X \mapsto \mathbb{R}$ is a suitable affine transformation of the data and $\mathcal{S} : X \mapsto \mathbb{R}$ is the regularization function. Main applications of this learning are to inherent structure and have been proven successful in exploratory data analysis applications such as clustering (Gerdova et al., 2002; Sever, 2015) and dimension reduction (Dolenko et al., 2015).

2.2.2 | Evolution of Neural Networks

The remaining literature review is restricted to supervised learning approaches using NN as these are more suited for velocity inversion. For a complete review, Lippmann (1987) and Chentouf (1997) provide further detail.

2.2.2.1 | Early Neural Nets and the Perceptron

The basic ideas of NN date back to the 1940's and were initially devised by McCulloch and Pitts (1943) when trying to understand how to map the inner workings of a biological brain into a machine. From a biological aspect, neurons in the brain are interconnected via nerve cells that are involved in the processing and transmitting of chemical and electrical signals (McCulloch and Pitts, 1943).

Early NNs with rudimentary architectures did not learn (McCulloch and Pitts, 1943) and the notion of self-organized learning only came about in 1949 by Hebb (1949). Rosenblatt (1958) extended this idea of learning and proposed the first and simplest neural network – the McCullock-Pitts-Perceptron. As shown in Figure 2.4, this consists of a single neuron with weights and an activation function. The weights are the learned component and determine the contribution of either input x to the output y. The activation function σ adds a non-linear transform, allowing the neuron to decide if the input is relevant for the paired output. Without an activation function, the neuron would be equivalent to a linear regressor (Minsky and Papert, 2017). Rosenblatt (1958) used this fundamental architecture to reproduce a functional mapping that classifies patterns that are linearly separable. This machine was an analogue computer that was connected to a camera that used 20×20 array of cadmium sulphide photocells to produce a 400-pixel image. Shown in Figure 2.5, the McCullock-Pitts-Perceptron had a patch-board that allowed experimentation with different combinations of input features wired up randomly to demonstrate the ability of the perceptron to learn (Bishop, 2006; Hecht-Nielsen, 1990).

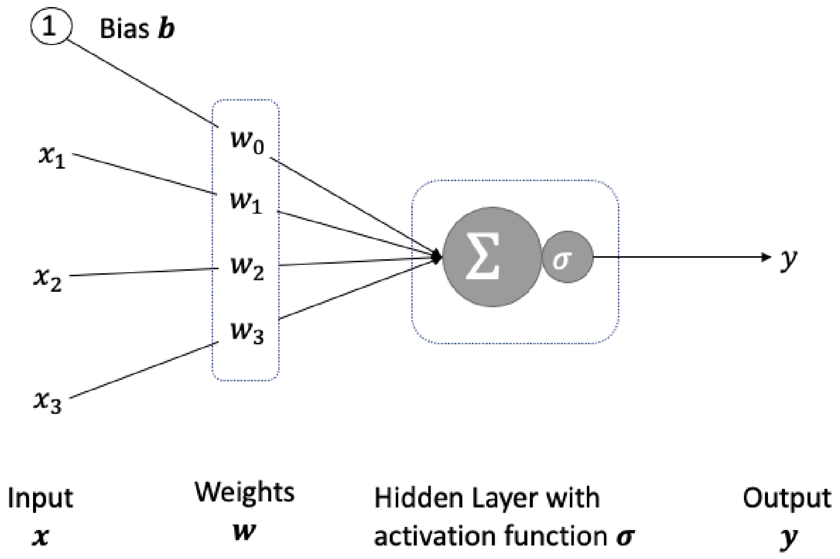


Figure 2.4: The Single Neuron Perceptron. The input values are multiplied by the weights. If the weighted sum of the product satisfies the activation function, the perceptron is activated and "fires" a signal. Adapted from Rosenblatt (1958).

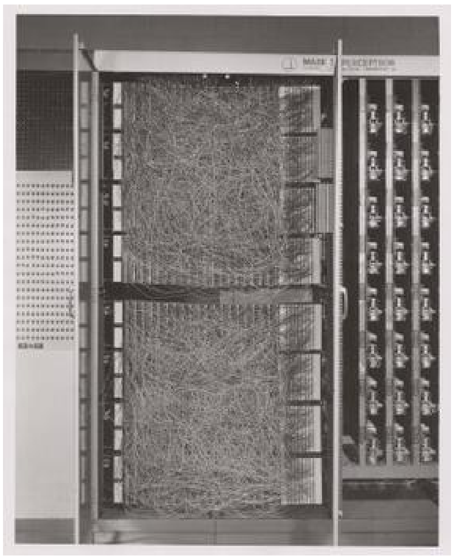


Figure 2.5: The Mark I Perceptron Machine was the first machine used to implement the Perceptron algorithm. The machine was connected to a camera that used 20×20 array of cadmium sulphide photocells to produce a 400-pixel image. To the right is a patch-board that allowed experimentation with different combinations of input features. This was usually wired up randomly to demonstrate the ability of the perceptron to learn. Adapted from Bishop (2006); Hecht-Nielsen (1990).

Rosenblatt's perceptron was the first application of supervised learning (Russell and Norvig, 2008). However, Minsky and Papert (2017) highlight limitations to the applications of a single perceptron and that Rosenblatt's claims that the "perceptron may eventually be able to learn, make decisions, and translate languages" were exaggerated. Following the publication of the book, research on perceptron-style learning machines practically halted (Minsky and Papert, 2017).

2.2.2.2 | Back-Propagation and Hidden Layers

Efficient error back-propagation in arbitrary, sparsely connected, NN networks were described in Linnainmaa's master thesis (Linnainmaa, 1970). This minimizes the errors through gradient descent in the parameter space (Hadamard, 1907) and allows for explicit minimization of the cost function. Back-propagation permits NNs to learn complicated multidimensional functional mappings (Dreyfus, 1973).

The back-propagation formulation lends itself from major developments in dynamic programming throughout the 1960s and 1970s (Bryson, 1961; Kelley, 1960; Linnainmaa, 1976). A simplified derivation using the chain rule was derived by Dreyfus (1973) and the first NN-specific application was described by Werbos (1981). It was until the mid-1980s that Rumelhart et al. (1986) made back-propagation mainstream for NNs through the numerical demonstration of internal representations of the hidden layer. Hidden layers reside in-between input and output layers of the NN.

Back-propagation was no panacea and additional hidden layers did not offer empirical improvements (Schmidhuber, 2015). Kolmogoro and Tikhomirov (1956), Hecht-Nielsen (1989) and Hornik et al. (1989) pursued development of back-propagation encouraged by the Universal Approximation Theorem. Namely, this theorem states that if enough hidden units are used in a NN layer, this can approximate any multivariate continuous function with arbitrary accuracy (Hecht-Nielsen, 1989). Although back-propagation theoretically allows for deep problems, it was shown to work on practical problems (Schmidhuber, 2015).

2.2.2.3 | The Vanishing Gradient and Renaissance of Machine Learning

The major milestone in NN came about in 1991. Hochreiter's thesis identified that deep NNs suffer from the vanishing or exploding gradient problem (Hochreiter, 1991). Gradients computed by back-propagation become very small or very large with added layers, causing convergence to halt or introduce unstable update steps. Solutions proposed to address this challenge included batch normalization (Ioffe and Szegedy, 2015), Hessian-free optimisations (Martens, 2010; Møller, 1993; Schraudolph, 2002), random weight assignment (Hochreiter and Schmidhuber, 1996), universal sequential search (Levin, 1973) and weight pruning (LeCun et al., 1990).

Prior to 2012, NN were apparently an academic pursuit. This changed when AlexNet (Krizhevsky et al., 2012) won the ImageNet (Russakovsky et al., 2015) visual object recognition by a considerable margin. AlexNet used a deep architecture consisting of eight layers (Krizhevsky et al., 2015) and was the only entry employing NN in 2012. All submissions in subsequent years were NN-based (Singh, 2015) and in 2015, NNs surpassed human performance in visual object recognition for the first time (Russakovsky et al., 2015) - see Figure 2.6. AlexNet is undoubtedly a pivotal event that ignited the renaissance in interest around deep learning.

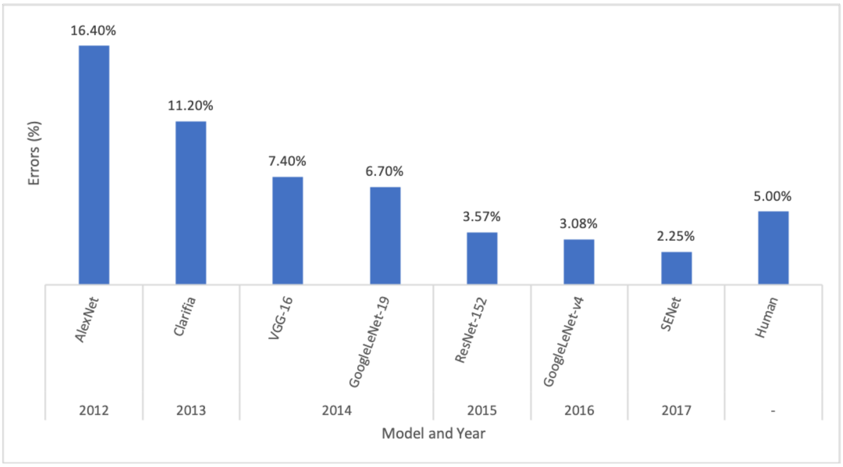


Figure 2.6: Evolution of the accuracy for the ImageNet challenge (Krizhevsky et al., 2012). AlexNet won ImageNet in 2012 with 16.4% error in accuracy. With each year of the competition, the accuracy has been increasing. Sources for these accuracies are Clarifia (Zeiler and Fergus, 2014), VGG-16 (Simonyan et al., 2014), GoogleLeNet-19 (Szegedy et al., 2014), ResNet-152 (He et al., 2016b), GoogleLeNet-v4 (Szegedy et al., 2016) and SENet (Hu et al., 2017).

2.2.3 | Deep Neural Network Architecture Landscape

According to Patterson and Gibson (2017), three of the most common major architectures are (i) neural network, (ii) Convolutional Neural Network (CNN), and (iii) Recurrent Neural Network (RNN).

As introduced and discussed in § 2.2.2.1, ANNs are non-linear models inspired by the neural architecture of the brain in biological systems. A typical neural network is known as a multi-layer perceptron and consists of a series of layers, composed of neurons and their connections (Goodfellow et al., 2016).

CNNs are regularized version of MLPs with convolution operations in place of general matrix multiplication in at least one of the layers (LeCun, 1989). These types of networks find their motivation from work by Hubel and Wiesel in the 1950s and 1960s (Hubel and Wiesel, 1959, 1962). Inspired by this work, Fukushima and Miyake (1982) introduced the two basic types of layers in CNNs: convolutional layers and downsampling layers. Zhou & Chellappa introduced the concept of max pooling (Zhou and Chellappa, 1988) and LeCun et al. (1990) utilized back-propagation to learn the convolution kernel coefficients directly from images of hand-written numbers. The architecture of the NN used is know as LeNet5 and is shown in Figure 2.7. This essentially laid the foundations for modern CNNs. LeCun et al. (2010) gives a comprehensive history up to 2010 and a more recent review is available by Khan et al. (2020).

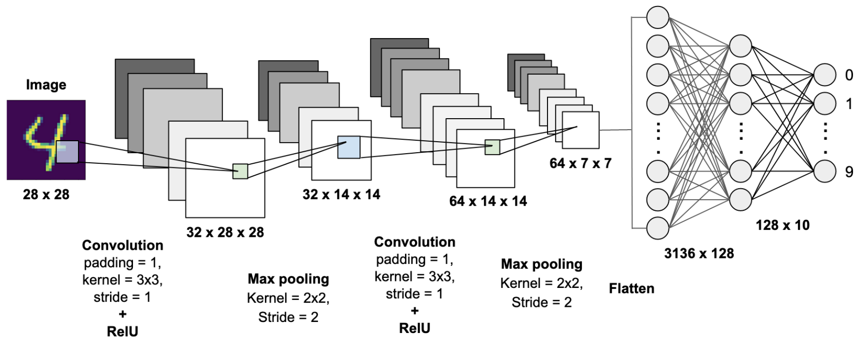


Figure 2.7: LeNet5 - LeCun et al.'s (1990) CNN architecture used to classify handwritten digits. LeNet-5 architecture consists of two sets of convolutional and max pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a soft-max classifier.

RNNs are in the family of feed-forward neural networks that add the concept of recurrent connections or parameter sharing (Lang and Hinton, 1988). At each time-step of sending input through a RNN, nodes receiving input along recurrent edges receive input activations from the current input vector and from the hidden nodes in the network's previous state (Waibel et al., 1989). The output is computed from the hidden state at the given time-step. The previous input vector at the previous time step can influence the current output at the current time-step through the recurrent connections (Lang et al., 1990). This is known as unrolling a RNN and is shown in Figure 2.8.

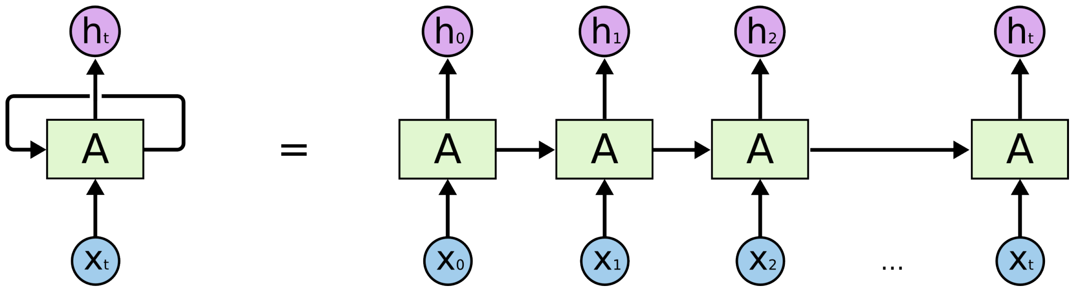


Figure 2.8: An unrolled RNN showing the recurrent connections between RNN cells A for some input x_t and outputs value h_t. From Olah (2015).

Long Short-Term Memory (LSTM) networks are the most commonly used variation of RNNs. LSTM networks were introduced by Hochreiter and Schmidhuber (1997). The critical component of the LSTM is the addition of memory gates or cells (Gers et al., 1999; Graves, 2012). The contents of the memory cell are modulated by input and forget gates. Assuming that both of these gates are closed, the contents of the memory cell will remain unmodified between one time-step and the next. The gating structure allows information to be retained across many time-steps, and consequently also allows gradients to flow across many time-steps. This allows the LSTM model to overcome the vanishing gradient problem that occurs with most RNN models.

2.2.4 | Not Just Algorithms

Apart from Machine Learning algorithms, re-interest in DNN has led to software architectures that allow for quick development. The most common include Tensorflow (Abadi et al., 2015), Keras (Chollet, 2015), PyTorch (Paszke et al., 2017), Caffe (Jia et al., 2014)) and Deeplearning4j (Nicholson and Gibson, 2016). These types of frameworks are facilitating interdisciplinarity between Machine Learning and geophysics. Indeed, Richardson (2018) employed DNN architecture within Tensorflow to solve for FWI. Utilizing a common DNN optimizer - Adam - it was shown how the cost function converged quicker in the inversion process as compared to conventional methods in FWI (Sharma et al., 2017). Richardson (2018) show by comparing the loss function as shown in Figure 2.9.

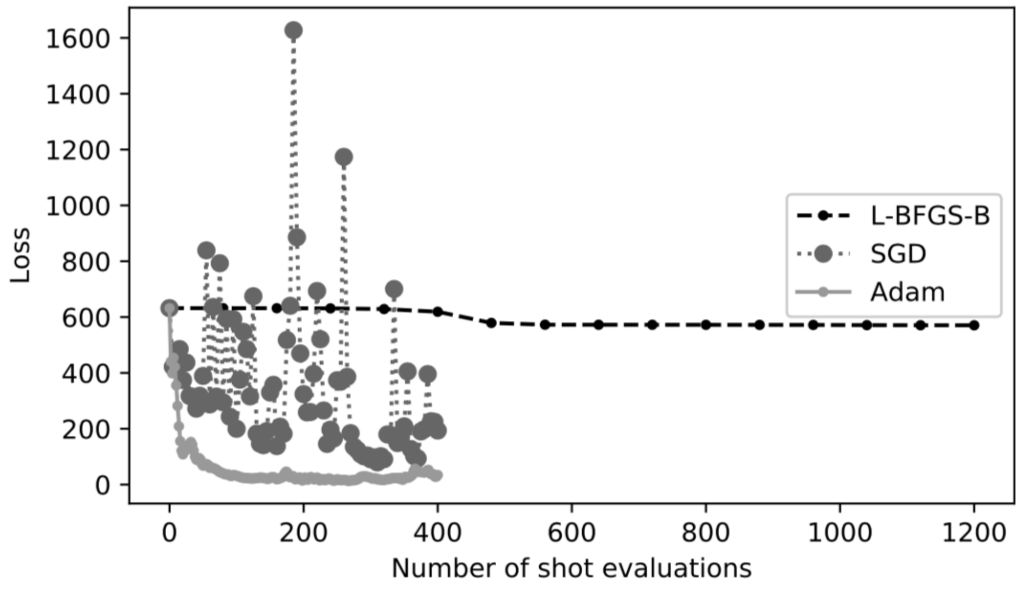


Figure 2.9: State-of-the-art Adam cost function calculation converged much more rapidly than conventional Stochastic Gradient Descent and L-BFGS-B. From Richardson (2018).

2.3 | Neural Networks in Geophysics

Machine Learning techniques have been utilised across different geophysical applications. Some notable examples include geo-dynamics (Shahnas et al., 2018), geology (Reading et al., 2015), seismology (Shimshoni and Intrator, 1998), paleo-climatology (Dowla and Rogers, 1996), climate change (Anderson and Lucas, 2018) and hydrogeology (Hamshaw et al., 2018). Unsupervised algorithms have also been investigated by Köhler et al. (2010) for pattern recognitions of wavefield patterns with minimal domain knowledge. Other geophysical application include seismic deconvolution (Calde On-Macías et al., 1997; Wang and Mendel, 1992), tomography (Nath et al., 1999), waveform recognition and first-break picking (Murat and Rudman, 1992), trace editing (McCormack et al., 1993), electricity (Poulton et al., 1992), magnetism (Zhang and Paulson, 1997), shear-wave splitting (Dai and MacBeth, 1994), event classification (Romeo, 1994), petrophysics (Downton and Hampson, 2018) and noise attenuation (Halpert, 2018; Li et al., 2018b).

2.3.1 | Legacy Velocity Inversion

More specific to velocity estimation, the first published investigation for the use of NN was a RNN by Michaels and Smith (1992). Their network architecture represented all components in an elastic FWI experiment with a seismic source, the propagation media and an imaging response. Figure 2.10 shows a block diagram representation for their network. The neural column consisted of two 1-layer neuron columns, one for particle displacement and another for particle velocity.

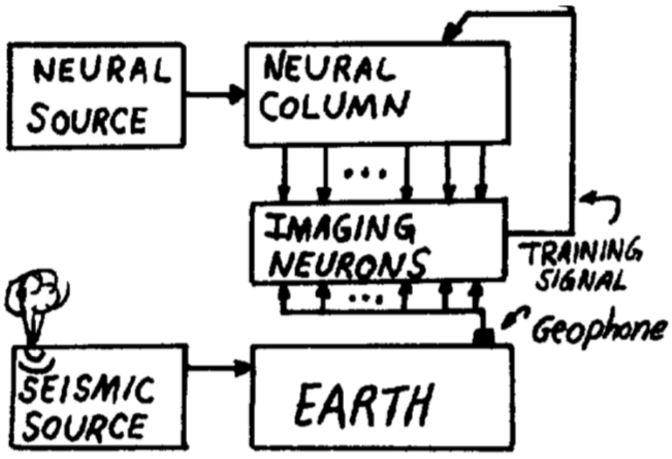


Figure 2.10: Block diagram for RNN system by Michaels and Smith (1992). The difference between a signal and the internal neural signals along a neural column are processed to provide a training signal that modifies neuron weights.

Röth and Tarantola (1994) published the first application of NN which estimated 1D velocity functions from shot gathers from a single layer NN in 1994. Figure 2.11 shows their NN architecture. This accepted synthetic common shot gathers from a single source as input and used to compute corresponding 1D large-scale velocity models. The training set used for learning consisted of 450 synthetic models built up of eight strata with constant layer thickness over a homogeneous half-space. Their network was able to approximate a true velocity model sufficiently to act as a starting model for further seismic imaging algorithms. The inferred velocity profiles of the unseen data provided 80% accuracy levels, and although the network was stable for noise contained data, it was not robust against strong correlated noise. Nonetheless, this investigation sets up NN as possible candidates to solve non-trivial inverse problems.

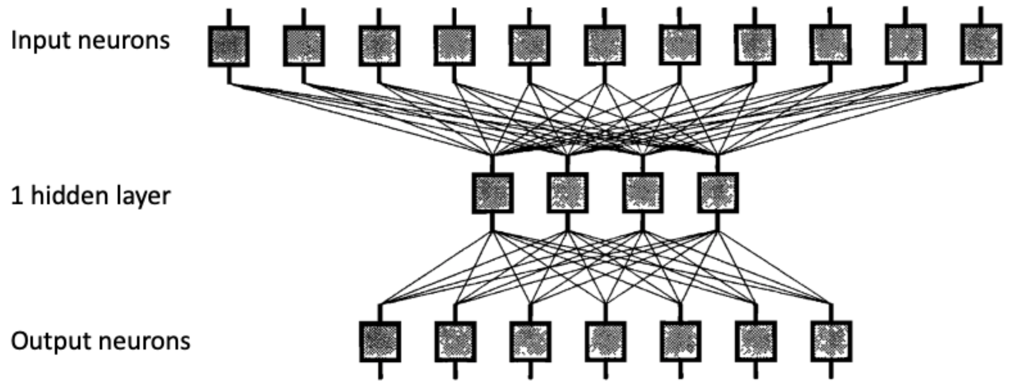


Figure 2.11: Architecture for first NN application to FWI. This was a very shallow NN with 1 hidden layer and non-symmetric input-output neurons. Adapted from Röth and Tarantola (1994).

NNs are not solely limited to creating initial models to FWI. Langer et al. (1996) show how this can invert for the governing parameters related to the seismic source and propagation medium. Given the correct network topology and a large enough training sample for the learning of the inversion, Langer et al. successfully inverted for the model parameters using a similar single hidden layer architecture as Röth & Tarantola. The difference in approach was two-fold; the NN employed a single seismogram as input as opposed to whole shot gather and pseudo-spectral data was used for training rather than time waveforms directly. The use of a single waveform did allow for improved result, however the use of pseudo-spectral data was instrumental. Transformed NN learned inference had better accuracies than the conventional time approach. Motivation to use pseudo-spectral data followed the work of Falsaperla et al. (1996) where they identified how the introduced sparsity within pseudo-spectral domain facilitated the learning process for the NN and was more robust to noise.

2.3.2 | Data-Driven Approaches to Velocity Estimation

The terminology of data-driven geophysics is not a novel-one. This was first introduced in the literature by Schultz et al. (1994) when estimating rock properties directly from seismic-data through statistical techniques. However, conceptually, this is similar to the deconvolution process within a seismic processing flow (Robinson, 1957, 1967). Namely, a filter is derived via autocorrelations and applied as a deconvolution operator (Webster, 1978). The term has only recently found a re-invigorated interest. Some modern applications of data-driven geophysical processes include dictionary learning

(Nazari Siahsar et al., 2017), time series analysis (Wu et al., 2018a), fault identification (Mangalathu et al., 2020), and reservoir characterization (Schakel and Mesdag, 2014).

Twenty-one years after Michaels and Smith (1992), Lewis and Vigh (2017) employed DNN architecture to learn prior models for seismic FWI. Their data driven approach at estimating initial models was applied to salt body reconstruction by learning the probability of salt geo-bodies and use this to regularize the FWI objective function. Araya-Polo et al. (2018) utilised DNN architecture and inverted for 2D high-velocity profiles. For the training process, they generated thousands of random 2D velocity models with up to four faults in them, at various dip angles and positions. Each model had three to eight layers, with velocities ranging from 2000 to 4000 ms^{-1}, with layer velocity increasing with depth. The DNN architecture is not defined in their paper, however when applied to unseen data with and without salt anomalies, their results achieved accuracies well above 80% for both cases. This was used to obtain a low-wavenumber starting model then passed to traditional FWI as an initial model. Wu et al. (2018b) proposed a convolutional-based network called "InversionNet" to directly map the raw seismic data to the corresponding seismic velocity model for a simple fault model with flat or curved subsurface layers. More recently, Li et al. (2019) extended this further and developed a DNN framework called "SeisInvNet" to perform the end-to-end velocity inversion mapping with enhanced single-trace seismic data as the input in time domain.

2.3.3 | Wave Physics as an Analogue Recurrent Neural Network

Recently, Raissi et al. (2019) and Hughes et al. (2019) derived a function between the physical dynamics of wave phenomena and RNNs. In their work they propose physics-informed neural networks that are trained to solve supervised learning tasks while respecting the laws of physics described by general non-linear partial differential equations. Fundamental to their approach is the ability for DNNs to be universal function approximators. Within this formulation, Raissi et al. (2019) are able to solve non-linear problems without the need to compute a priori assumptions, perform linearisation or employ local time-stepping. Under this new paradigm in modelling, back-propagation is used to differentiate neural networks with respect to their input coordinates and model parameters to obtain physics-informed neural networks. Such NNs are constrained to respect any symmetries, invariances, or conservation principles originating from the physical laws that govern the observed data, as modelled by general time-dependent and non-linear partial differential equations. In particular, following up from this work, Sun et al. (2019) recast the forward modelling problem in FWI into a deep learning network by recasting the acoustic wave propagation into a RNN framework. Figure 2.12 shows velocity inversion results from Sun et al. (2019) applied to the Marmousi velocity model. These theory-guided inversions still suffer from cycle-skipping, local-minima and high computational cost (Sun et al., 2019). Recent research suggests that Stochastic Gradient Descent algorithms have the capacity to escape local minima to a certain extent (Sun et al., 2019).

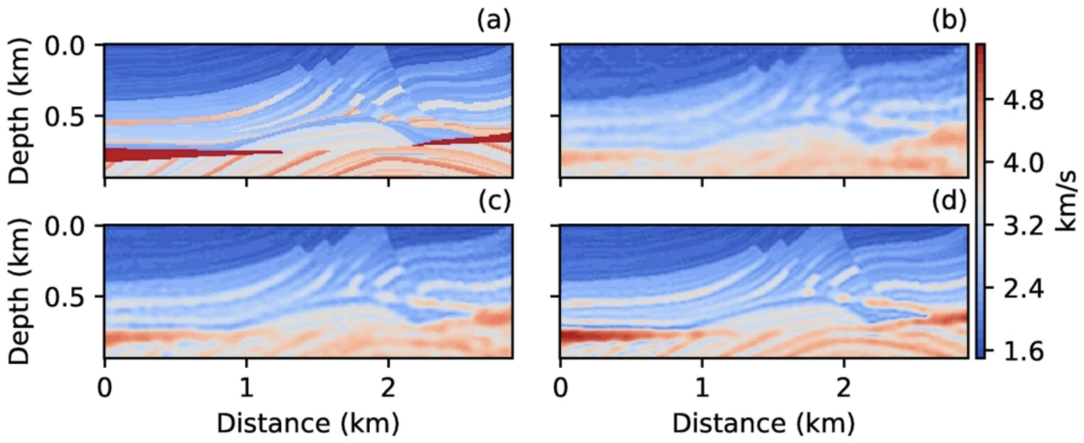


Figure 2.12: The inversion of Marmousi velocity model using RNN forward modelling framework with Adam algorithm optimizer. (a) True Marmousi. (b) 25th iteration. (c) 50th iteration (d) 100th iteration. From Sun et al. (2019).

3

Theoretical Considerations

This chapter reviews and derives the key theoretical approaches used in this dissertation. The first section introduces a classical FWI formulation and summarizes the local optimization problem as a flowchart of elements. Using a combination of ANNs, CNNs and RNNs present in DNN architectures, FWI is recast as a simulation within a deep learning framework. The next two sections introduce the theoretical setups of data-driven and theory-guided pseudo-spectral FWI and associated theory which can be used to solve FWI as a Deep Learning problem. For a more in-depth assimilation, Appendix A provides complementary material.

3.1 | FWI as Local Optimization

Lailly and Bednar (1983) and Tarantola (1984a) recast the migration imaging principle introduced by Claerbout (1971) as a local optimization problem. For an anisotropic medium, particle motion is based on the wave equation given by:

$$\frac{1}{c(\boldsymbol{x})^2}\frac{\partial^2 p(\boldsymbol{x},t)}{\partial t^2} - \nabla^2 p(\boldsymbol{x},t) = s(\boldsymbol{x},t), \tag{3.1}$$

where $p(\boldsymbol{x},t)$ is the pressure wavefield, $c(\boldsymbol{x})$ is the acoustic p-wave velocity and $s(\boldsymbol{x},t)$ is the source. This can be expressed as a linear operator and solved numerically.

After forward modelling the physical system through the data, the objective is to minimize the difference between the observed data and the modelled data. The difference or misfit between the two datasets is known as the misfit-, objective- or cost-function ϕ. The most common cost function is given by the $L_2 - norm$ of the data residuals:

$$\phi(x) = \frac{1}{2}\,||\boldsymbol{y_{obs}} - \boldsymbol{y_{cal}}(x)||_D^2 = \frac{1}{2}\Delta\boldsymbol{y}^\dagger\Delta\boldsymbol{y}, \tag{3.2}$$

where D indicates the data domain given by n_s sources and n_r receivers, † is the transpose and $\boldsymbol{y}_{obs}, \boldsymbol{y}_{cal}$ are the observed and calculated data respectively. The misfit function ϕ can be minimized with respect to the model parameters y by setting the gradient to zero:

$$\nabla\phi = \frac{\partial\phi}{\partial y} = 0. \tag{3.3}$$

To solve for the misfit function, FWI utilizes a linearised and iterative optimisation scheme. Based on the Born approximation in scattering theory (Born and Wolf, 1980), consider the first model calculated to be $\boldsymbol{x}_0$. After the first pass via forward modelling, the model needs to be updated by the model parameter perturbation $\Delta\boldsymbol{x}_0$. This newly updated model is then used to calculate the next update and the procedure continues iteratively until the computed mode is close enough to the true model based on a residual threshold criterion. At each iteration k, the misfit function $\phi(\boldsymbol{x}_k)$ is calculated from model $\boldsymbol{x}_{k-1}$ of the previous iteration giving:

$$\phi(\boldsymbol{x}_k) = \phi(\boldsymbol{x}_{k-1} + \Delta\boldsymbol{x}_k). \tag{3.4}$$

Assuming that the model perturbation is small enough with respect to the model, Equation 3.4 can be expanded via Taylor expansions up to second orders as:

$$\phi(\boldsymbol{x}_k) = \phi(\boldsymbol{x}_{k-1}) + \delta\boldsymbol{x}^T\frac{\partial\phi}{\partial x} + \frac{1}{2}\delta\boldsymbol{x}^T\frac{\partial^2\phi}{\partial x^2}\delta\boldsymbol{x}. \tag{3.5}$$

Taking the derivative of Equation 3.5 and minimizing to determine the model update leads to:

$$\partial\boldsymbol{x} \approx -\boldsymbol{H}^{-1}\nabla_x\phi, \tag{3.6}$$

where $\boldsymbol{H} = \frac{\partial^2\phi}{\partial x^2}$ is the Hessian matrix and $\nabla_x\phi$ the gradient of misfit function evaluated at $\boldsymbol{x}_0$. The Hessian matrix is symmetric and represents the curvature trend of the misfit function.

3.1.1 | Model Update

Several methods can be employed to solve the model update. Newton methods update the model directly (Newton, 1687), whilst Gauss-Newton use Hessian approximations (Tarantola, 1984b). The latter is referred to as the step-length (Menke, 1989) and Equation 3.6 becomes:

$$\partial x \approx -\alpha\nabla_x\phi, \tag{3.7}$$

where α is the step-length parameter and the magnitude of α is derived via first perturbing the initial model $\boldsymbol{x}_0$ by a differential change in the direction opposite to the gradient.

3.1.2 | Regularization

FWI often turns out to be an ill-posed problem given the incomplete parameter space measured in the field. Artefacts and over-fitting might be introduced in the model due to noise or high frequency component of the data (Sirgue and Pratt, 2004). Well-posedness can be imposed on the model through *a priori* information (Asnaashari et al., 2013). This is introduced into the misfit function (Equation 3.2) with the addition of Tikhonov $L_2 - norm$ regularization (Tikhonov and Arsenin, 1977):

$$\phi(x) = \frac{1}{2}\,||\boldsymbol{y}_{\boldsymbol{obs}} - \boldsymbol{y}_{\boldsymbol{cal}}\,(\boldsymbol{x})||_D^2 + \frac{1}{2}\lambda\,||\boldsymbol{x} - \boldsymbol{x}_{a\,priori}||_M^2\,, \tag{3.8}$$

where $||\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{prior}}||_M^2$ is the regularization term and λ is the regularisation parameter which controls the trade-off between data and model residuals. Namely, the regularization parameter, λ, gives relative weight to model optimization term with respect to data optimisation term and acts as a smoother on the modelling (Tikhonov, 1963).

3.1.3 | Difference between Time and Pseudo-spectral FWI

Forward modelling for FWI can be done in the pseudo-spectral or time-domain. In case of an attenuating medium, the pseudo-spectral domain is the preferred method as frequency dependent attenuation (quality factor Q) is represented by the imaginary component of the velocity (Pratt, 1999). Frequency domain application requires the solution of a linear system of equations by a factorization method (Sirgue and Pratt, 2004). This improves the chance for the inversion to locate the global minimum (Sirgue and Pratt, 2004), however it scales poorly with the size of the problem (Operto et al., 2007) or else requires assumptions on the physical propagation of the wave equation (Ben Hadj Ali et al., 2007). Time-domain approach has a simpler implementation (Vigh and Starr, 2008), however it is highly more sensitive to cycle skipping (Vigh and Starr, 2008) and prone to problematic low-wavenumber estimation for the gradient (Sirgue and Pratt, 2004). Time-domain approaches either consider Q to be constant or utilise some relaxation mechanism, thus not fully representing the physics of the underlying problem (Blanch et al., 1995). Either approach has been successfully applied to production datasets, both with merits in different environments. For time implementation reference is made to Vigh and Starr (2008), Vigh et al. (2010), Liu and Sen (2011) and Cai et al. (2015), and for frequency to Ben-Hadj-Ali et al. (2008), Operto et al. (2015), Plessix (2009). Differences in numerical implementation between time and frequency FWI given in Appendix A.1.

3.1.4 | FWI Algorithm Summary

Excluding considerations related to the practical implementation of FWI, the local optimization iterative scheme for FWI described in the previous section can be summarised in Algorithm 1 and schematic is illustrated in Figure 3.1.

Algorithm 1: FWI as Local Optimization

```
1 Choose an initial model $x_0$ and source wavelet $s(\boldsymbol{x})$.
2 For each source, the forward problem $\Gamma : X \mapsto Y$ is solved everywhere in the
   model space to get a predicted wavefield $\boldsymbol{y}_i$. This is sampled at receivers $r(\boldsymbol{x})$.
3 At every receiver, data residuals are calculated between the modelled wavefield
   $\boldsymbol{y}_i$ and the observed data $\boldsymbol{y}_{obs}$.
4 Data residuals are back-propagated to produce a residual wavefield.
5 For each source location, the misfit function $\phi(\boldsymbol{x})$ is applied for the observed
   data and back-propagated residual wavefield to generate the gradient $\nabla_\phi$
   required at every point in the model.
6 The gradient is scaled based on the step-length $\alpha$, applied to the starting model
   and an updated model is obtained $\boldsymbol{x}_{i+1}$.
7 The process is iteratively repeated from Step 2 until a convergence criterion is
   satisfied.
```

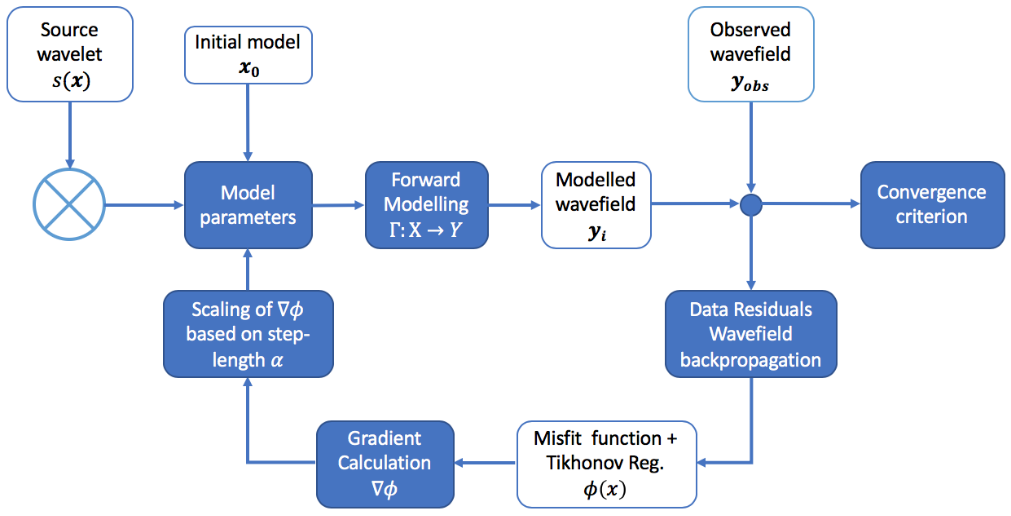

Source wavelet $s(\boldsymbol{x})$
Initial model $\boldsymbol{x}_0$
Observed wavefield $\boldsymbol{y}_{obs}$
Model parameters
Forward Modelling $\Gamma: X \to Y$
Modelled wavefield $\boldsymbol{y}_i$
Convergence criterion
Scaling of $\nabla\phi$ based on step-length α
Data Residuals Wavefield backpropagation
Gradient Calculation $\nabla\phi$
Misfit function + Tikhonov Reg. $\phi(x)$

Figure 3.1: Schematic FWI workflow solved as an iterative optimization process.

3.2 | FWI as a Data-Driven DNN

When applied to inverse problems, NNs can simulate the non-linear functional of the inverse problem $\Gamma^{-1} : Y \mapsto X$. That is, using a NN, a non-linear mapping can be learned which will minimize

$$||\boldsymbol{x} - g_{\Theta}(\boldsymbol{y})||^2 , \tag{3.9}$$

where g_{Θ} is the learned formulation of the inverse problem functional Γ^{-1}, and Θ the large data set of pairs $(\boldsymbol{x}, \boldsymbol{y})$ used for the learning process (Lucas et al., 2018).

3.2.1 | Artificial Neuron, Perceptron and Multi-Layer Perceptron

The most elementary component in a NN is a neuron. This receives excitatory input and sums the result to produce an output or activation (Raschka and Mirjalili, 2017). For a given artificial neuron, consider n inputs with signals $\boldsymbol{x}$ and weights $\boldsymbol{w}$. The output $\boldsymbol{y}$ of a neuron from all input signals is given by:

$$y = \sigma\left(b + \sum_{j=0}^{n} w_j x_j\right) , \tag{3.10}$$

where σ is the activation function and b is a bias term enabling the activation functions to shift about the origin. Popular activations functions include Binary Step, Linear, Sigmoid, Tanh, Softmax and ReLu functions (Goodfellow et al., 2016). The ReLu or Rectified linear unit is the most widely used activation function. It is non-linear, allowing easily back-propagation of errors. When employed on a network of neurons, the negative component of the function is converted to zero and the neuron is deactivated, introducing sparsity with the network and making it efficient and easy for computation. More information on the other functions is provided in Appendix A.2.

3.2.2 | Feed-forward Architectures and Deep Networks

The architecture of a NN refers to the number of neurons, their arrangement and connectivity. When a single neuron is used in the NN architecture, the result is a Perceptron. Stacking multi layers of the simple neurons and fully connecting all the inputs results in the multi-layer perceptron or fully connected layer. Within MLP architecture, the input from the initial nodes $\boldsymbol{x}$ is connected to a hidden layer of neurons, or a sequence of these neurons, and the output layer of the neurons. Communication proceeds layer by layer from the input layer via the hidden layers up to the output layer. Figure 3.2 shows a fully connected MLP consisting of 2 hidden layers. The output of the unit in each layer is the result of the weighted sum of the input units, followed by a non-linear element-wise function. The weights between each unit are learned as a result of a training procedure.

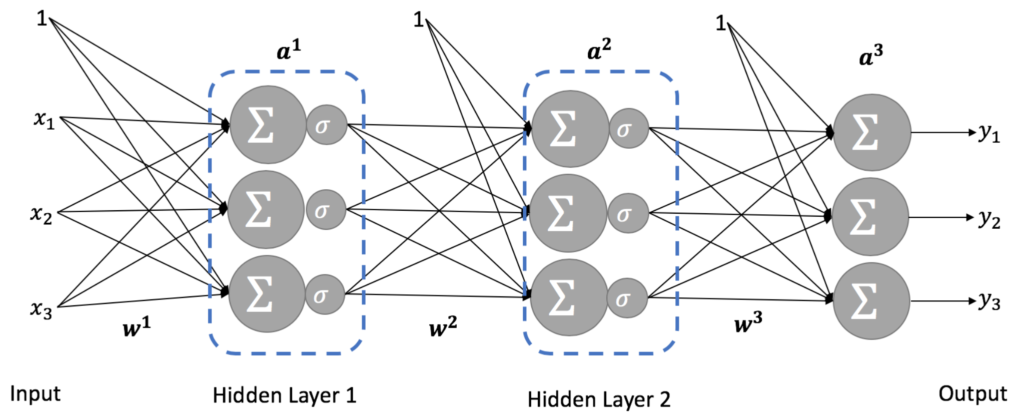


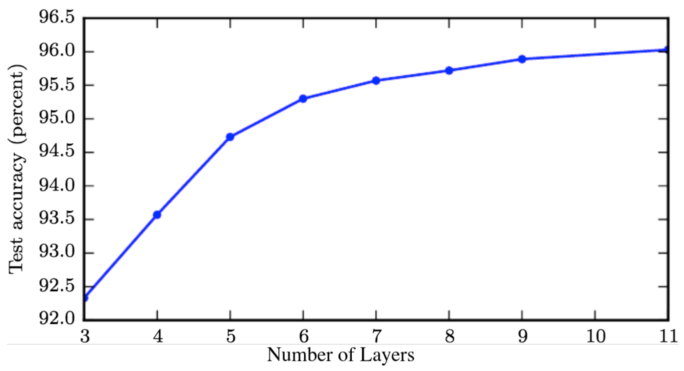
Figure 3.2: Fully connected NN with 2 hidden layers. Adapted from Lucas et al. (2018).

When designing a NN, choosing the depth of the network and the width of each layer is key. A network with a single hidden layer is sufficient to fit the training set (Goodfellow et al., 2016). Deeper NN are able to generalize better to different non-linear functions (see Figure 3.3) and are harder to optimize (Haykin, 2009). Indeed, Montúfar et al. (2014) showed that a shallow NN equivalent to a DNN could require an exponential number of hidden units.

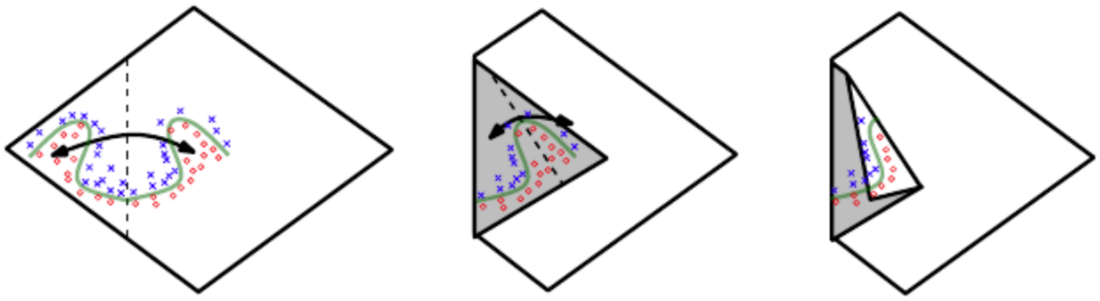
3.2.3 | A Universal Approximation Framework

A fundamental attribute to all deep feed-forward NN is that these provide a universal approximation framework (Hornik et al., 1989). In particular, the Universal Approximation Theorem states that a feed-forward network with a linear output layer and at least one hidden layer with an appropriate activation function can approximate any Borel measurable function[1] from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network has sufficient hidden layers (Goodfellow et al., 2016; Leshno et al., 1993).

[1]The real-valued function f defined with domain $\mathcal{E} \subset \Omega$, for measurable space $(\Omega, \mathcal{F})$, is Borel measurable with respect to $\mathcal{F}$ if the inverse image of set $\mathcal{B}$, defined as $f^{-1}(\mathcal{B}) \equiv \{\omega \in \mathcal{E} : f(\omega) \in \mathcal{B}\}$ is an element of σ-algebra $\mathcal{F}$, for all Borel sets $\mathcal{B}$ of $\mathbb{R}$ (Stephens, 2006).



(a) Deeper networks have been empirically shown to improve the accuracy and generalize better. Adapted from Goodfellow et al. (2015)



(b) Visual representation of the improved generalization for deep NNs over shallow NNs.
Left: A shallow NN framework has the same output for every pair from its input. The axis of symmetry is given by the hyperplane defined from the weights and bias of the units. A function computed on top of that unit (the green decision surface) will be a mirror image of a simpler pattern.
Centre: A simpler pattern can be obtained when considering the axis of symmetry.
Right: Another repeating pattern can be folded on top of the first to obtain another symmetry (which is now repeated four times, with two hidden layers). Adapted from Montúfar et al. (2014).

Figure 3.3: Advantages of deep NN over shallow NN.

3.2.4 | Back-Propagation and Learning with DNN

When training a DNN, the forward propagation through the hidden layers from input x to output y needs to be measured for its misfit. In terms of regression modelling, the most commonly used cost function is the Sum of Squared Error, defined as:

$$\mathcal{J} \equiv \frac{1}{2} \sum_{j=1}^{\mathcal{J}} \left(y^i - y_{true} \right)^2, \tag{3.11}$$

where y_{true} is the labelled true datasets and y^i is the output from the i^{th} pass forward pass through the network.

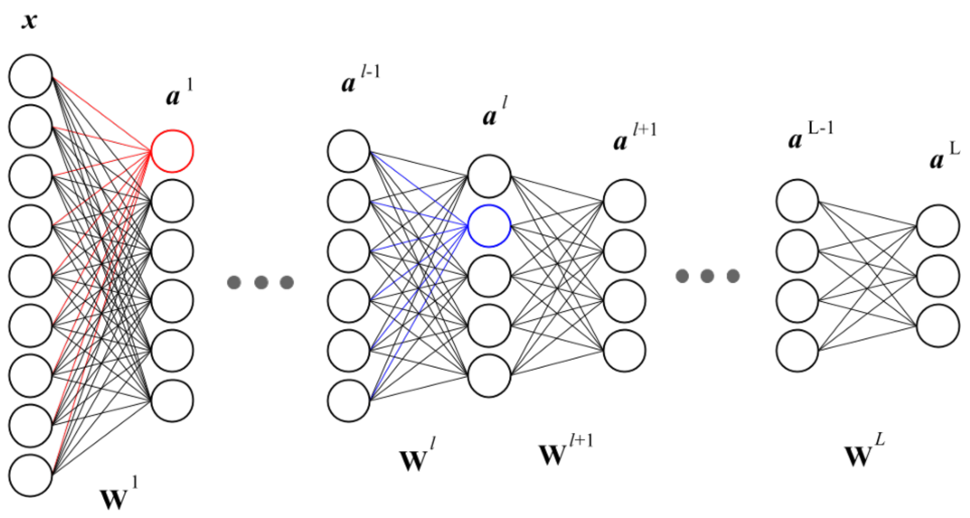


Figure 3.4: Schematic of a DNN with input $\boldsymbol{x}$, L layers, $\boldsymbol{a}^l$ activation function at layer l and weights $\boldsymbol{w}^l$. From Hallström (2016).

Consider the schematic for a fully connected DNN shown in Figure 3.4. Applying forward propagation through the layers of neurons:

$$\text{Input sum of neuron } k \text{ in layer } l \qquad z_k^l = b_k^l + \sum_j w_{kj}^l a_j^{l-1} \tag{3.12}$$

$$\text{Activation function on layer } l \qquad a_k^l = \sigma\left(z_k^l\right) \tag{3.13}$$

$$\text{Input sum of neuron } m \text{ in layer } l+1 \qquad z_m^{l+1} = b_m^l + \sum_k w_{mk}^{l+1} a_k^l. \tag{3.14}$$

The objective is to minimize the function $\mathcal{J}$ with respect to the weights. Employing the chain rule, the derivative with respect to a single weight in layer l is given as:

$$\frac{\partial \mathcal{J}}{\partial w_{kj}^l} = \frac{\partial \mathcal{J}}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{kj}^l} = \frac{\partial \mathcal{J}}{\partial a_k^l}\frac{\partial a_k^l}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{kj}^l}. \tag{3.15}$$

Substituting Equations 3.12-3.14 in Equation 3.15 yields

$$\frac{\partial \mathcal{J}}{\partial w_{kj}^l} = \left(\sum_m \frac{\partial \mathcal{J}}{\partial z_m^{l+1}}\frac{\partial z_m^{l+1}}{\partial a_l^k}\right)\frac{\partial a_k^l}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{kj}^l} = \left(\sum_m \frac{\partial \mathcal{J}}{\partial z_m^{l+1}} w_{mk}^{l+1}\right)\sigma'\left(z_k^l\right) a_j^{l-1}. \tag{3.16}$$

The error signal of a neuron k in layer l is defined as the total error when the input sum of the neuron is changed and is given by:

$$\delta_k^l \equiv \frac{\partial \mathcal{J}}{z_k^l}. \tag{3.17}$$

Substituting Equations 3.12-3.14 in Equation 3.16 results in a recursive formulation for the error given by

$$\delta_k^l = \left(\sum_m \delta_m^{l+1} w_{mk}^{l+1} \right) \sigma^{'}(z_k^l). \tag{3.18}$$

Estimation of the error of the neurons in the final layer L can be a sequential calculation of the error from the previous layer, until all error signals within the network are thus computed. The only derivative to be calculated is the derivative of the cost function $\sigma^{'}$, which is the output of the network. Bias b_k^l for l^{th} layer is a weight to be optimized and the error signal for the bias is given by:

$$\frac{\partial \mathcal{J}}{\partial b_k^L} = \frac{\partial \mathcal{J}}{\partial z_k^l} \underbrace{\frac{\partial z_k^l}{\partial b_k^l}}_{1} = \delta_k^l. \tag{3.19}$$

Hence, error signals for all neurons in the network can be recursively calculated throughout the network and the derivative of the cost function with respect to all the weights can also be calculated. Training of the DNN is achieved via gradient descent algorithm, referred to as Delta Rule in the machine learning community (Sutton, 1988). A small fraction of the derived derivative is subtracted from the weight and updated weight is given as:

$$w_{kj}^l = w_{kj}^l - \eta \delta_k^l, \tag{3.20}$$

where η is a small scalar referred to the learning rate which controls how much of an adjusting is applied to the DNN with respect to the loss gradient. This is synonymous to the model update in FWI presented in § 3.1.1. Equivalence of back-propagation and the classical adjoint method is numerically shown in Appendix A.9.

3.2.5 | Optimizing the Loss Function

Updating of the error gradient in a steepest gradient descent manner might be conceptually straightforward to understand. However, a major drawback with this algorithm is the high risk of getting stuck in local minima (Fletcher, 1987). This is an active area of research and Ruder (2016) does an extensive review of optimizers. Four of the most widely used are (i) Adagrad, (ii) Adadelta, (iii) Adam and (iv) RMSprop. Difference in the implementation for these optimizers is given in Appendix A.3.

3.2.6 | Generalization of DNN for "unseen" Data

Central to DNN is how to make the network able to perform well not just on the training data, but also on new "unseen" inputs. DNN resolve this via two regularization strategies: (i) functional-based, and (ii) NN architecture-based regularizations. These are schemes which update the cost function. On the other hand, architecture-based regularization are alterations to the NN architecture in the form of (i) Dropout, (ii) Data augmentation, and (iii) Early Stopping. Further information on these is available in Appendix A.4. Either of these are essential for DNN as shown in Figure 3.5.

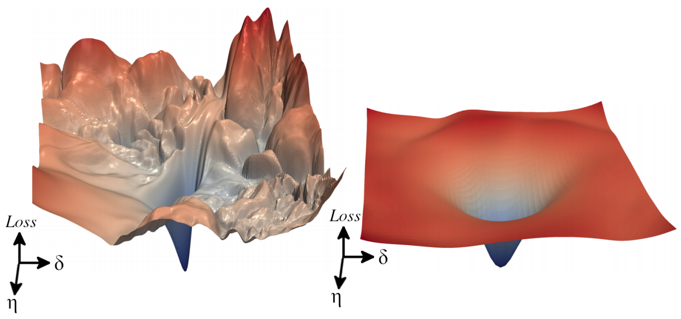


Figure 3.5: NN loss function surface without (Left) and with (Right) regularization. Two random vectors (δ, η) are chosen on the parameter space and values of the loss computed. Vertical axes are in logarithmic scale to show dynamic range. It is highly evident that without regularization, the loss function is susceptible to local minima and at risk to converge to sub-optimal levels. On the other hand, regularization promotes flatness and prevent chaotic behaviour. Adapted from Li et al. (2018a).

3.2.7 | Convolutional Neural Networks

CNNs are feed forward multi-layered networks with layers consisting of convolutions and linear transformers able to perform multiple transformations (LeCun et al., 2010). Apart from Dropout, Activation functions and Fully Connected Layers components, CNNs introduce (i) Convolutional, (ii) Pooling, and (iii) Batch Normalization layers. Convolutional layers add convolutions across the neurons. Pooling layers are dimension reducing layers applied after convolutional layers since location of elements within feature-maps become less important as long as their position relative to others is preserved (Khan et al., 2020). Batch Normalization shifts the covariance of the distribution of hidden units to zero mean and unit variance for each batch (Ioffe and Szegedy, 2015), ensuring smooth gradient descent and regularization (Santurkar et al., 2018). Further details on the specific building blocks for CNNs is available in Appendix A.5.

3.2.8 | Neural Network Architecture Types

Combining the components introduced in the previous sections, different DNN architectures can be developed. The simplest form for a network would be MLP. This is considered to be a rudimental and basic network. An improvement would be the inclusion of convolutional, max-pooling and batch normalization layers in either one-dimension (1D) or two-dimensions (2D). One such architecture is AlexNet (Krizhevsky et al., 2012). Khan et al.'s 2020 survey provides a detailed overview of the different CNN architectures and how they evolved over time. In particular, two architectures which are considered state-of-the-art as a consensus in the literature are VGG (Simonyan et al., 2014) and ResNet (He et al., 2016a). These are only three of a myriad of architectures (Figure 3.6). More information on these architectures is available in Appendix A.6.

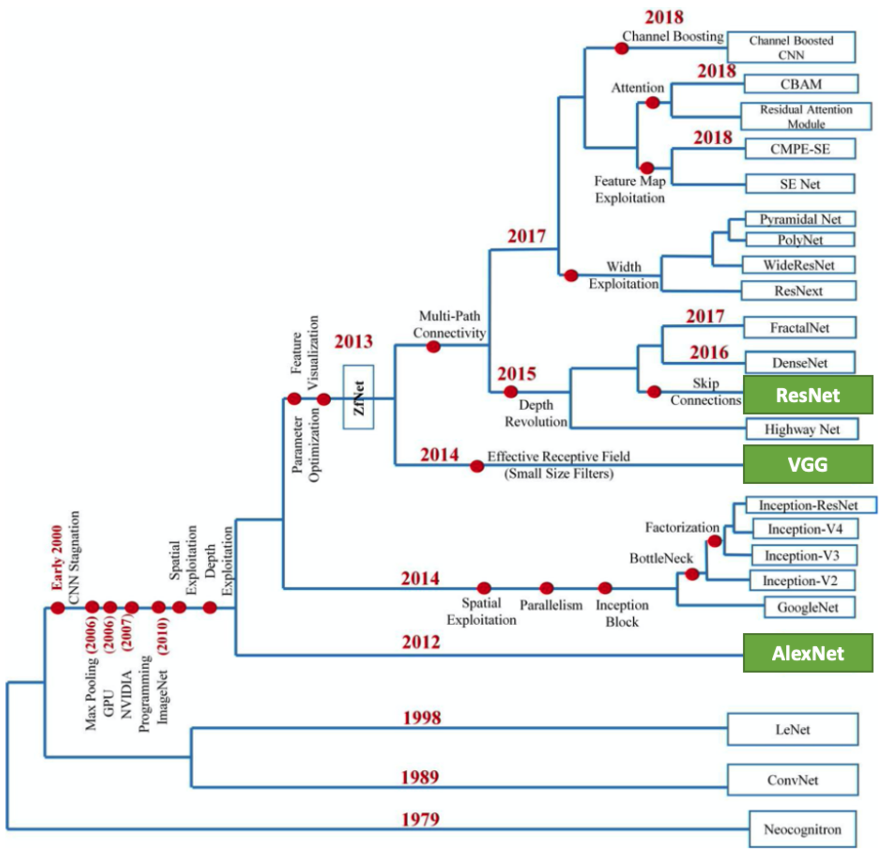


Figure 3.6: Historic overview of CNN architectures, highlighting timeline position for AlexNet, VGG and ResNet. Adapted from Khan et al. (2020).

3.2.9 | Outline for Solving FWI with Data-Driven DNNs

Utilizing NN architecture and formulae, training of a DNN for FWI can be summarized as in Algorithm 2. Schematic of the learning process for DNN is given Figure 3.7.

Algorithm 2: FWI as a data-driven DNN

```
1 Setup a deep architecture from those shown in § 3.2.8, with regularization
    measures.
2 Initialise the set of weights $\boldsymbol{w}^l$ and biases $\boldsymbol{b}^l$.
3 Forward propagate through the network connections to calculate input sums
    and activation function for all neurons and layers.
4 Calculate the error signal for the final layer $\delta^L$ by choosing an appropriate
    differentiable activation function.
5 Back propagate the errors for all neurons in each $l$ layer $\delta^l$.
6 Differentiate the cost function with respect to biases $\left(\frac{\partial \mathcal{J}}{\partial b^l}\right)$
7 Differentiate the cost function with respect to weights $\left(\frac{\partial \mathcal{J}}{\partial w^l}\right)$
8 Update weights $\boldsymbol{w}^l$ via gradient descent.
9 Recursively repeat from Step 3 until the desired convergence criterion is met.
```

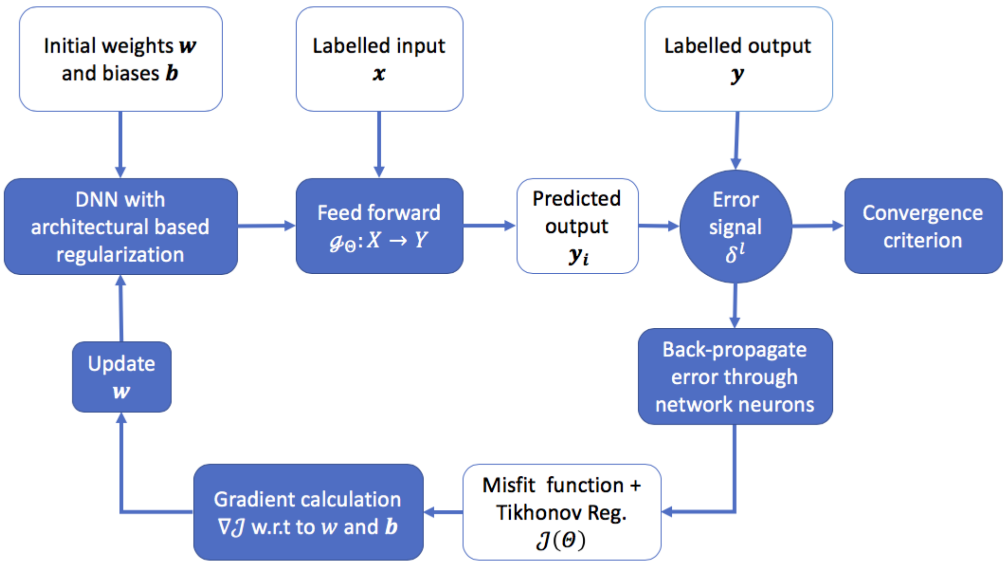

Initial weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$
Labelled input $\boldsymbol{x}$
Labelled output $\boldsymbol{y}$
DNN with architectural based regularization
Feed forward $g_\Theta : X \to Y$
Predicted output $\boldsymbol{y}_i$
Error signal δ^l
Convergence criterion
Update $\boldsymbol{w}$
Back-propagate error through network neurons
Gradient calculation $\nabla \mathcal{J}$ w.r.t to $\boldsymbol{w}$ and $\boldsymbol{b}$
Misfit function + Tikhonov Reg. $\mathcal{J}(\boldsymbol{\theta})$

Figure 3.7: Schematic of DNN workflow solved via supervised learning with input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$. This is analogous to the FWI algorithm previously shown in Figure 3.1, with the difference that feed forward operator $g_\Theta : X \mapsto Y$ is a learned function.

3.3 | Theory-Guided RNN as an Analogue of FWI

Feed forward NNs presented in § 3.2 do not allow for cyclical connections between neurons. If this condition is relaxed, we obtain RNNs. This difference is illustrated in Figure 3.8. Although this might seem trivial, cyclical connections directly enable sequence learning as they allow a "memory" of previous inputs to persist in the network's internal state. RNN theoretically can map the entire history of previous inputs to each output (Graves, 2012). Through equivalence results of Universal Approximation Theorem for MLP, Hammer (2000) prove that a RNN with sufficient number of hidden units can approximate any measurable sequence-to-sequence mapping to arbitrary accuracy.

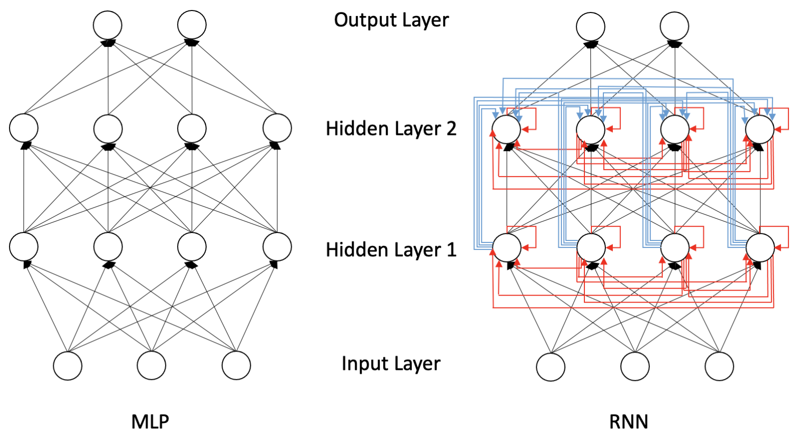


Figure 3.8: Difference between MLP and RNN architectures are Red and Blue cyclical connections. Red are intra-layer connections, whereas Blue are across layers.

3.3.1 | Forward Pass

The forward pass of a RNN is identical to that of a MLP, except that both the current input and the hidden layer activations from the previous time-step contribute to the current activation function.

Consider a sequence x of length T representing a RNN with input units I, hidden units H and output units K. Let x_i^t be the value of input i at time t, and let a_j^t and b_j^t be respectively the network output to unit j at time t and the activation of unit j at time t. For the hidden units we have

$$a_h^t = b_h^0 + \sum_{i=1}^{I} w_{ih} x_i^t + \sum_{h'=1}^{H} w_{h'h} b_{h'}^{t-1}, \tag{3.21}$$

where b_h^0 is the initial bias term. Zimmermann et al. (2006) show how initial non-zero bias improve RNN stability and performance. This term is set to always zero to simplify the equation notation. Non-linear, differentiable activation functions σ are applied similarly to MLP to get:

$$b_h^t = \sigma_h\left(a_h^t\right) = \sigma_h\left(\sum_{i=1}^{I} w_{ih}x_i^t + \sum_{h'=1}^{H} w_{h'h}b_{h'}^{t-1}\right). \tag{3.22}$$

The complete sequence of hidden activations can be calculated by starting at $t = 1$ and recursively applying Equation 3.22. The output y_k^t at the k^{th} neuron at time t is then given by:

$$y_k^t = \sum_{h=1}^{H} w_{hk}b_h^t. \tag{3.23}$$

3.3.2 | Backward Pass

Given that components of the forward pass and the loss function definition are synonymous with those of MLPs, what is left to determine are the derivatives with respect to the model weights. There are two widely used algorithms for this: (i) Real Time Recurrent Learning developed by Robinson and Fallside (1987), and Back-Propagation Through Time by Werbos (1988). Back-Propagation Through Time will be considered going forward. A review of the implementation of Real Time Recurrent Learning is given by Williams and Zipser (1989) or more recently by Haykin (2009).

Like standard back propagation, Back-Propagation Through Time consists of repeated application of the chain rule. The only difference is that for RNN, the loss function $\mathcal{J}$ is now dependent on the activation of the hidden layer at the next time-step. Equation 3.18 can be re-written as:

$$\delta_h^t = \frac{\partial \mathcal{L}}{\partial a_j^t} = \theta'\left(a_h^t\right)\left(\sum_{k=1}^{K} \delta_k^t w_{mk} + \sum_{h'=1}^{H} \delta_{h'}^{t+1} w_{hh'}\right). \tag{3.24}$$

The complete sequence of partial derivate terms can be calculated by starting at time $t = T$ and recursively applying Equation 3.24. Summing over the whole sequence to get network weights results in:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_j^t b_i^t. \tag{3.25}$$

3.3.3 | Vanishing Gradient

The range of information accessed by standard RNN architectures can be quite limited. The problem is that the influence of a given input on the hidden layer either decays or blows up exponentially as it cycles through the network's recurrent connections Graves (2012). This is known as the vanishing gradient problem (Hochreiter, 1991) and is schematically illustrated in Figure 3.9. Numerous attempts were made in the 1990s to address this problem. These included gradient-descent variants (Elman, 1990; Fahlman, 1991; Pearlmutter, 1989; Schmidhuber, 1992b; Williams and Zipser, 1989), explicitly introduced time delays or time constants (Lang et al., 1990; Lin et al., 1996; Mozer, 1992; Plate, 1993), simulated annealing and discrete error propagation (Bengio et al., 1994), hierarchical sequence compression (Schmidhuber, 1992a) and Kalman filtering (Puskorius and Feldkamp, 1994). A comprehensive list is available within Hochreiter and Schmidhuber (1997).

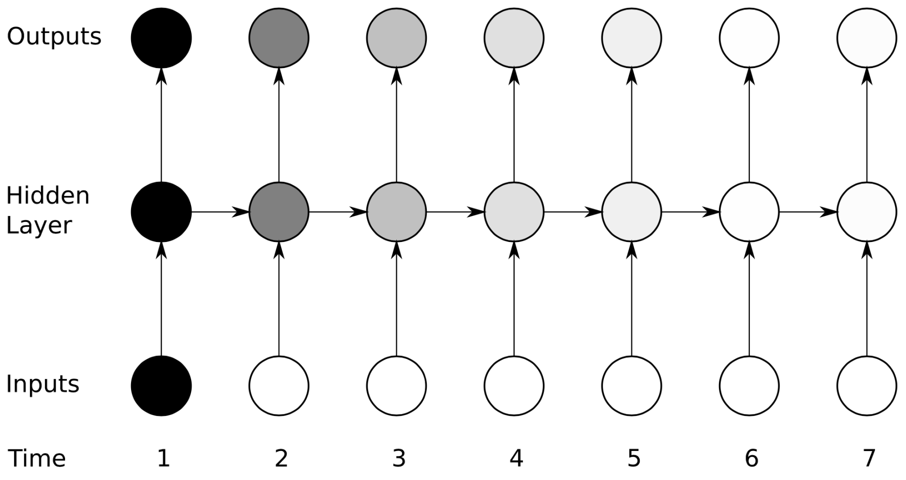
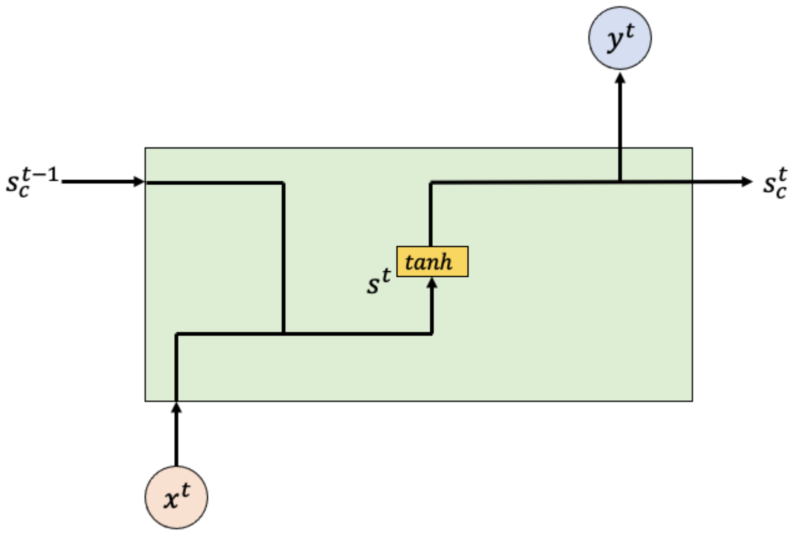


Figure 3.9: Vanishing gradient problem for RNNs. The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one. The darker the shade, the greater the sensitivity. The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network "forgets" the first inputs. From Graves (2012).

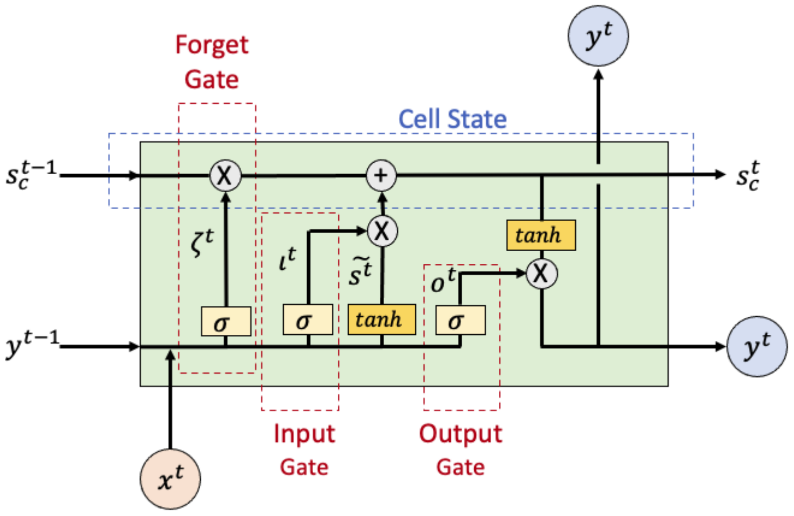
3.3.4 | Long Short-Term Memory (LSTM)

Apart from the subset of algorithms able to handle the vanishing gradient problem in the previous section, Hochreiter and Schmidhuber (1997) introduce a different type known as LSTM. This NN introduces a set of recurrently connected subnets, known as memory blocks (the input, output and forget gates), and the cell state. Figure 3.10(a) shows a standard RNN with a single tanh layer. Figure 3.10(b) shows the LSTM chain structure but with the additional four interaction layers. Mathematical detail for each of these components is given in Appendix A.8.

As before, w_{ij} is the weight from unit i to unit j, a_j^t is the network input to unit j at time t and b_j^t is the activation of unit j at time t. Given the LSTM memory blocks, the subscripts ι, ζ and o refer to the input, forget and output gate respectively, the subscript c refers to one of the memory cells C and s_c^t is the state of cell c at time t.



(a) The repeating module in a standard RNN contains a single layer.



(b) A LSTM memory block has an additional three gates – Input, Output and Forget Gate (red) and a cell state block (blue).

Figure 3.10: Comparison between RNN and LSTM blocks. Adapted from Olah (2015).

3.3.5 | Network Training

The theory reviewed for MLP optimizers in § 3.2.5 and regularization in § 3.2.6 remain unchanged for LSTMs.

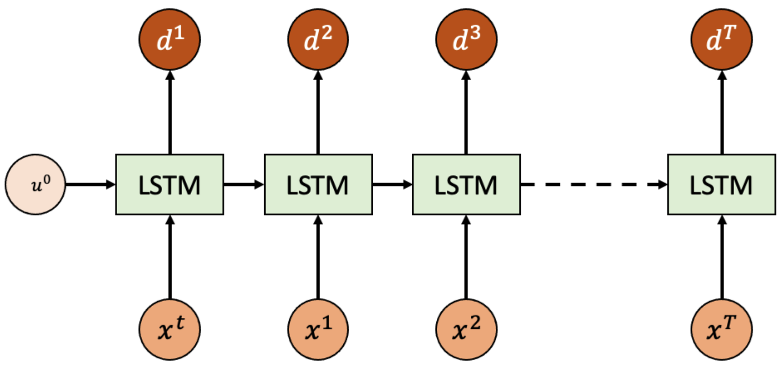
3.3.6 | LSTM as a Substitute for Wave Propagation

Revisiting the discretized FD stencil for wave propagation given as,

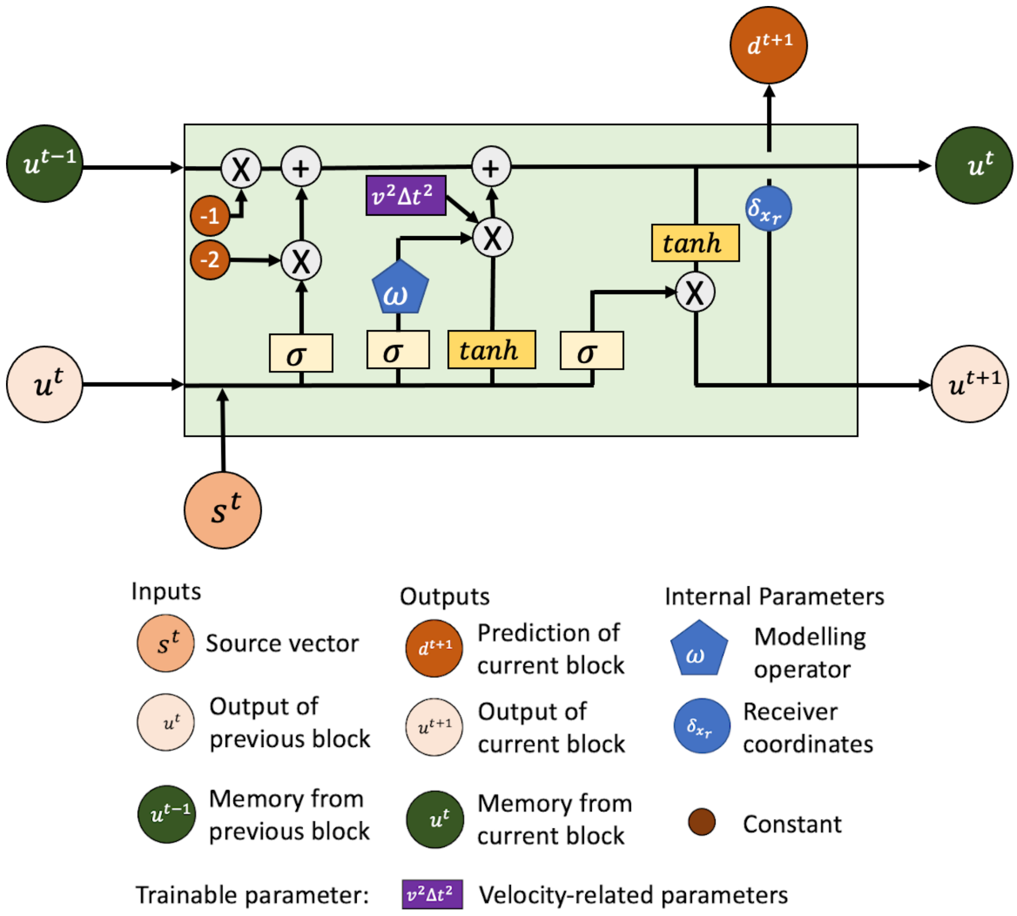
$$p_j^{n+1} = \partial t^2 \left[c_j^2 \mathcal{F}^{-1} \left[k^2 \mathcal{P}_v^n \right] + s_j^n \right] + 2p_j^n - p_j^{n-1}, \tag{3.26}$$

it is clear how pressure wave p and source impulse s at current time step n are not affected by the future values $n+1$, but only dependent on the previous state of pressure at $n-1$. This is, by definition, a finite impulse with directed acyclic graph under graph theory definitions (Thulasiraman and Swamy, 2011). With slight modification to the LSTM blueprint in Figure 3.10(b), a Deep Learning architecture supporting forward modelling can be cast as a LSTM cell using a finite difference operator that takes the pressure wavefield at one point in time $n-1$, produces the modelled shot record at current time n and stores this in memory for the next step $n+1$. Collection of every output at every time instant and sorting with respect to the time coordinate would equal to the measurements of the wavefield locally at a geophone. This LSTM architecture capable of simulating wave propagation is shown in Figure 3.11(a) as an unrolled graph and Figure 3.11(b) as the building block components within an LSTM.

The inputs to the LSTM cell are the source term at current time s^t, the wavefield at current u_t and previous time step u_{t-1} stored in memory of the LSTM. These wavefields are combined together with untrainable modelling operator ω and constants -1 and -2 to replicate the incremental time stepping in forward modelling. Deciding to model in time is equivalent to setting ω to the Laplacian, whereas setting it to calculate pseudo-spectral second order derivates will lead to pseudo-spectral wavefield modelling. The trainable velocity-related parameter $v^2\Delta t^2$ is applied to get the current modelled wavefield u^{t+1}. This is stored in memory, passed to the forget gate and receiver location discretization δ_{x_r} applied to get the predicted outputs d^{t+1}. To train the velocity parameters, seismic shot records are provided as labelled data for training.



(a) Unrolled form of acyclic graph of LSTM for FWI.



(b) Modified LSTM cell block supporting of forward modelling.

Figure 3.11: Recasting of forward modelling of FWI within an LSTM deep learning framework. Adapted from Sun et al. (2019).

3.3.7 | Outline for Solving FWI with a RNN

Physics-informed RNNs for FWI is identical to classical FWI, apart from the forward modelling component which is done within an LSTM framework. This is summarized by Algorithm 3 and schematic shown in Figure 3.12.

Algorithm 3: FWI as RNN Implementation

```
1 Choose an initial model $x_0$ and source wavelet $s(x)$.
2 For each source, solve the forward problem through LSTM time-stepping to get
    a predicted wavefield $y_i$. This is sampled at receivers $r(x)$.
3 At every receiver, data residuals are calculated between the modelled wavefield
    $y_i$ and the observed data $y_{obs}$.
4 Data residuals are back-propagated to produce a residual wavefield.
5 For each source location, the misfit function $\phi(x)$ is applied for the observed
    data, regularized and back-propagated through residual wavefield to generate
    the gradient $\nabla_\phi$ required at every point in the model.
6 The gradient is scaled based on loss optimization function, applied to the
    starting model and an updated model is obtained $x_{i+1}$.
7 The process is iteratively repeated from Step 2 until a convergence criterion is
    satisfied.
```

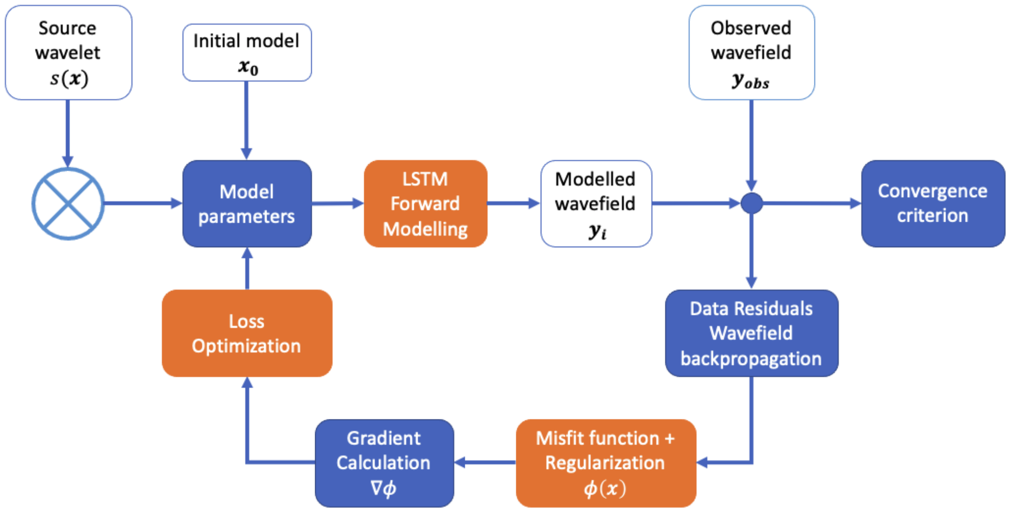

Source wavelet $s(x)$
Initial model x_0
Observed wavefield y_{obs}
Model parameters
LSTM Forward Modelling
Modelled wavefield y_i
Convergence criterion
Loss Optimization
Data Residuals Wavefield backpropagation
Gradient Calculation $\nabla\phi$
Misfit function + Regularization $\phi(x)$

Figure 3.12: Schematic of RNN workflow. The orange components are modified from the original FWI workflow in Figure 3.1.

4

Numerical Results

In the previous section, two formulations of FWI within Deep Learning frameworks were presented. This chapter presents numerical results and additional outcomes of these implementations.

4.1 | Software Setup

Throughout the work shown in this chapter, Python 3.7 with the Anaconda distribution was used as the primary development language.

4.2 | FWI as a Data-Driven DNN

4.2.1 | Train-Test Data Split

Learning the inversion from time to pseudo-spectral domain requires a training dataset which maps time to pseudo-spectral components and their respective velocity profile. A data generator was designed to create synthetic data on-the-fly for a 2000ms time window. The steps involved in the data generator are:

i) Randomly create velocity profile v_p for a 2000ms distance, with value ranging from 1450ms^{-1} and 4500ms^{-1}. The lower bound of 1400ms^{-1} was selected for the water column since the observed velocity in normal off-shore seismic exploration conditions ranges from 1440ms^{-1} to 1540ms^{-1} (Cochrane and Cooper, 1991). The upper bound of 4000ms^{-1} was selected as this is the upper limit of velocity in porous and saturated sandstones (Lee et al., 1996). The assumption is made that limestones, carbonates and salt deposits are not present in the subsurface model being inverted since these would have velocity in excess of 4000ms^{-1} and go beyond the above defined parameters.

ii) Calculate the density ρ based on Gardner's equation (Gardner et al., 1974) given by $\rho = \alpha v_p^{\beta}$ where $\alpha = 0.31$ and $\beta = 0.25$ are empirically derived constants that depend on the geology.

iii) At each interface, calculate the Reflection Coefficient $\mathcal{R} = \frac{\rho_2 v_{p_2} - \rho_1 v_{p_1}}{\rho_2 v_{p_2} + \rho_1 v_{p_1}}$ where ρ_i is density of medium i and v_p is the p-wave velocity of medium i.

iv) For each medium, calculate the Acoustic Impedance $\mathcal{Z} = \rho v_p$.

v) Define a wavelet $\mathcal{W}$. This was selected to be a Ricker wavelet at 10Hz (Ryan, 1994). The Ricker wavelet is a theoretical waveform that takes into account the effect of Newtonian viscosity and is representative of seismic waves propagating through visco-elastic homogeneous media (Wang, 2015), thus making it ideal for this numerical simulation. The central frequency of 10Hz was chosen as a nominal value based on literature results to be representative of normal FWI conditions (Morgan et al., 2013). Beyond 10Hz would be considered to be super-high-resolution FWI (Mispel et al., 2019), which goes beyond the scope of this work.

vi) The reflection coefficient time series and wavelet are convolved to produce the seismic trace $\mathcal{T}$.

vii) Fourier coefficients for magnitude $\mathcal{M}(\zeta)$ and phase $\mathcal{M}(\phi)$ are derived based on the Fast Fourier Transform (FFT).

To exploit higher dimensionality and use 2D CNNs, a secondary generator was designed to perform a Continuous Wavelet Transform (CWT). This was identical to the previous generator, expect that in Step (vii), produce a CWT with sampling frequencies from 1-75Hz and wavelet identical to the wavelet given in Step (v). This is referred to as Step (viii). The different steps for these two generator flows are shown in Figure 4.1 for a sample velocity profile. These generators will be referred to as Generator 1 and Generator 2 respectively.

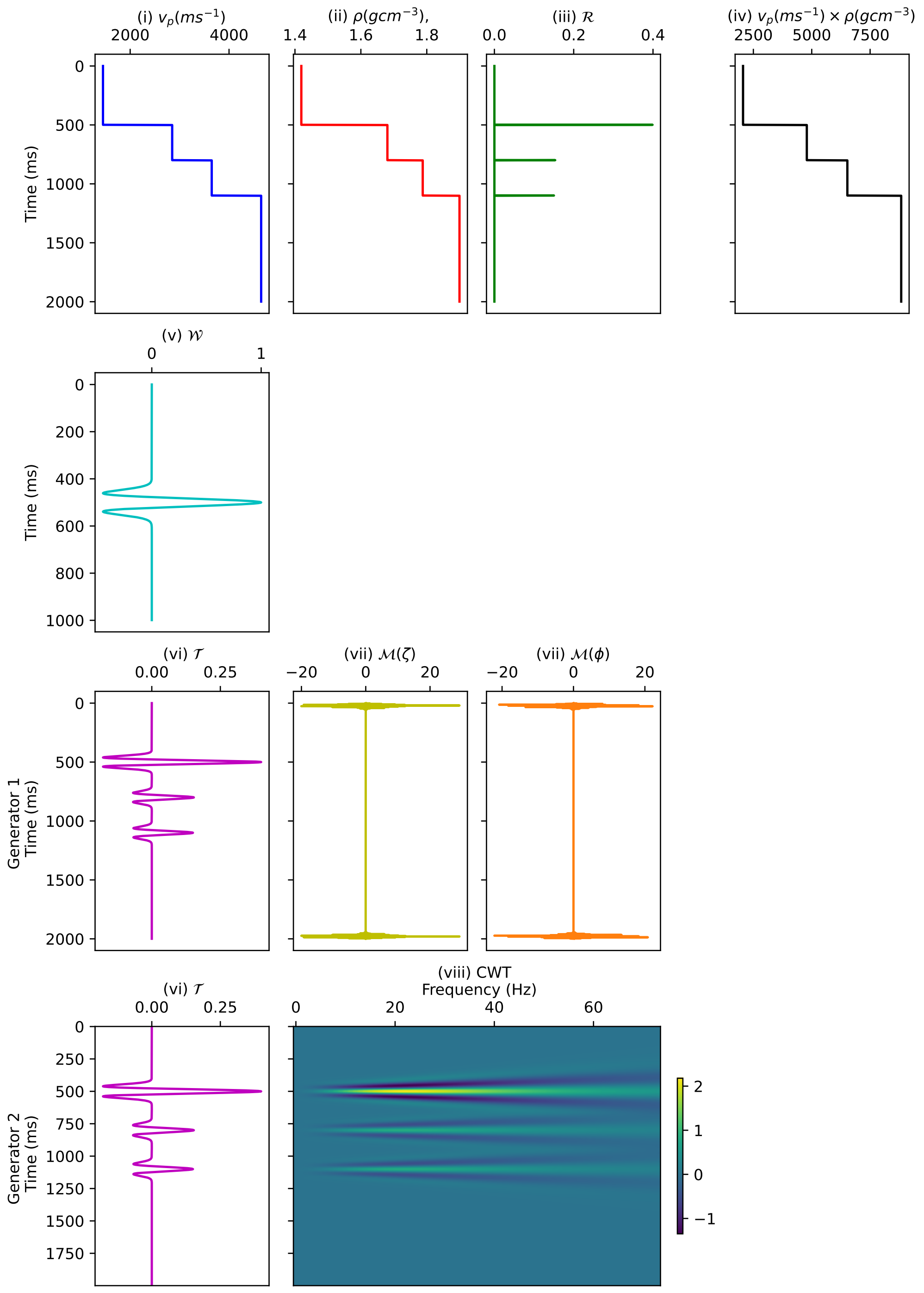


Figure 4.1: Workflow for creating a pseudo-spectral synthetic trace.

The parameters assigned with the generators are given in Table 4.1 and distribution of layers within a training run for 1,000,000 samples and 100,000 testing samples are given in Figure 4.2. The **vel_min_separation**, **time_min_separation** are two key parameters as they control the velocity and temporal resolution of the model respectively.

Parameter	Description	Value
length (ms)	Length of trace	2000
vel_min (m/s)	Minimum velocity	1450
vel_max (m/s)	Maximum velocity	5000
vel_min_separation (m/s)	Minimum velocity separation	10
time_min (ms)	Minimum time sample	500
time_max (ms)	Maximum time sample	1500
time_min_separation (ms)	Minimum time separation	2
layers_min	Minimum number of layers	1
layers_max	Maximum number of layers	4
dominant_frequency	Hz of dominant frequency	10

Table 4.1: Synthetic data generator parameters.

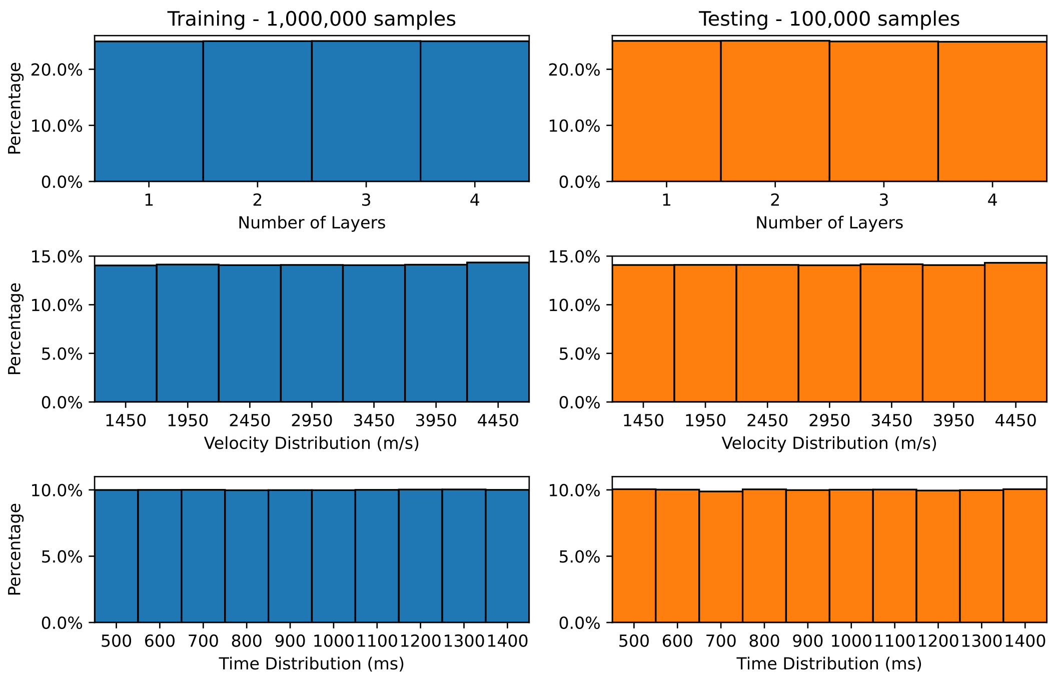
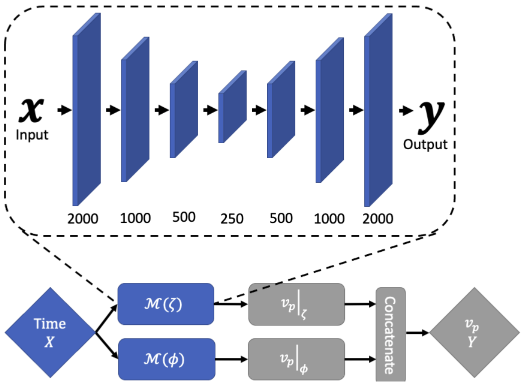


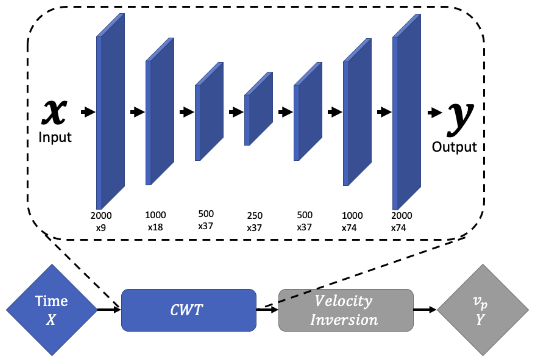
Figure 4.2: The overall statistics on the training and testing dataset.

4.2.2 | DNN Framework

Figure 4.3(a) illustrates the DNN framework used to first invert for the Fourier coefficients from the time domain and then invert for velocity profile. The complete workflow has five modules, with each module consisting of a NN with 5 fully connected hidden layers. The layer distributions consisted of an input layer of 2000 neurons, a set of 5 hidden layers of sizes 1000, 500, 250, 500, 1000 neurons, and an output layer of 2000 neurons. This hour-glass design can be considered representative of multi-scale FWI (Bunks et al., 1995) since at each hidden layer, the NN learns an abstracted frequency component of the data at a different scale. This is Indeed synonymous with modern DNN approaches such as encoder-decoders and U-Net (Ronneberger et al., 2015) and how they extract data representations (Berthelot et al., 2018; Yu and Principe, 2019). The final concatenate network learns the optimal way for combining the outputs. In total, the DNN had 25 hidden layers. In the case of the CWT pseudo-representation, we designed a similar framework to that shown in Figure 4.3(a), except that the learned CWT network has an additional dimension to be able to create the CWT. This is shown in Figure 4.3(b). The learned CWT network has layers of shape $(2000 \times 9), (1000 \times 18), (500 \times 37), (250 \times 37), (500 \times 37), (1000 \times 74), (2000 \times 74)$. The velocity inversion DNNs were built to be representatives of Conv1D, Conv2D, VGG, ResNet architectures respectively. A full architectural summary and training process for these network is provided in Appendix B.4.1.



(a) Fourier components.



(b) CWT.

Figure 4.3: Pseudo-spectral FWI DNN frameworks to invert for Fourier Transform and CWT. X is the input time domain, Y is the output v_p velocity and $\mathcal{M}$ is the Fourier domain, with magnitude ζ and phase ϕ. Each component (blue or grey box) is a network.

4.2.3 | Multi-layer Numerical Results

The first experiment was to assert the validity of the framework in a 1D synthetic case. Using Generator 1 with 1,000,000 training and 100,000 testing samples, with DNN framework A for the Fourier components, loss function was set to be the Sum of Squared Error, stabilized via fixed $L_2 - norm$ regularization, data batching, early stopping, and executed for 120 epochs (the number times that the algorithm passes through the entire training dataset). Gradient descent update was optimized via an ADAM optimizer. The DNN was implemented using Keras 2.2.4 and TensorFlow 1.13.1 backend. This was trained on an Intel i7-7800x X-series CPU workstation provided by the Department of Physics at the University of Malta.

Figure 4.4 illustrates the application of DNN architecture for a sample of unseen data and the respective inversion. Inspection of the first 750ms indicates that the DNN approach is able to reconstruct both the velocity and the waveform profile near perfectly, irrespective of the number of layers and the magnitude of the acoustic difference in this time range. Indeed, these indicate the validity of this approach. Beyond 750ms, reconstructions start suffering from slight degradation. As illustrated in the velocity reconstruction of the middle figure, the inaccuracy is minimal and ranges $\pm 100\text{ms}^{-1}$. This leads to perturbations in the reconstruction and does not allow for perfect matching. Further inspection suggests that the main source of error is due to the magnitude component of the network (red).

Figure 4.5 shows the DNN mean squared error performance over the different epochs per DNN component. This graph indicates that the network is learning since mean squared error is overall decreasing at each epoch. The drastic decreases in the mean squared error at different epoch levels can be attributed to the step-wise reductions in learning rate shown in Figure 4.5(c). This varying learning rate allows the network to move to a deeper optimization level and approach a more global minima for the optimization problem. Interestingly, this performance plot indicates that the technique might suffer from a compounding error issue. The two best performing components are the first layer of learning for the inversion, namely Time-to-FFT-Magnitude and Time-to-FFT-Phase, as their mean squared error performance plateaus are at 10^{-1}. In the second phase of the inversion from the respective FFT components to velocities (FFT-Magnitude-to-Velocity and FFT-Phase-to-Velocity) error plateaus are at 10^{1}, which is two orders of magnitude greater. The final network component sits even higher on the scale at 10^{2}.

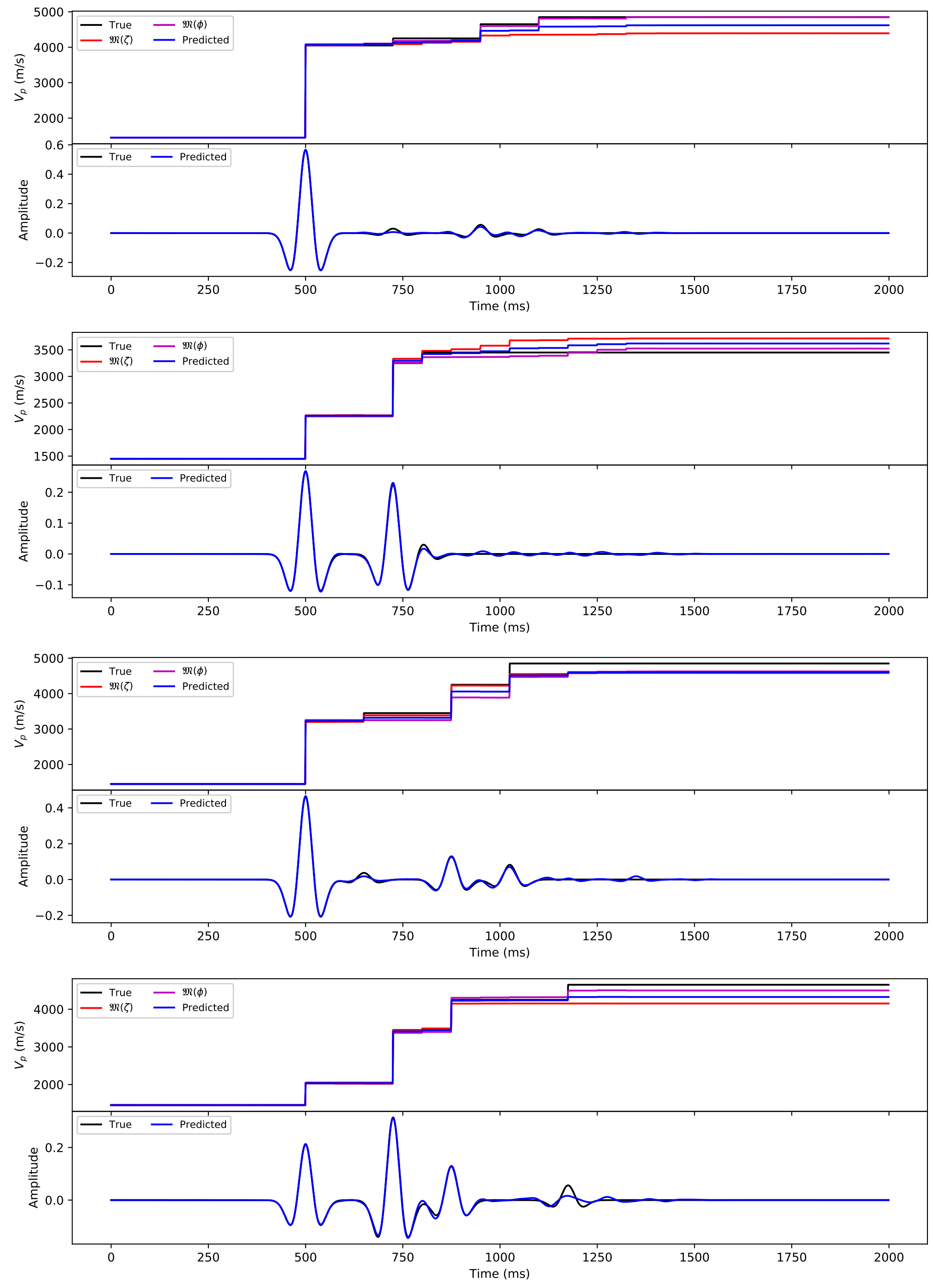
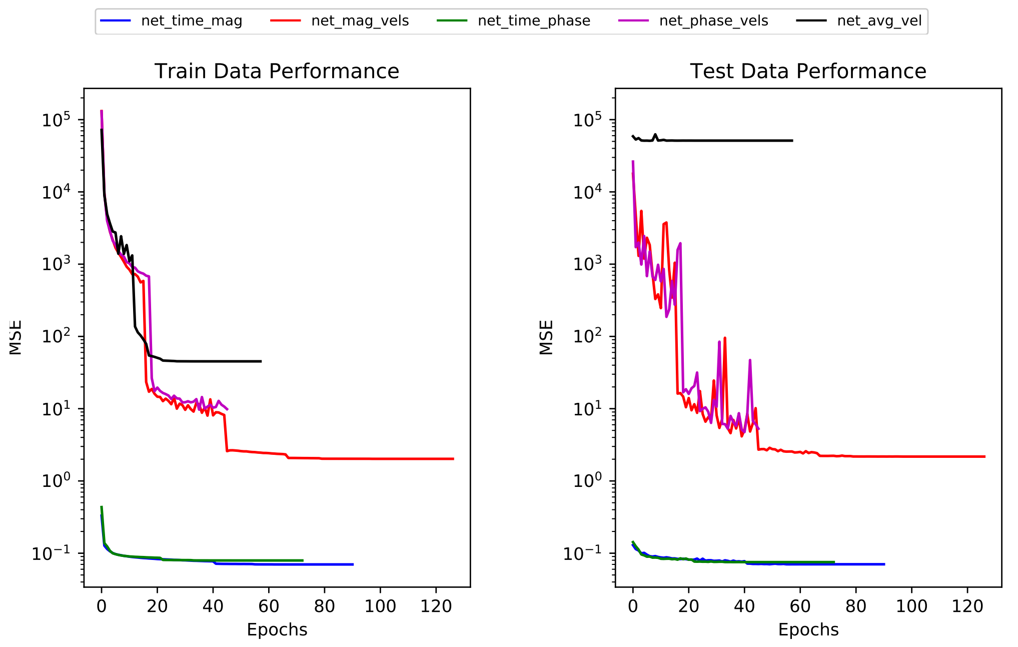
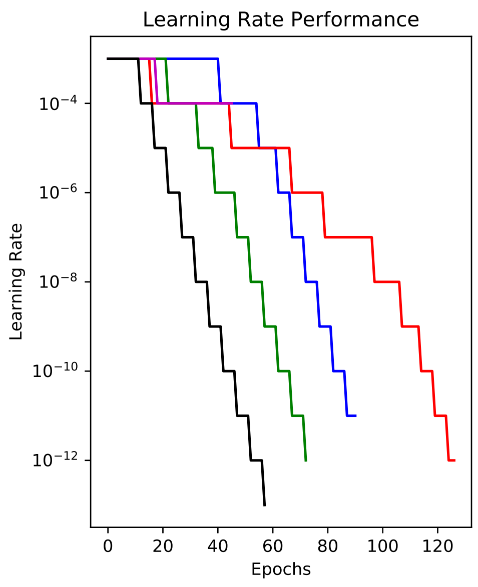


Figure 4.4: Four different predictions obtained from learned weights of the DNN on unseen data. The top panels are the velocity profile reconstructions from the two NN architecture branches ($\mathcal{M}(\zeta)$ and $\mathcal{M}(\phi)$) and the combined result. Bottom panels are the observed and inverted waveforms.



(a) Training dataset MSE over the different epochs per DNN component. Overall performance is decreasing per epoch, indicating that the DNN is learning to invert.

(b) Test dataset MSE over the different epochs per DNN component. Overall MSE is decreasing per epoch and there are not signs of over-fitting.



(c) Learning Rate performance over the different epochs per DNN component.

Figure 4.5: DNN training performance metrics.

4.2.4 | Pre-Processing

In classical DNN approaches, it is best practice to normalise or standardize the dataset. An experiment was executed to assess what would happen with and without normalisation of the data for the Magnitude component architecture shown in the previous section. 10,000 training and 1,000 validation traces were generated using Generator 1, stored in memory so the only variance will be the scaling and trained for 200 epochs with early stopping and reducing learning rate monitor. The compute and memory resources necessary for this test were small enough such that this experiment was executed on a 2.6 GHz 6-Core Intel Core i7, 16GB RAM personal computer. The scaling approaches considered are the Standard Scaler and Min-Max Scaler. These are defined as:

$$x_{MM} = \frac{x - x_{MIN}}{x_{MAX} - x_{MIN}}, \tag{4.1}$$

$$x_{SS} = \frac{x - \mu}{\sigma}, \tag{4.2}$$

where x_{MM}, x_{SS} are the scaled values for Min-Max and Standard Scaler respectively, x_{MIN}, x_{MAX} are the minimum and maximum values of the data, μ is the mean and σ is the standard deviation of the training samples.

The mean square error for the dataset with-out and with processing was evaluated and is shown in Table 4.2. The value of the mean squared error indicates that pre-processing in the form of normalisation or standardization should not be applied to the problem dataset. The impact of the pre-processing on the inverted velocity profiles is shown in Figure 4.6. In either case, Min-Max scaling was the worst performant, only able to reconstruct the first layer at 500ms.

Pre-Processing	Mean Square Error
No Normalisation	4,041
Min Max Normalisation	296,672
Standard Scaling	17,653

Table 4.2: Quantitative assessment on the impact of pre-precessing

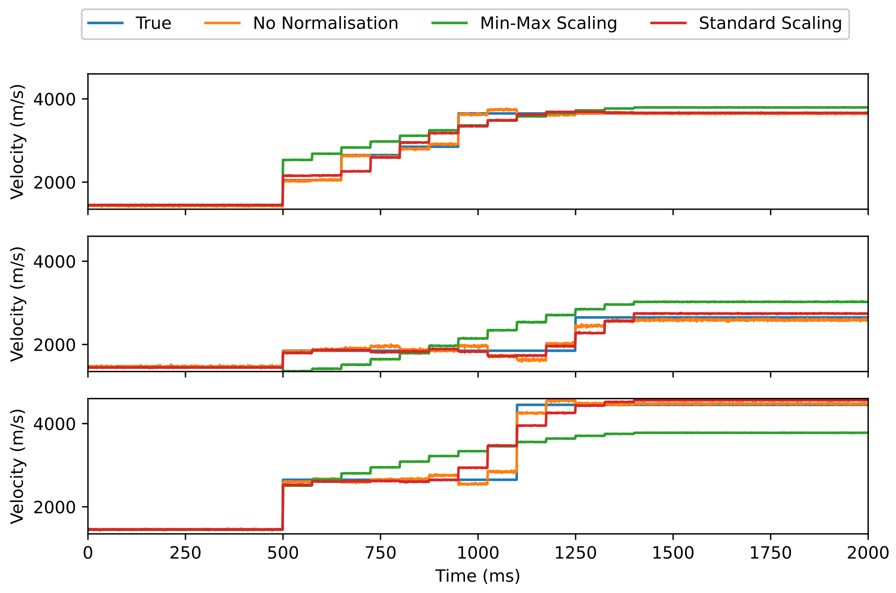


Figure 4.6: Comparison of velocity profiles with and without scaling.

4.2.5 | Architecture Comparison

In § 4.2.3, the validity of the approach was assessed and pitfalls identified for a simple example. This was extended to identify the ideal configuration in terms of architecture, loss, time-to-train and over-fitting. The following computation was made using Python 3.7 and Tensorflow 2.0.0 with a Keras backend. It was executed on an NVIDIA Titan V Graphical Processing Unit with 5120 cores and 12GB ram provided in collaboration with Dr. Carlo Giunchi at Istituto Nazionale di Geofisica e Vulcanologia at Pisa.

The conversion from time trace to pseudo-spectral representation was fixed for all 1D and 2D networks such that the comparison was done on only the different inversions architectures. This architecture given as “Time to Pseudo-Spectral 1D” or “Time to Pseudo-Spectral 2D” can be found in Appendix B.4.1. Figure 4.7 gives a comparison of different DNN architectures, loss optimizers, duration of training and validation curves. The networks were trained for the same number of epochs without early stopping. The training and validation data consisted of 1,000,000 and 100,000 generated traces using Generator 1 and Generator 2 respectively. The loss was fixed to be the MSE and lr represents the learning rate on a secondary axis in Red.

Considering all loss curves, the best performing setup is that for Conv2D and RMSprop with 20,000 loss and worst performant are MLP-Adam and Conv2D-Adam. The deeper more complex Conv2D, VGG and ResNet architectures in general seem to be experiencing some under-fitting due to the parallel, non-convergent training-validation curves. This would be indicative that the complexity in the current problem is not high and less complex network type such as Conv1D would be more suitable. As evidenced by the gradual monotonic decrease in validation curves, none of the architecture-loss optimizer combinations experienced over-fitting. Conv1D-Adam and MLP, Conv1D and Conv2D for RMSprop seem to indicate increases between epochs 2 and 10 on the validation curves. This would be symptomatic to over-fitting; however, the learning rates move to lower orders of magnitude. This enables networks to continue training and move to lower orders of loss value. The lr for some of the combinations is remaining unvaried, namely Conv1D-Adagrad, Conv2D-Adagrad and Conv2D-Adadelta, all of VGG and ResNet-Adadelta and ResNet-Adam. This is indicative that these networks are not close to reaching a global minimum and would benefit from training for more epochs. Indeed, this would indicate that more complex architectures such as VGG and ResNet require longer training epochs. Indeed, the loss-validation curves further highlight this as, in general, they do not plateau. The more complex and deeper the architectures required longer training times. MLP averaged training time of 41.5 compute hours, whereas ResNet averaged 117 compute hours. These performance metrics should not be considered in isolation and visual inspection of inversion should be equally assessed.

Figure 4.8 show two sample traces inverted for all these architecture and loss optimizer combinations. Upon initial qualitative inspection, it is clearly evident how Conv1D is the superior architecture. This further confirms the previous assertion that for this given experiment, deeper and more complex architecture types such as Conv2D, VGG and ResNet are not necessary and can be detrimental to overall performance. Common to all architectures is a ringing effect on the time trace inversion. This is due to inadequate inversion for the velocity profile where the initial velocity increase is identified correctly, but beyond this, there is a step-wise incremental velocity profile. This is present in simple MLP and more complex DNNs. Conv1D is the only architecture type which is symptom free.

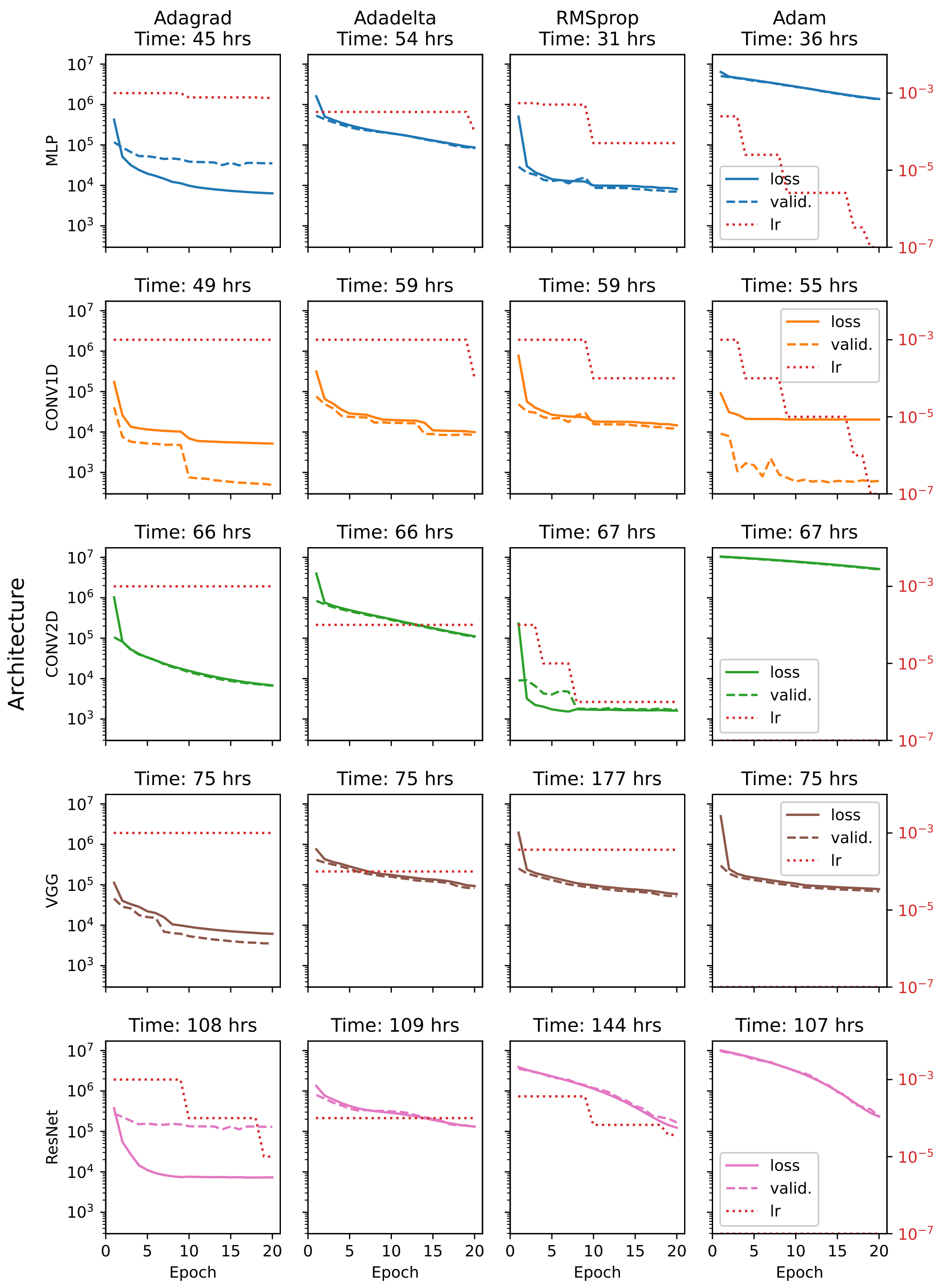
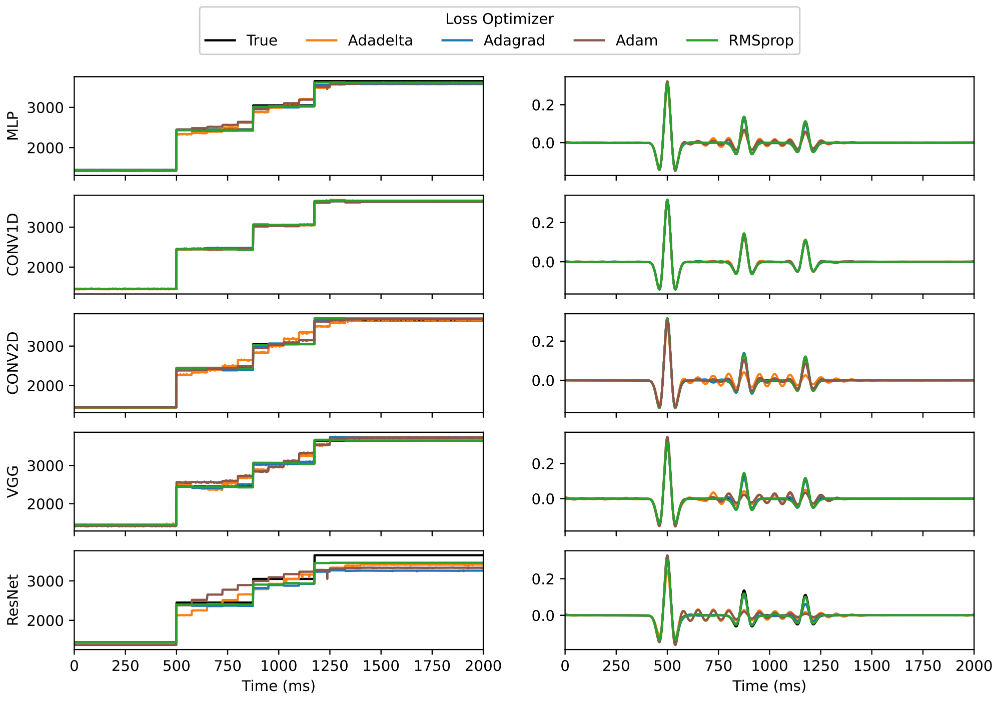
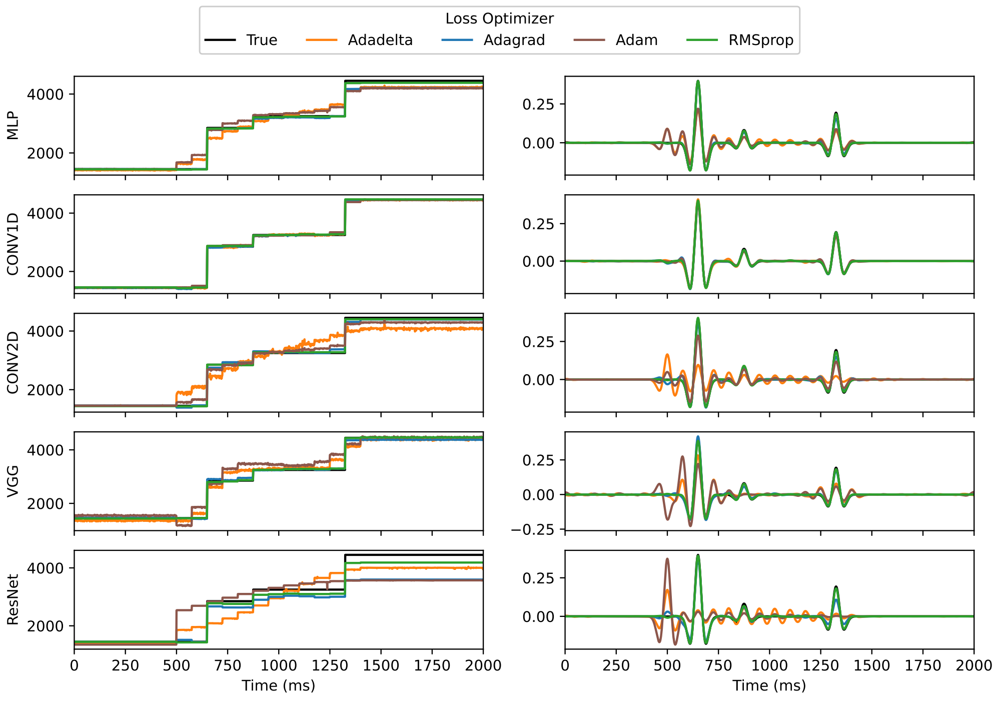


Figure 4.7: Normalised comparison of DNN architectures, loss optimizer, duration of training and validation curves. The networks were trained for the same number of epochs without early stopping. The loss is the MSE and lr is the Learning Rate.



(a) Trace A



(b) Trace B

Figure 4.8: Velocity inversion for different DNN architectures and losses.

4.2.6 | Architecture-Loss Combination

All results are summarised in Table 4.5 to quantitatively assess the inversion process and the DNN performance metric. The evaluation criteria are:

- Duration (d): Duration of training
- Train (t): Lowest MSE within training
- Validation (v): Qualitative assess of under-fitting/over-fitting and learning rate performance
- Inversion (i): RMSE of 100,000 validation velocities as compared to true velocity

These criteria are ranked from 1-20, with 20 being the best result. The score was calculated as

$$\text{Score} = d + t + v + 2i. \tag{4.3}$$

The formula is arbitrarily chosen and linear in nature, making ideal for interpretation and understanding. The additional weight of 2 for the inversion rank emphasizes the inversion is the most important criteria. The rank criteria ranks all scores, with the highest score being best. The best performing architecture-loss combination is identified as **Conv1D-Adadelta**. Table 4.3 and Table 4.4 summarize Table 4.5 per architecture and loss optimizer respectively. Table 4.3 further reinforces the choice for ideal setup being of type Conv1D since this architecture ranked in the top four, irrespective of Loss Optimizer. Table 4.4 is in agreement that Adadelta is the better loss optimizer for our setup, however the difference is relatively small and not substantial. Choosing a different loss optimizer would not result in deterioration of our result. Full results used to build these tables is given in Appendix B.4.

Architecture	Score		
	Avg	Min	Max
Conv1D	74.3	70	79
MLP	63.8	61	66
Conv2D	51.8	48	56
VGG	48.5	34	55
ResNet	33.5	28	42

Table 4.3: Quantitative assessment for architectures.

Architecture	Score		
	Avg	Min	Max
Adadelta	58.0	42	79
Adagrad	54.6	31	70
RMSprop	54.4	33	72
Adam	50.4	28	76

Table 4.4: Quantitative assessment for loss optimizers.

Architecture	Loss Optimizer	Duration	Train	Validation	Inversion	Score	Overall Rank
MLP	Adagrad	18	4	20	11	64	6
MLP	Adadelta	16	13	19	8	64	6
MLP	RMSprop	20	7	9	15	66	5
MLP	Adam	19	19	9	7	61	8
Conv1D	Adagrad	17	2	15	18	70	4
Conv1D	Adadelta	14	8	19	19	79	1
Conv1D	RMSprop	14	9	9	20	72	3
Conv1D	Adam	15	10	17	17	76	2
Conv2D	Adagrad	12	5	15	12	56	9
Conv2D	Adadelta	12	15	15	4	50	14
Conv2D	RMSprop	10	1	9	14	48	15
Conv2D	Adam	10	20	3	10	53	11
VGG	Adagrad	8	3	15	13	52	13
VGG	Adadelta	8	14	15	9	55	10
VGG	RMSprop	1	11	9	16	53	11
VGG	Adam	8	12	4	5	34	17
ResNet	Adagrad	4	6	17	2	31	19
ResNet	Adadelta	4	17	15	3	42	16
ResNet	RMSprop	2	16	3	6	33	18
ResNet	Adam	5	18	3	1	28	20

Table 4.5: Quantitative assessment for architecture and loss optimizers.

4.2.7 | Marmousi Model

4.2.7.1 | Dataset

The Marmousi-2 model (Martin et al., 2002) was used to evaluate the technique on an industry standard dataset. Figure 4.9 and Figure 4.10 illustrate the Marmousi-2 model and velocity profile respectively. The model has a lateral extension of 17 km and a depth of 3.5 km and includes a total of 199 layers geophysical layers, as well as an extended water layer of 450 m depth to simulate a deep-water setting. The grid spacing was 10m vertically by 25m laterally, resulting in a 2801 by 13601 grid. The velocity in the model ranges from 1500ms^{-1} up to 4700ms^{-1} and after the application of a 150m vertical median filter to reduce the vertical resolution, the number of layers in each velocity profile was analytically calculated to range between 20 to 50 layers. The salt density as taken constant throughout. The generation for this model is provided in Appendix B.2. This will be referred to as the Marmousi model for the rest of the thesis.

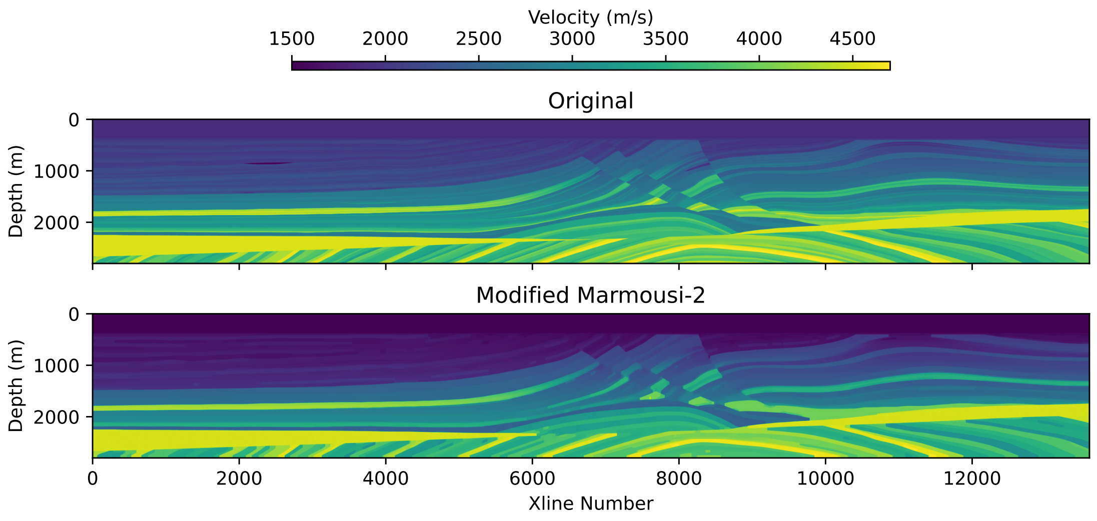


Figure 4.9: Marmousi-2 and modified Marmousi-2 velocity model.

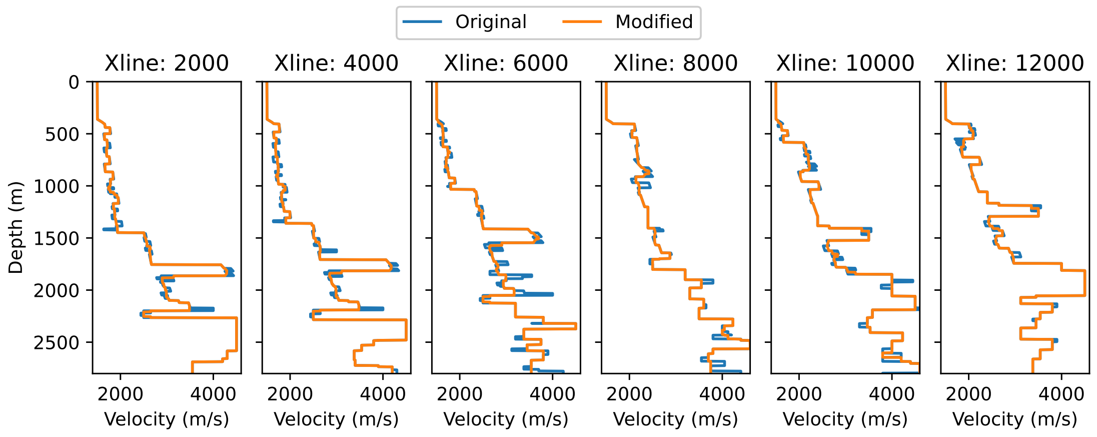


Figure 4.10: Velocity profiles through crosslines (Xlines) on Marmousi-2 and modified Marmousi-2 models.

4.2.7.2 | DNN FWI Generator

Following from the work in Sections 4.2.1- 4.2.6, a generator was constructed to be able to invert for the Marmousi model. The generator parameters are given in Table 4.6. A sample of the velocity, trace and CWT generated by this generator are available in Figure 4.11.

Parameter	Description	Value
length (ms)	Length of trace	2801
vel_min (m/s)	Minimum velocity	1450
vel_max (m/s)	Maximum velocity	5000
vel_min_separation (m/s)	Minimum velocity separation	15
time_min (ms)	Minimum time sample	0
time_max (ms)	Maximum time sample	2801
time_min_separation (ms)	Minimum time separation	5
layers_min	Minimum number of layers	20
layers_max	Maximum number of layers	50
dominant_frequency	Hz of dominant frequency	5

Table 4.6: Marmousi data generator parameters.

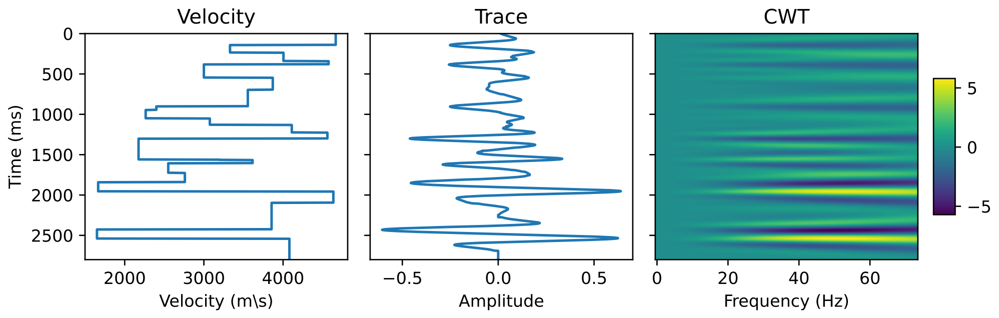


Figure 4.11: Sample velocity profile, trace and CWT generated by Marmousi generator.

4.2.7.3 | DNN Training and Architecture Performance

The network was trained for 30 epochs, at 1,000,000 traces and 100,000 traces per training and testing dataset respectively. The network was a slightly modified version of the ideal network in § 4.2.6 due to the longer time length of trace. This network is given as two part network "Marmousi - Time to Pseudo-Spectral" and "Marmousi - Pseudo-Spectral to Velocity" in Appendix B.1.

As expected from previous work, the DNN training performance shown in Figure 4.12 indicates how this workflow performs well with a monotonically decreasing loss per epoch, with no symptoms of over-fitting or under-fitting. From the Learning Rate plot, the DNN might benefit from a couple more additional training epochs since the automatic reducing learning rate callback within the network was never initiated. Figure 4.13 reinforce this suggestion of good training and with additional metrics calculated per epoch of Explained Variance and R2 Score, both of which are gradually approaching one per epoch.

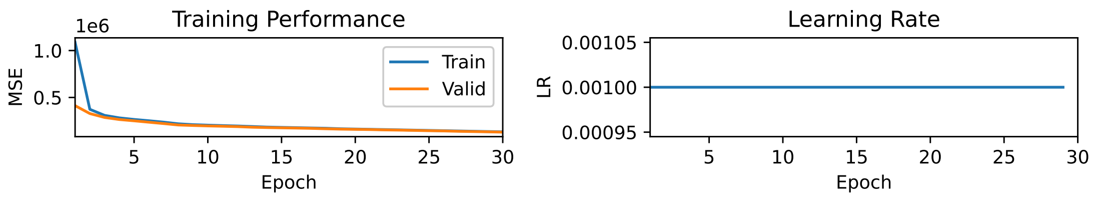


Figure 4.12: Training and Validation curves for DNN training and Learning Rate values.

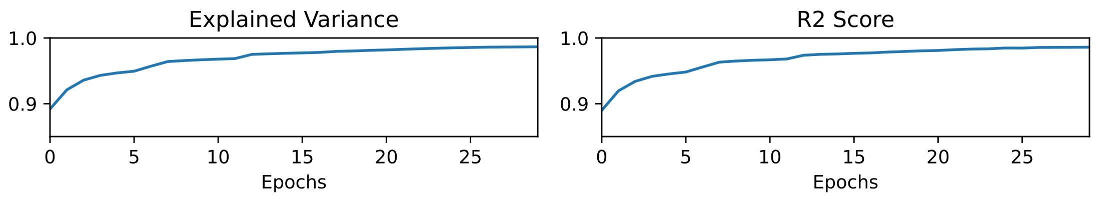


Figure 4.13: Explained variance and R2 Score metrics calculated per epoch. Both these metrics are approaching one per epoch, suggesting a good overall DNN performance.

Figure 4.14 show histograms for the evolution of network trainable parameters per epoch. The y-axis of these plots are the epochs and the darker saturation indicate the older the epoch value. The z-axis is the density of values represented at x-axis. The first 6 rows show histograms for the convolution and batch normalisation layers as groups per row. Each group is composed of Convolution – Convolution – Batch Normalisation layers which is getting repeated 6 times in the network. For each convolution, the trainable parameters are the bias and kernel, whilst for Batch Normalisation γ and β are the per epoch standard deviation and mean respectively, moving variance and moving mean. Convolution kernels are relatively flat, with narrow distribution from $[-0.15, 0.15]$ to $[-0.04, 0.04]$. This indicates that the input signal is getting collapsed into smaller probabilities for explanation. Comparing convolution kernels with each row, these are very similar. They both have wider distributions indicating that more abstractions are being included with each pass. The bias starts off with multiple peaks then centre about 0, moving from a left skewed to a more gaussian distribution. This shows that the initial layers are identifying multiple parts of the input signal as being important due to the multiple peaks, but as we go deeper into the network with more epochs, we start seeing that the network "understanding" starts to converge into a gaussian distribution and collapsing all the abstracted information from the upper layers into the desired output. The last row shows histograms for the two dense layers, with trainable parameters for bias and the kernel. The kernel distribution for both dense layers is centred around zero, with a very narrow standard deviation. This indicates how the last two layers a selectively choosing components from the different abstracted feature maps and adding small components to build up the final inversion. The shifting bias from left skewed to more Gaussian, with mean close to 0.004, indicates that the final reconstruction is happening within a Gaussian environment for the DNN. This is an ideal setup as this will facilitate regularizations to unseen data.

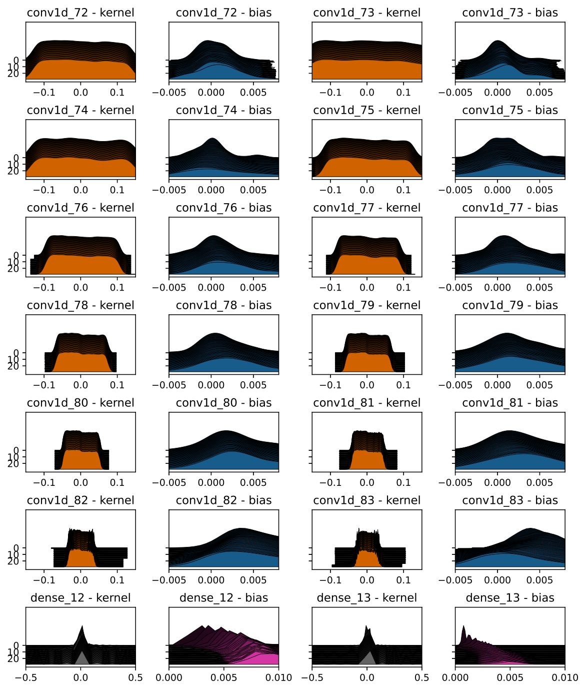


Figure 4.14: Evolution of network histograms. The depth (y-dimension) of these plots are the epochs and the darker saturation indicate the older the epoch value. The z-dimension is the density of values represented at x-dimension. MM is the moving average and MV is moving variance.

Figure 4.15 and Figure 4.16 show the velocity inversion and resultant trace for Xline 2000, 8000 and 12000 from that Marmousi model respectively for every training epoch for the DNN. From the initial epoch, the DNN was able to invert most of the information within the velocity profile. The inversion is not perfect, since there is a form of leakage/spikes coming in up to about epoch 10. This is the learning-process of the DNN, since this gets gradually removed from the velocity profile with additional epoch, and at about epoch 20 there is an almost perfect reconstruction. From epoch 20 to epoch 30, the differences are minimal as can be seen by the very small changes to the MSE shown in the plot and as well in the overall DNN loss values in Figure 4.12.

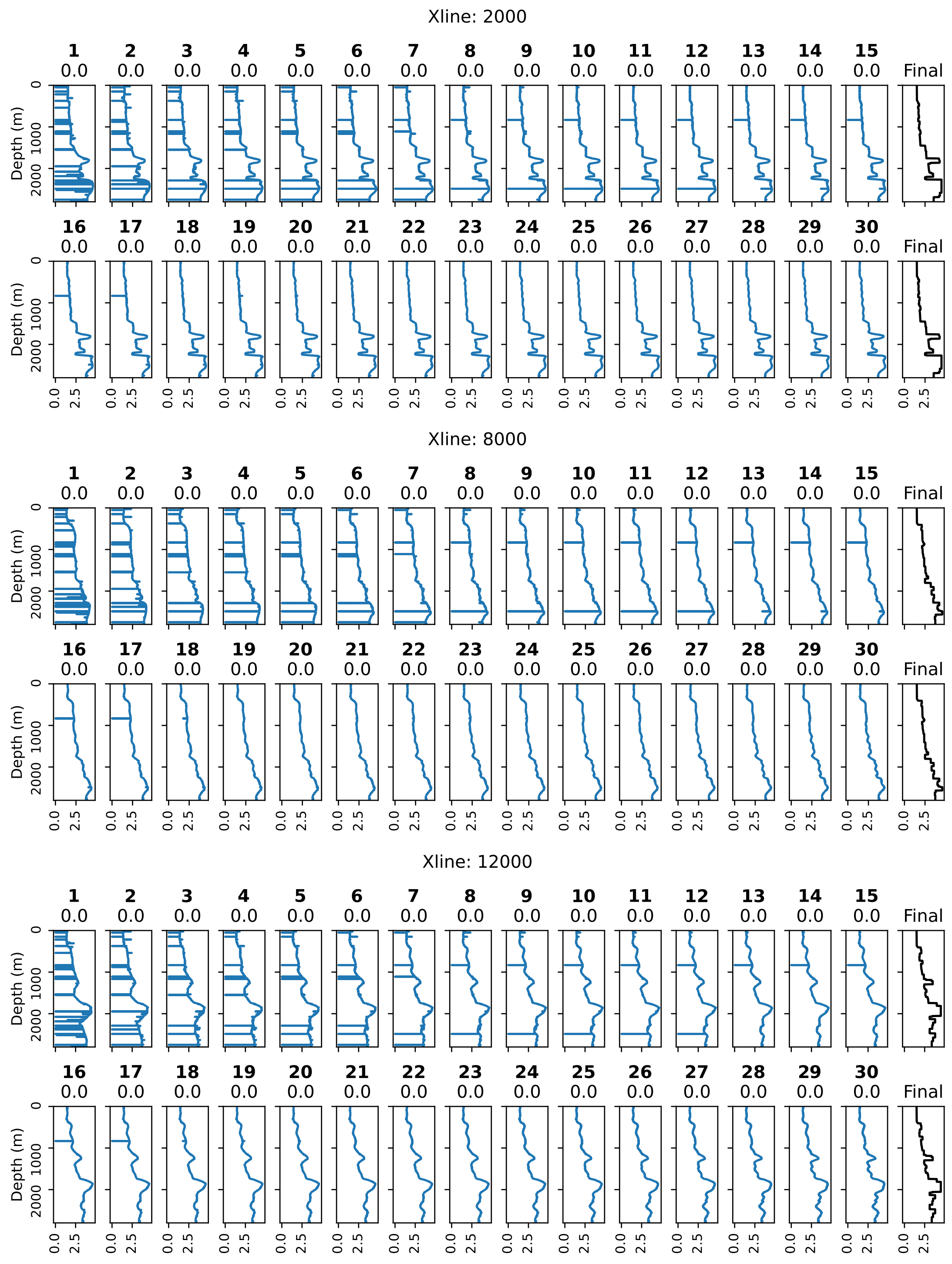


Figure 4.15: Evolution of velocity profile for Xline 2000, 8000 and 12000 through the different epochs respectively. Above each plot, there is the Epoch number in bold, and the MSE.

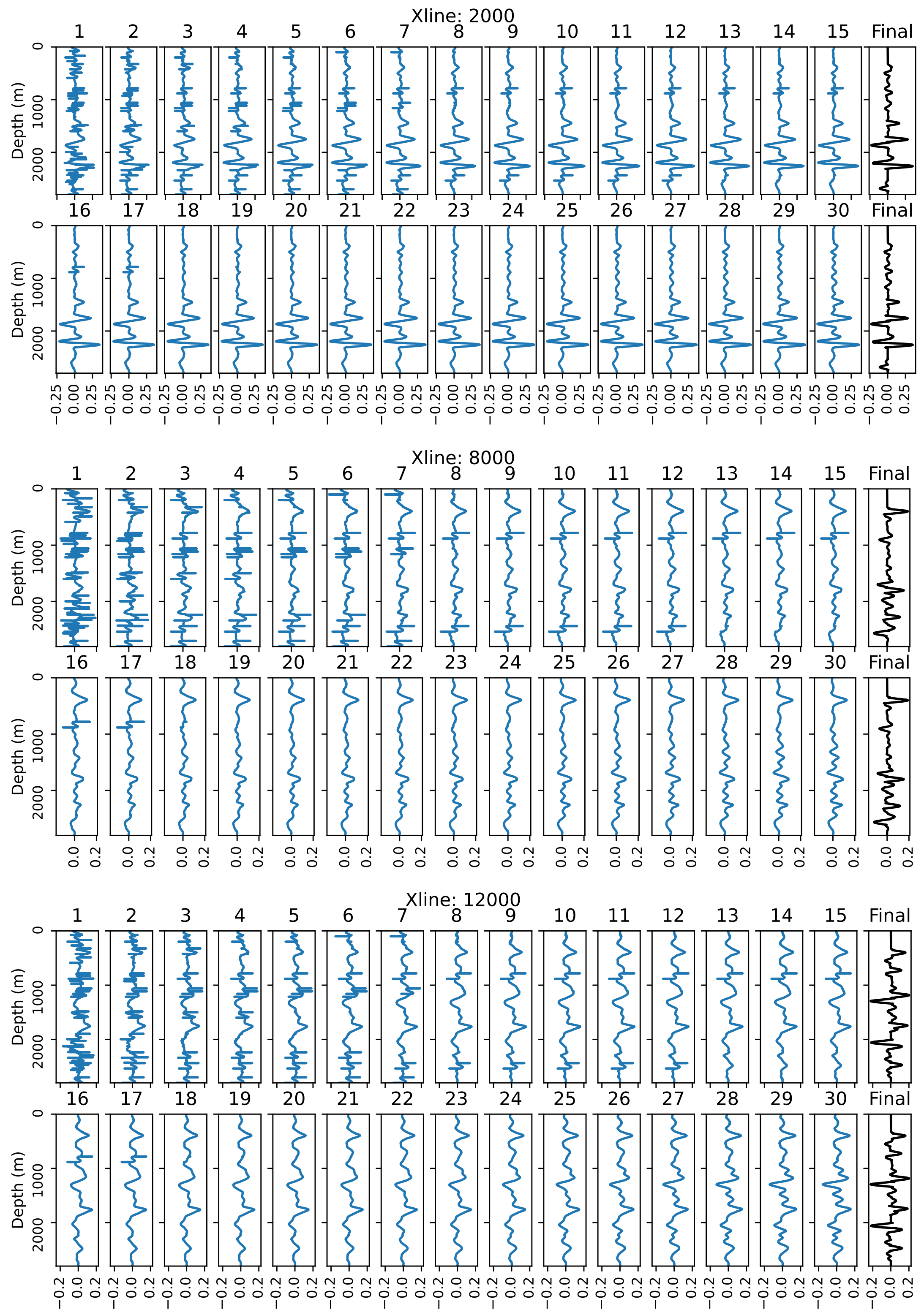


Figure 4.16: Evolution of trace for Xline 2000, 8000 and 12000 through the different epochs respectively.

4.2.7.4 | Deterministic FWI

Classical FWI with Sobolev space norm regularization was employed for comparative purposes – see Figure 4.17 and Figure 4.18. This was a modified version of the FWI optimization framework provided by Kazei and Ovcharenko (2019). The maximum frequency of the inversion process was set to be 3.5Hz. This results in a minimum update resolution of 414m given by $\lambda = \frac{v_{min}}{f_{max}}$ where v_{min} is 1450ms^{-1} and f_{max} is 3.5Hz. The dataset was resampled and interpolated by a factor of 10 to enable a faster implementation and still retain the maximum update resolution. The iterative update process started from frequency 1Hz and iteratively updated by a factor of 1.2 until reaching a maximum frequency of 3.45Hz. The optimization algorithm was L-BFGS-B (Zhu et al., 1997), with 50 iterations per frequency band in each update. Forward shot modelling was done every 100m, starting from 100m offset, and receivers spaced every 100m. The detailed implementation and inversion parameters are provided in Appendix B.3. More advanced FWI code could have been implemented to improve lateral continuity and imaging, however results obtained by this implementation provided acceptable runtimes and results, thus making feasible for our experimentation. Examples of state-of-the-art code which usually available for consortiums are FULLWAVE[1] or CREWES[2].

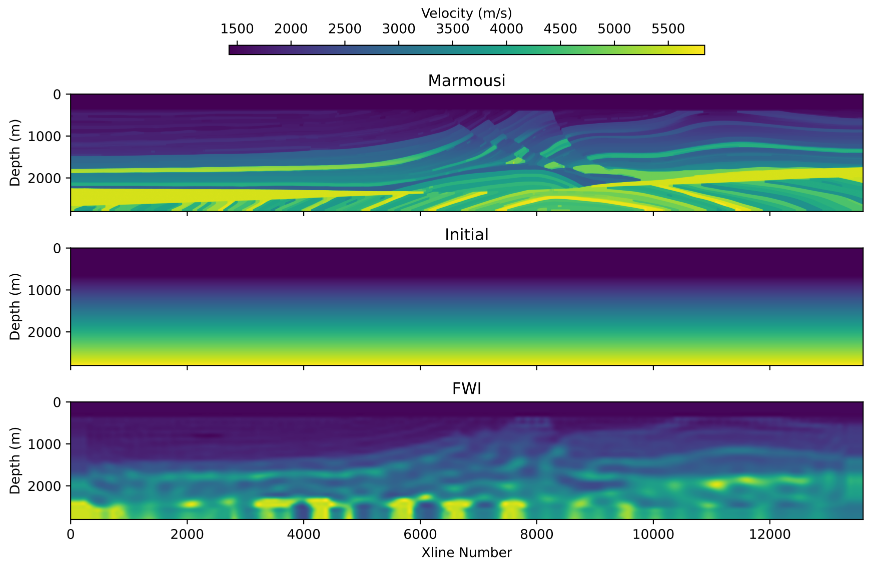


Figure 4.17: Classical FWI with Sobolev space norm regularization result. Top: Initial modified Marmousi model. Middle: Initial velocity provided for FWI. Bottom: FWI result following from inverting for 3.6Hz.

[1]`https://fullwave3d.github.io/`

[2]`https://www.crewes.org/ResearchLinks/Full_Waveform_Inversion/`

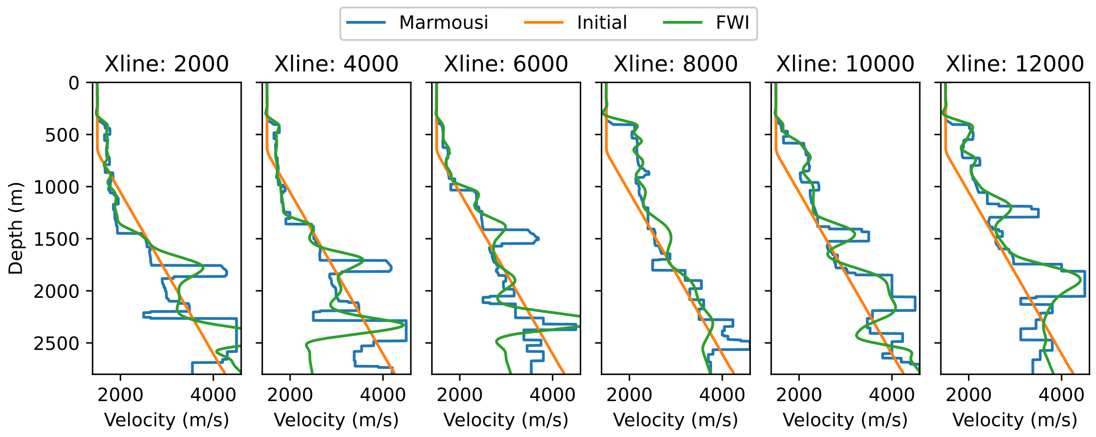


Figure 4.18: Velocity profiles through Xlines on Marmousi, Initial and FWI results as shown in Figure 4.17.

4.2.7.5 | DNN and Classical FWI

To evaluate the performance of our DNN approach, classical FWI and the DNN FWI approach are compared in Figure 4.19. Off the start, it is clearly evident how the DNN approach is producing a lot more uplift than the standard approach. There is improved imaging in the sediment layers, with distinct layers being reconstructed which would otherwise be missed with classical FWI – Zoom 1 in Figure 4.20. The middle section, with the heavily over-trusted layers shown in Zoom 2, the velocity layers are also being reconstructed to good levels and the small sedimentary pockets at the pinch of the over-thrust are being to be imaged as well. These are being missed completely in Classical FWI. Sub-salt in Zoom 3, DNN is once again producing much better imaging up to the salt and below the salt. Indeed, sub-salt, we are starting to image partially some of the layer coming up into the salt. The inversion process is not perfect as shown by the differences in velocities in Figure 4.22 for either of the three zoomed sections. Comparison of the error maps, the problematic areas of DNN are also those for classical FWI. In Zoom 1, the amplitude of the large velocity layer coming in at 1400m depth is not being inverted properly. The onset of this layer is not as problematic, but leakage is evidently present. Similarly, for Zoom 3, the salt arrival at 2200m depth, is being imaged by DNN and not by FWI. In Zoom 2, the error hotspot for DNN are similar to those of FWI, however the magnitude of the error is of an order different. This would indicate that DNN is very good performant when it comes to inverting for large velocity packages. Figure 4.21 gives the amplitude spectra for the full and zoomed velocity models respectively. This show that the frequency content is similar in either approach, yet both are lower than the true.

The velocity profiles (Figure 4.22) and trace reconstruction (Figure 4.23) confirm that our DNN approach inverted more of the signal than classical FWI. Upon closer investigation, we are seeing small spikes on the velocity on the salt section of Xlines 2000, 4000 and 6000. Further training would potentially mitigate this, or a median filter could be applied post-inversion to resolve this. From the velocity profiles, we see how FWI is able to update the shallow sections up to 1400m really well, potentially better than DNN as it is able to identify a velocity inversion at depth 500m on Xline 8000 and a pronounced segment layer at depth 800m on Xline 12000. However, beyond 1400m depth, the geometry and forward-modelling physical constraints from ray-tracing come into play and FWI is unable to provide more uplift at deeper velocity packages.

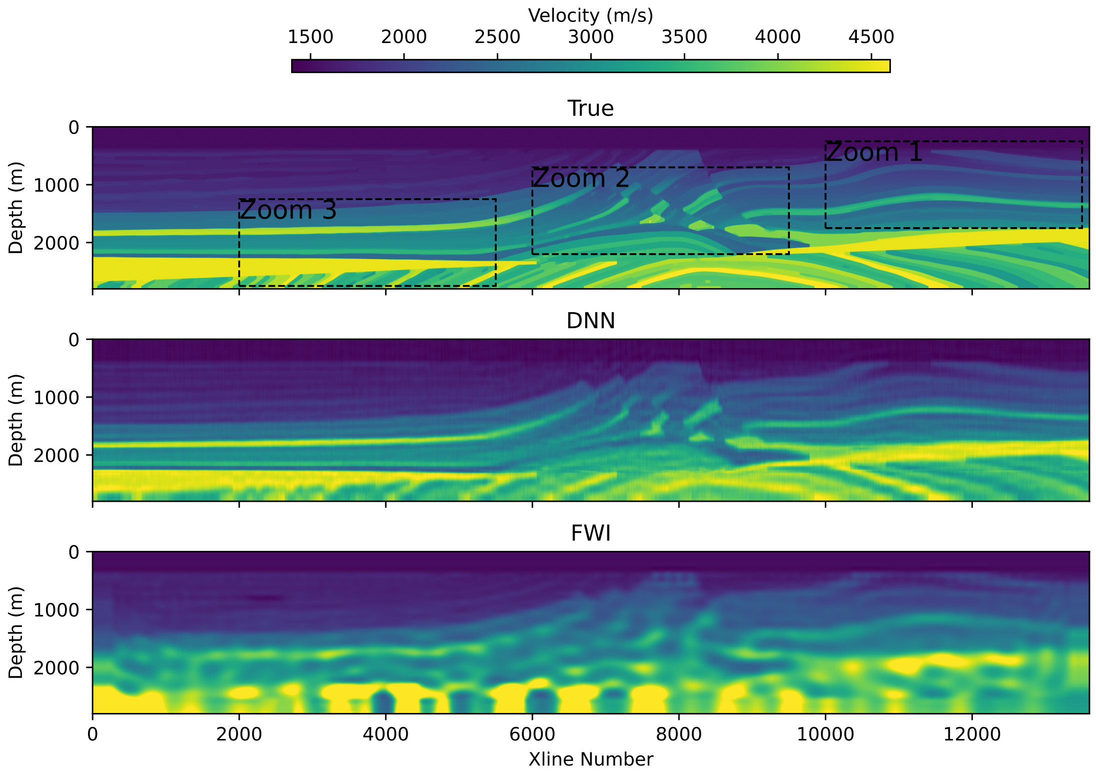


Figure 4.19: Comparison of DNN and Classical FWI reconstructed velocity models. Top: Initial Marmousi model with highlighted Zoom 1-3 used in Figure 4.22. Middle: DNN FWI result. Bottom: Classical FWI result following from inverting for 3.6 Hz.

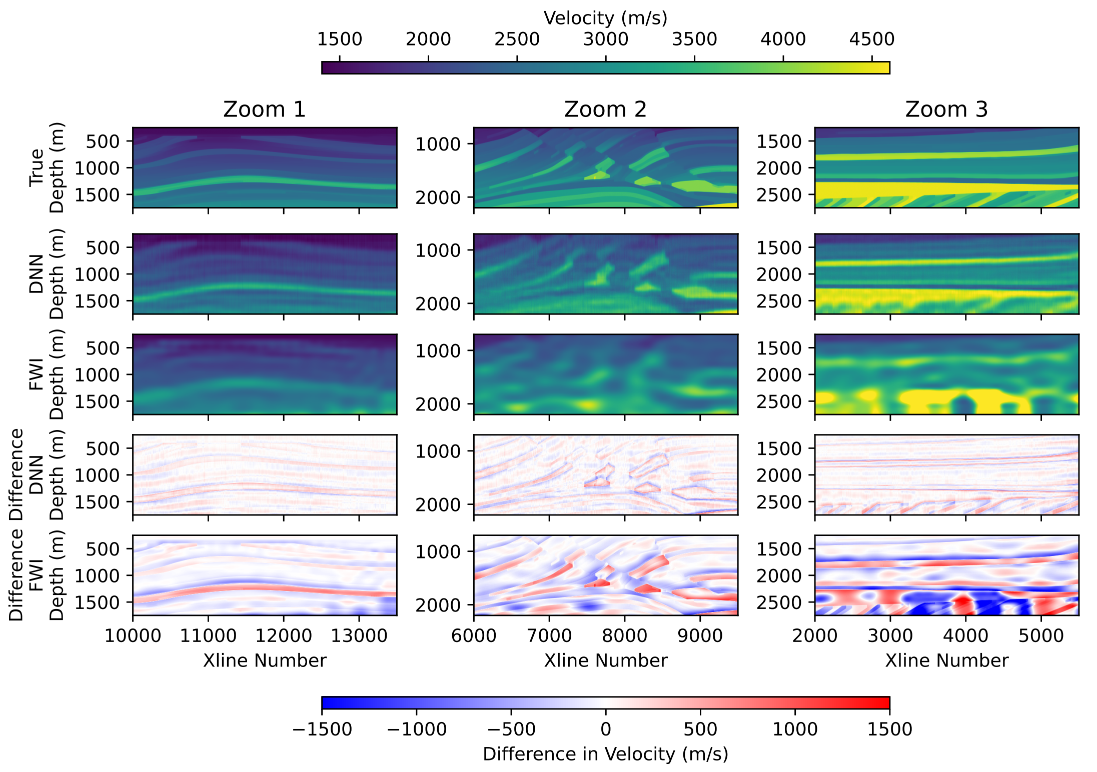


Figure 4.20: Zoomed comparison of DNN and Classical FWI reconstructed velocity models and corresponding errors.

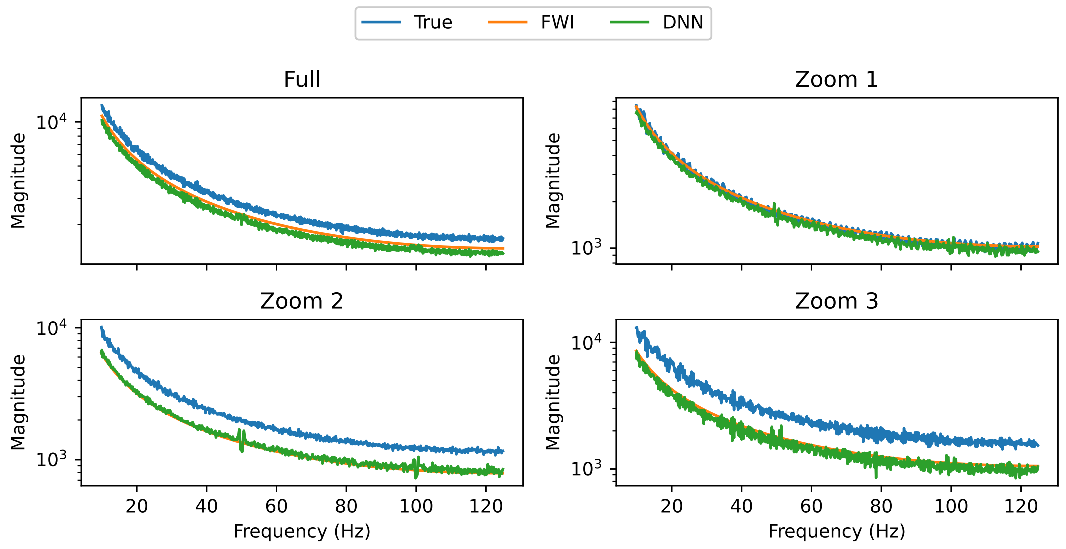


Figure 4.21: DNN and Classical FWI amplitude spectra show that the frequency content is similar in either approach, yet both are lower than the true.

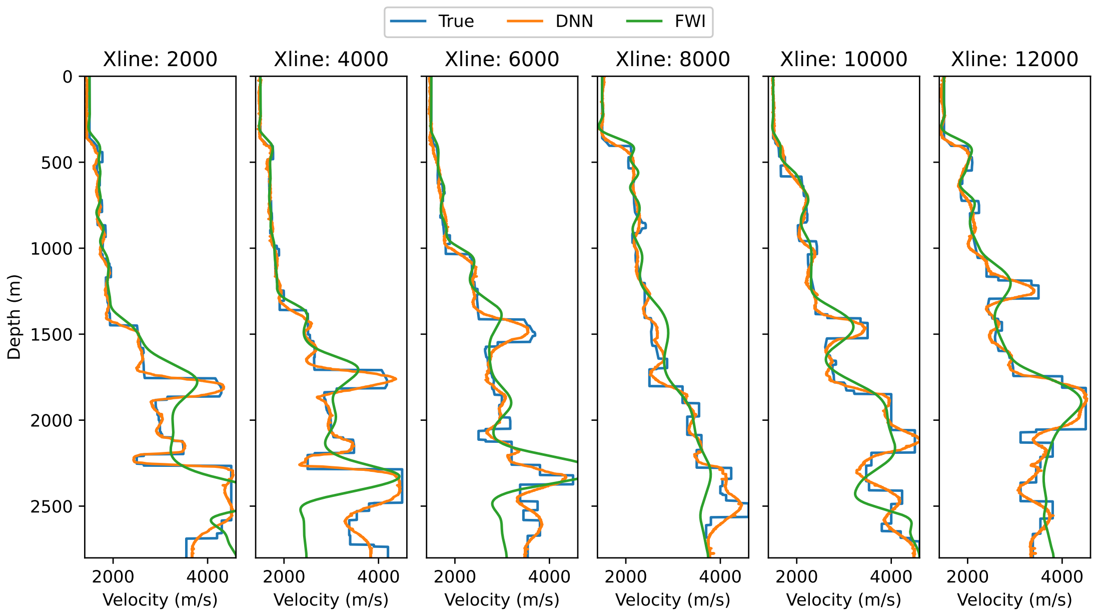


Figure 4.22: Velocity profiles through Xlines on Initial, DNN and FWI results as shown in Figure 4.19.

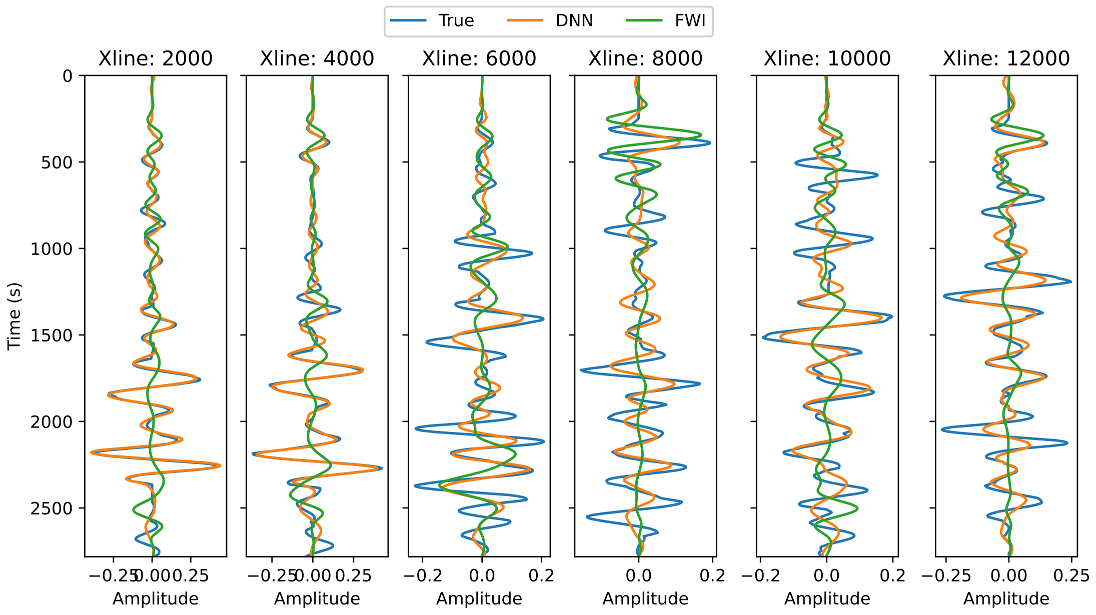


Figure 4.23: Time inversions of velocity profiles through Xlines on Initial, DNN and FWI results as shown in Figure 4.19.

4.3 | Theory-Guided RNN as an Analogue of FWI

Based on the formulation presented in the previous Chapter, results for theory-guided RNN as an analogue for FWI are presented in this Section. As indicated in the Literature Review, this idea is not novel. However, the use of pseudo-spectral spatial gradient calculation for the inversion process is novel. Two-dimensional experiment formulation are shown, and confirm the pseudo-spectral forward modelling implementation. This is then applied to synthetic data results.

4.3.1 | Experiment Setup

The original code for the time inversion as RNN framework was provided by Richardson (2018) and developed in TensorFlow v1.4. TensorFlow has been in active development since 2018, with a major release of v2.0 in September 2019. This provided much improvement in terms of efficiency and it allowed for easier implementation on GPUs. For this reason, the framework was re-written in v2.0 to enable the use of INGV's NVIDIA Titan V GPU. This code is published as part of the additional resources to this dissertation in Appendix B.1.

4.3.2 | Forward Modelling using RNNs

RHH should be able to model the different wave field components if it is to replace the forward modelling component. This was first tested by considering the 1D case for both Time and Frequency implementations and compared to a 1D Green function solution. This was achievable via a custom RNNcell unit developed in TensorFlow and can be inspected code repository provided in Appendix B.1. The 1D implementation provided promising results and is available as part of Appendix B.5.1.

The experiment was extended to 2D and tested for all wavefield components. A 25Hz Ricker wavelet was propagated through a 2D 1500ms^{-1} constant velocity model (Figure 4.24(a)) with a multi-source multi-receiver geometry setup. The 25Hz source wavelet goes into the hyper-resolution realm for FWI and is beyond the resolution that will be investigated on the synthetic model, however this allows for gauging the limit of accuracy. This model setup was forward propagated for 5333 time-steps at 1ms, with a 10m grid spacing. Namely, this implies that 5333 LSTM cells where employed for the forward modelling. The resulting direct waves are illustrated in Figure 4.24(b), with True being the analytical solution calculated using a 2D Green's function, RNN Time and RNN Freq are the RNN implementations for forward modelling using Time and

Fourier spatial derivatives respectively. Qualitatively, there is no visible difference between either approach.

Reflected and transmitted arrivals were tested using a simple step velocity model ranging from 1500ms^{-1} to 2000ms^{-1} (Figure 4.25(a)). Figure 4.25(b) is the forward modelled wavefield for the two receiver locations (RCV-1 and RCV-2), top and bottom respectively. RCV-1 at ground level interacts with the direct wave at 125ms and reflected wave at 250ms. RCV-2 is below the acoustic impedance layer at 30m and shows the transmitted wave. Comparing these to the analytical solution, either are able to model the wave components perfectly.

The remaining wavefield components are scattering waves. A constant velocity model of 1500 ms^{-1} was created with a 1550ms^{-1} point scatterer (Figure 4.26(a)). RNN implementations were modelled to be depended non-linearly on the scattering amplitude and then approximately linearised. The results are given in Figure 4.26(b). The direct wave was not included in the scattered wavefield reconstruction. Similarly to previous components, scattering are modelled successfully.

Table 4.7(a) lists quantitative metrics for the wavefield components. RNN Freq was found to be better for imaging the direct wave (Table 4.7(a)), with an improvement of 0.01 in error tolerance and 0.3% Relative Percentage Error (RPE). Metrics in Table 4.7(b) and Table 4.7(c) indicate that RNN Time matches the 2D Green's function near perfectly, whilst RNN Freq introduce error of less than 0.04 and 0.1% RPE. RNN Time is able to model the wavefield within a maximum 0.06 error tolerance and 1.74% RPE, whilst RNN Freq is overall more accurate with 0.05 and 1.449% respectively. Given these metrics and the observed models, the discrepancies between the analytical solution and the RNN implementation are deemed acceptable and should be suitable for the modelling process.

Modelling	Error Tolerance	RPE (%)
RNN Time	0.060	1.740
RNN Freq	0.050	1.449

(a) Direct wave.

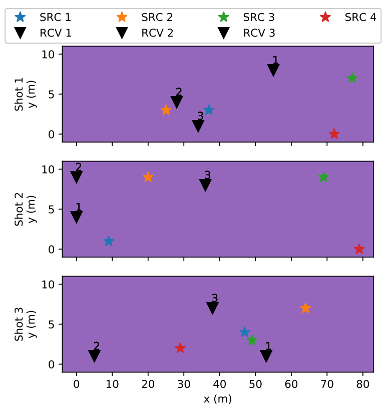
Modelling	Error Tolerance	RPE (%)
RNN Time	0.001	0.002
RNN Freq	0.020	0.013

(b) Reflected and transmitted wave.

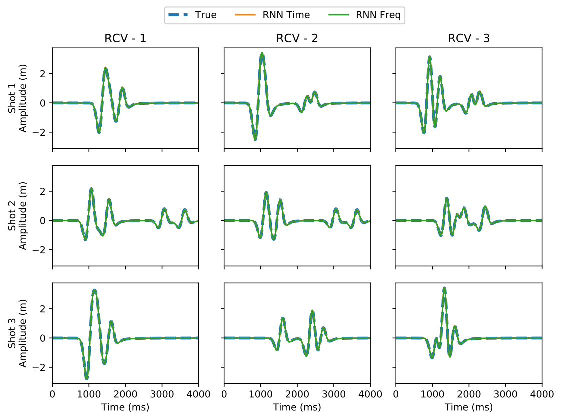
Modelling	Error Tolerance	RPE (%)
RNN Time – Non-linear	0.003	0.010
RNN Time – Linear	0.010	0.025
RNN Freq – Non-linear	0.030	0.076
RNN Freq – Linear	0.040	0.097

(c) Scattering wave.

Table 4.7: Empirical comparison of 2D wavefield components.

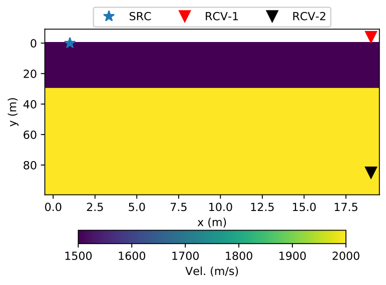


(a) Constant velocity model.

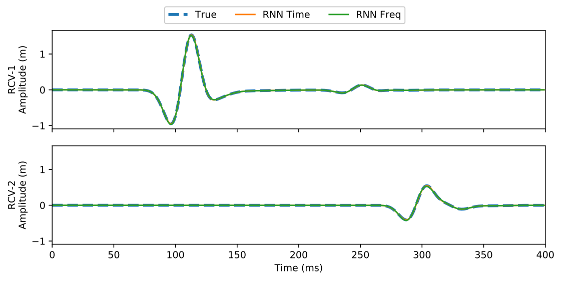


(b) Direct wave RNN forward modelling.

Figure 4.24: Direct wave forward modelling for multi-source, multi-receiver geometry.

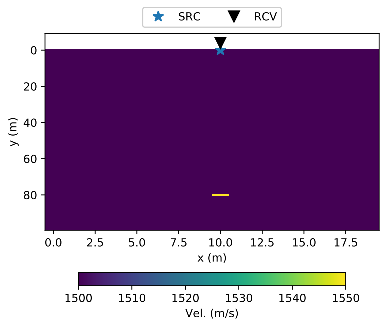


(a) Step velocity model.

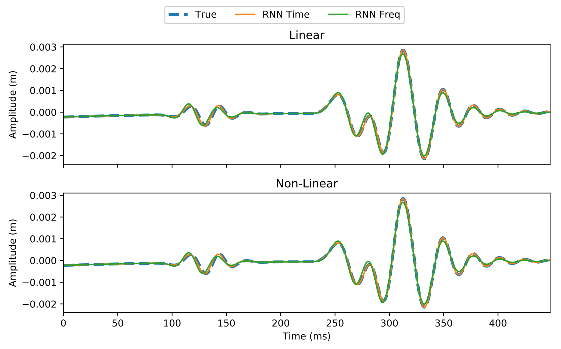


(b) Top: RCV-1 located at ground level reacts to direct arrival at 125ms and reflected arrival at 250ms. Bottom: RCV-2 shows transmitted arrival.

Figure 4.25: Reflected and transmitted wave RNN forward modelling.



(a) Point-scattering velocity model.



(b) Scattering wavefield modelling. Direct wavefield was excluded in the modelling.

Figure 4.26: Scattering wave RNN forward modelling.

4.3.3 | Gradient Comparison

The gradient of the cost function defines the direction in which the model needs to be updated to reach a global minimum (§ 3.1.1). Classical FWI approaches generally use the adjoint state method to calculate gradients or the finite differences approach (although computationally expensive), whereas DNN frameworks use automatic differentiation. Theoretical equivalence has been shown in Appendix A.9, and we now confirm computational equivalence following the approach described by the work of Richardson (2018).

A random 1D model was generated, randomly perturbed and gradient of cost function evaluated along the trace. Figure 4.27 and Table 4.8 compare the gradients at each point for classical finite differences and adjoint techniques to automatic differentiation (AutoDiff.). The adjoint state and AutoDiff. Freq react similarly and slightly overestimates the gradient, with the pseudo-spectral approach being worse. AutoDiff. Time under-estimates the gradient with an infinitesimal error. Gradients deviate at the edges in either case, with AutoDiff. Freq producing evident perturbation in the initial few time-steps. This is due to the choice of the batch-size within the inversion process and is further discussed in subsection § 4.3.4. Although this might seem worrying, the scale of this deviation is very minimal and no concerning effects were observed within the previous experimentation leading to this investigation. The other discrepancies are attributed to numerical inaccuracies as per Richardson (2018).

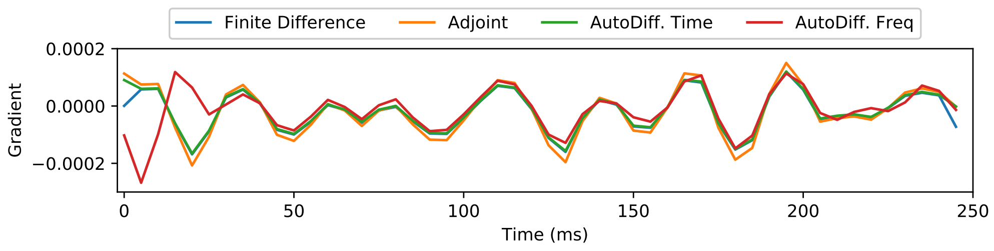


Figure 4.27: Gradient comparison of of RNN implementation with classical approaches. AutoDiff. is the automatic differentiation implementation in Tensorflow v2.0.

Finite Difference gradient baseline	Adjoint	AutoDiff. Time	AutoDiff. Freq
Error tolerance	1.000×10^{-5}	-2.196×10^{-9}	3.000×10^{-4}
RPE (%)	0.593	1.302×10^{-5}	1.779

Table 4.8: Empirical comparison of gradient calculations.

4.3.4 | Hyper-Parameter Tuning

Similarly to the approach shown in Sun et al. (2019), a benchmark 1D 4-layer synthetic profile, with velocities [2, 3, 4, 5]kms^{-1}, was used to identify the ideal parameters for the RNN architecture. This is illustrated as the Black line in Figure 4.28. Classical 1D second-order FD modelling was used to generate the required true receiver data. Multiple learning rates for the different loss optimizers were investigated to try and identify the ideal combination. Figure 4.28 shows the best combination for all losses with an ideal batch size of three. The full investigation for this tuning is given in Appendix B.5.2.

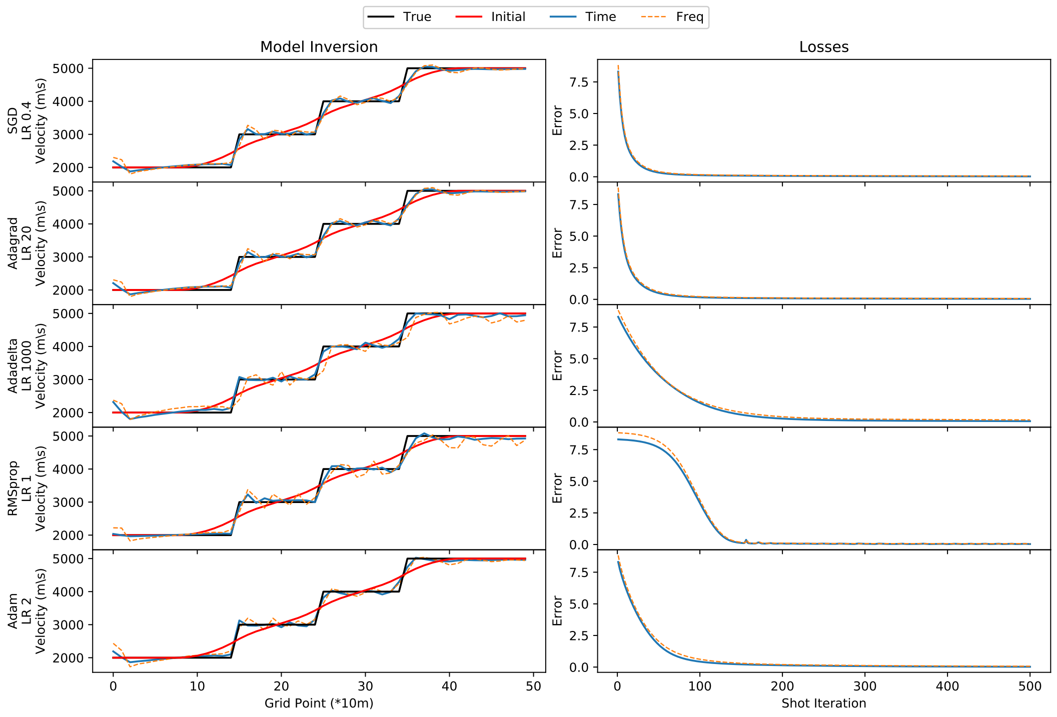


Figure 4.28: Tuning of hyper-parameters to identify ideal loss optimizer combination.

Left side of Figure 4.28 shows the inverted velocity profiles, with Red being the initial velocity profile. For Stochastic Gradient Descent, the learning rates was found to be both between zero and one. This is as expected and follows conventional loss optimization. On the other hand, the other loss optimizers had to be scaled to beyond one due to the magnitude differences brought by accumulated squared-norms of the gradients as investigated by Sun et al. (2018). This is allowed provided the scaling coefficient is between zero and one. For Adagrad, following from Duchi et al. (2011), the β hyper-parameter was fixed at 0.9 and learning rate found to be 20. Adadelta, RMSprop and Adam optimal learning rates were identified at 1000, 1 and 2 respectively.

The right side of Figure 4.28 gives the loss progression. All optimizers iteratively reduce the error with additional shots and on similar scales. Stochastic Gradient Descent and Adagrad do this relatively sooner than the rest, yet the inverted velocity is not as good as the other optimizers. RMSprop follows a rather slow gradual decrease in loss, which then sudden increases. This is expected given that RMSprop updates are derived from a moving average of the square gradients and require an inertial start.

Based on this investigation, **Adam** with a learning rate of 2 was identified as the best optimizer. This provided the most stable inversion for either RNN Time or Freq, with the most update and reasonable error loss performance. Mis-match in the shallow part of the velocity is due to the choice of batch-size within the RNN update process. Figure 4.29 shows the Adam optimizer fixed with learning rate 2 and inverted for batch sizes ranging from one to five. The smaller the batch size, the greater the error since the inversion is more localized and amplifies the gradient onset error shown in Figure 4.27. The larger the batch size, the better the inversion as more data is being used. This poses a limitation since batch size is limited by the Graphical Processing Unit RAM. Given fore-sight that this approach will be used on a large dataset, this was taken as a caveat and batch size fixed at one for the rest of the implementation.

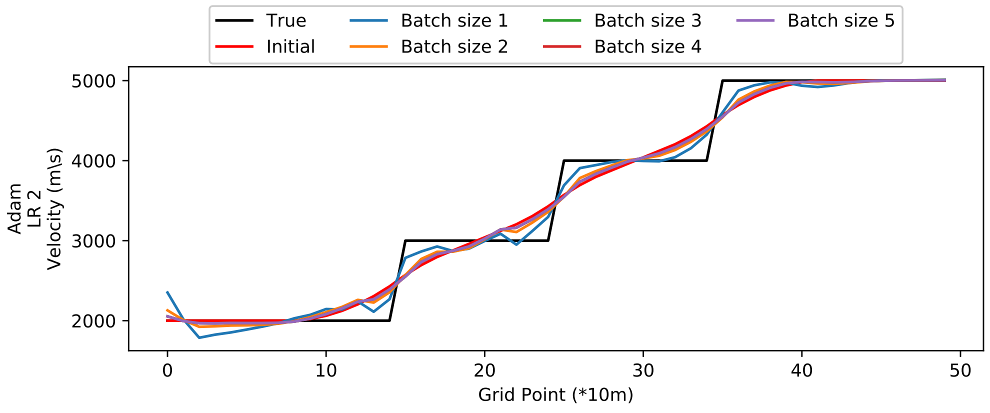


Figure 4.29: A smaller batch-size introduces error at the initial part of the velocity profile due to more localized updates. This was derived for a time time implementation for RNN architecture with Adam loss optimizer and learning rate of 2.

4.3.5 | 2D Synthetic

4.3.5.1 | True and Initial Models

To expand the numerical analysis of FWI through RNN training, a 2D scalar model of the Marmousi-2 was formulated similar to that in § 4.2.7.1. This was re-sampled to a 50m×50m grid and smoothed to create the initial model model. These velocity models are plotted in Figure 4.30. True synthetic receivers were computed by forward modelling through the RNN framework. 56 shots at 300m intervals at depth 200m were generated with a Perfectly Matched Layer at the boundaries. Receivers were set at 50m intervals and modelled for 12s duration.

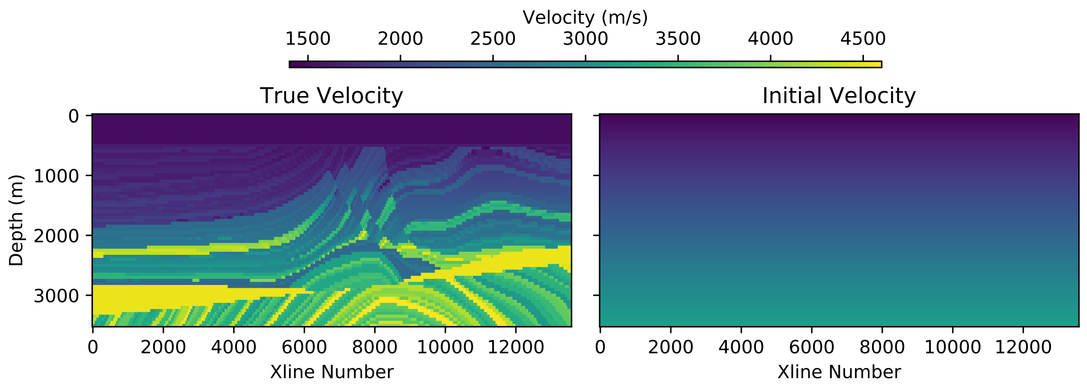


Figure 4.30: 50m×50m grid 2D Marmousi models for RNN training.

4.3.5.2 | Training of RNN

As in standard RNN approaches, the receiver dataset was split into a training and development datasets with at 75%-25% split. Training was run for 100 epochs, with early stopping on an NVIDIA Titan V Graphical Processing Unit courtesy of Istituto Nazionale di Geofisica e Vulcanologia. Development loss was calculated every 5th training shot. Figure 4.31 gives the RNN performance for training and development datasets using Adam optimizer with learning rate of 2.0 and batch size 1. The horizontal labels shows the epoch number and respective number of shots evaluated for training and development. Computational run times are of 14 hours per approach. Both RNN Time and RNN Freq follow similar reductions in loss per epoch and indicate that either implementation converge to an optimal loss. L-BFGS-B loss for classical FWI is shown and is discussed is in the next section.

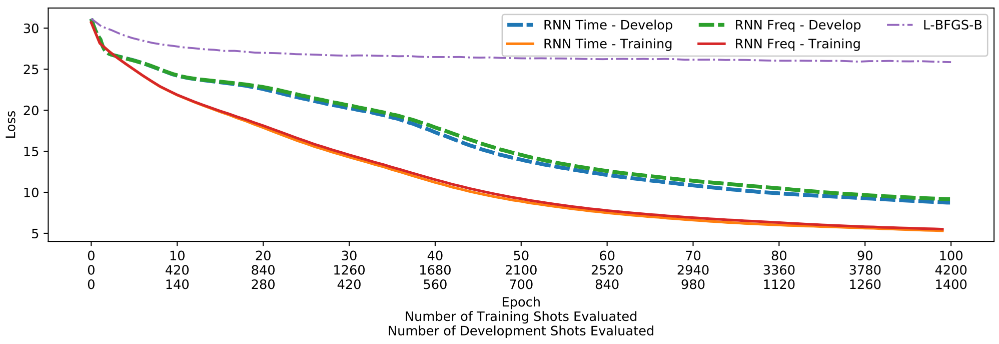


Figure 4.31: RNN loss performance for RNN training and development datasets using Adam optimizer with learning rate of 2 and batch size 1. The horizontal labels shows the epoch number and respective number of shots evaluated for training and development. L-BFGS-B is the cost function evaluation for classical FWI plotted on shot number equivalent. Either RNN approach converge quicker than L-BFGS-B, and RNN Freq provides a more stable convergence and better performance then RNN Time.

4.3.5.3 | Comparison with classical FWI

Figure 4.31 plots the cost function versus the number of shot evaluation equivalent for classical FWI and RNN. The RNN framework is more computationally efficient since either RNN approach converge significantly quicker than L-BFGS-B. RNN Freq provides a more stable convergence and is better performant then RNN Time. The classical FWI is plotted as a shot number equivalent and not the epoch number. The full cost function performance is provided in Appendix B.3.

Figure 4.32 compares the inverted velocities and residuals for FWI, together with RNN Time and RNN Freq implementations. Complementary plots showing the model update progressions for this sections are provided as part of Appendix B.5.3. The true model velocity in Figure 4.32 identifies three zoomed areas which are shown in Figure 4.33 and Figure 4.35 are velocity profiles taken at 2000 Xline intervals. Figure 4.34 show the resolution spectra derived via FFT on the velocity models. Comparing FWI and the RNN model in either of these figures, it is clear that the resolution recovery is different. Figure 4.34 confirms the frequency content in these approaches and shows how RNN models invert more of the lower frequencies in Zoom 2 and Zoom 3. In Zoom 1, FWI is slightly better at frequency recovery beyond 25Hz.

Residual plots (Figure 4.32- 4.33) and the velocity profiles (Figure 4.35) show how RNN approaches are able to recover more of the signal in the shallow right side (Zoom 1) and the over-thrust middle area (Zoom 2) of the model. Almost all the signal up to depth 1500m is inverted correctly in Zoom 1 whereas over-thrust faults are near perfectly recovered in Zoom 2 and 4.33F. Zoom 3 is of most interest. The prominent layer at depth circa 2000m is nearly completely missed by RNN models, whereas FWI is able to recover this partially. On the other hand, the deeper 3000m strata are hardly identified with FWI. Residual figures in the full sections show that the RNN model amplitude recover is not as good when compared to FWI (Labels 4.33A-B). Indeed, some layers are missed at depth greater than 1500m for Xline number greater than 10,000 (Label 4.33C-D). Considering either RNN approach in Figure 4.32, there is a low-frequency *shadow* artefact introduced till depth 2300m from Xline 0 to 6000 and Xline 8000 to 13900. This is attributed to the practical implementation of batch-size discussed in § B.5.2.

Figure 4.36 shows labelled receivers for either model at Common Depth Point (CDP) 60, 150 and 300. These CDPs split the model into three sections, representing the different extremities. Label A and B reiterate that the shallow left side is better imaged for FWI, whilst shallow right side is better for RNNs respectively. Label C is the missing high velocity at depth 2000m which has incorrect amplitude for the RNNs, but positioned correctly. Classical FWI has less prominent leakage in this area, yet very evident. Label D is the badly imaged layer at depth between 2000m and 2500m on the right side of the model. Labels E throughout the residuals highlight better low frequency resolution imaging by RNN approaches. Indeed, RNN Freq is able to recover slightly more of these low frequencies and identified by E^1 and E^2. Similar improvements are visible throughout the other plots.

Intentionally left black space.

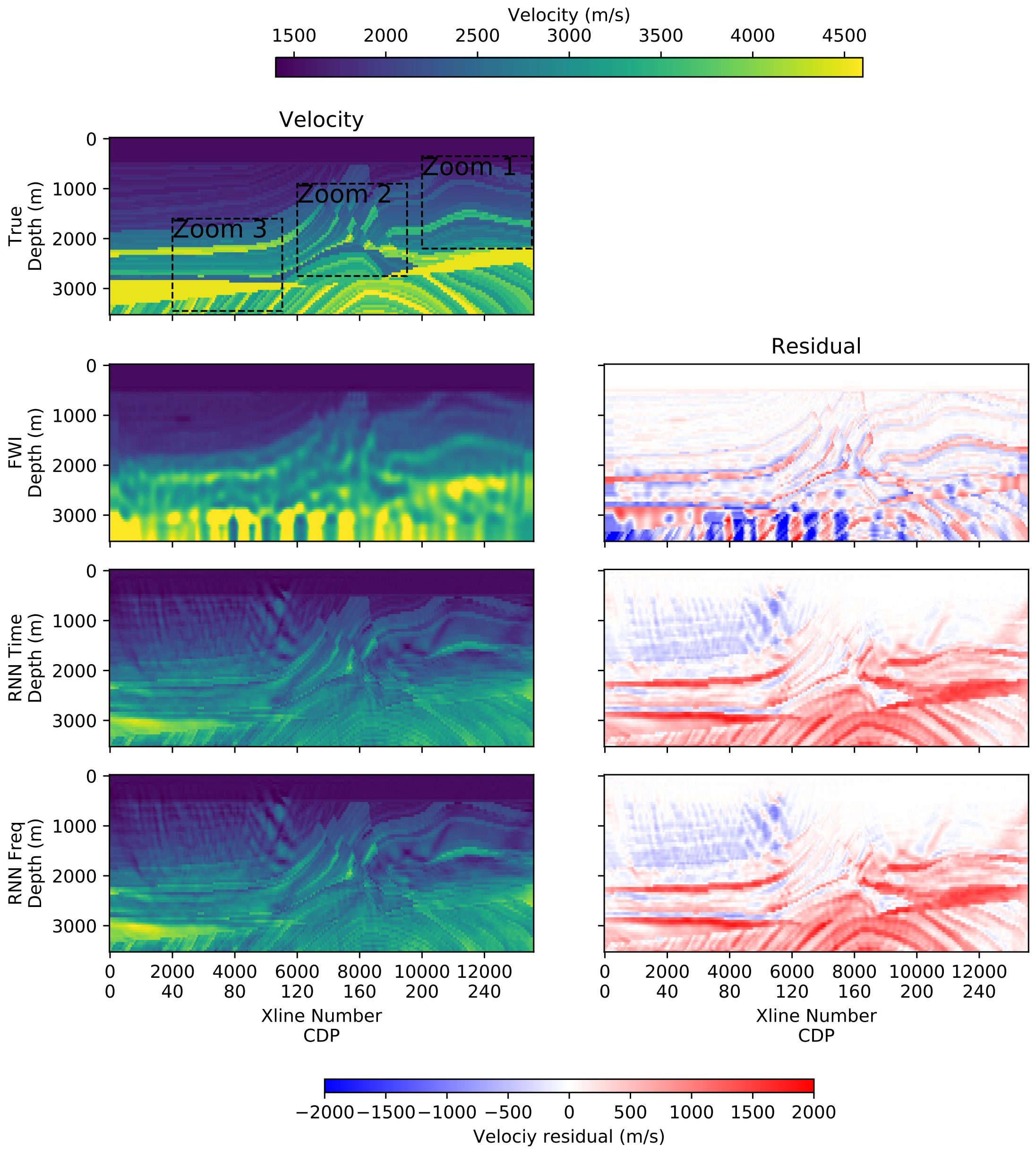


Figure 4.32: Classical FWI and RNN implementation velocity model inversion.

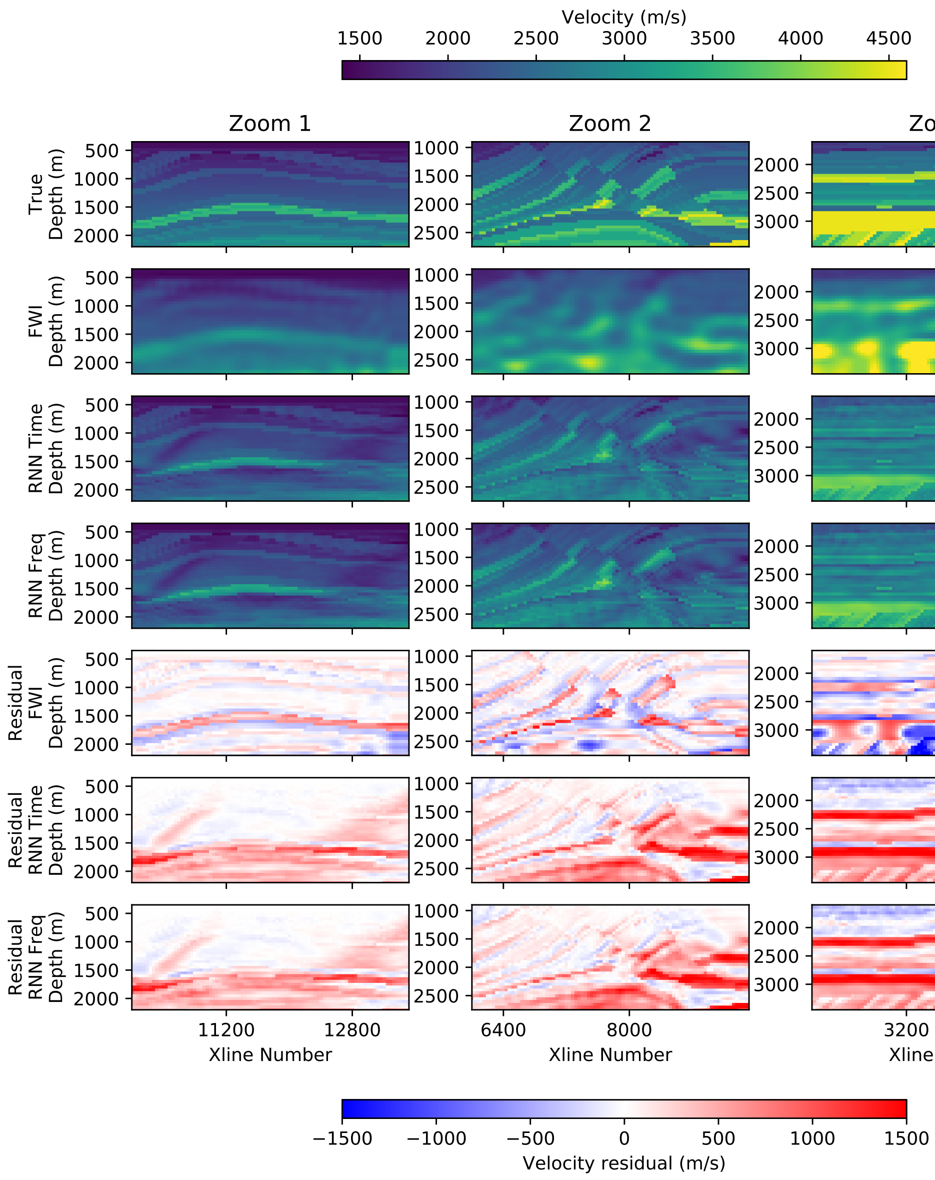


Figure 4.33: Zoomed In RNN Models

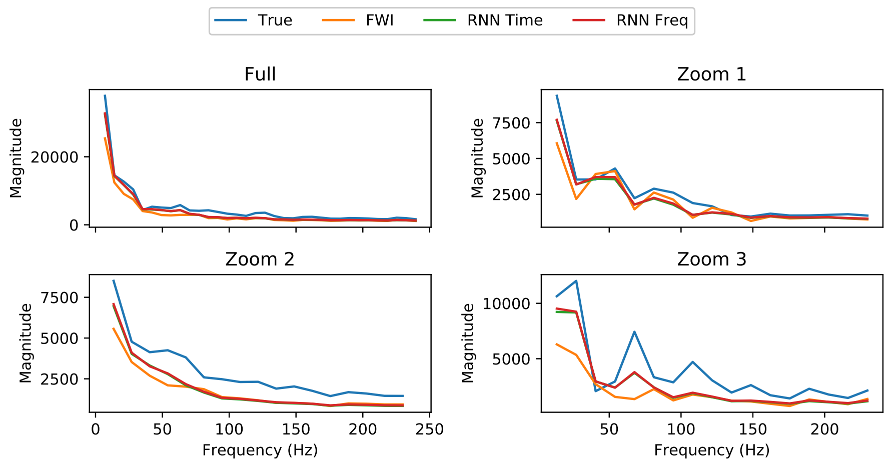


Figure 4.34: RNN model velocity resolution spectra.

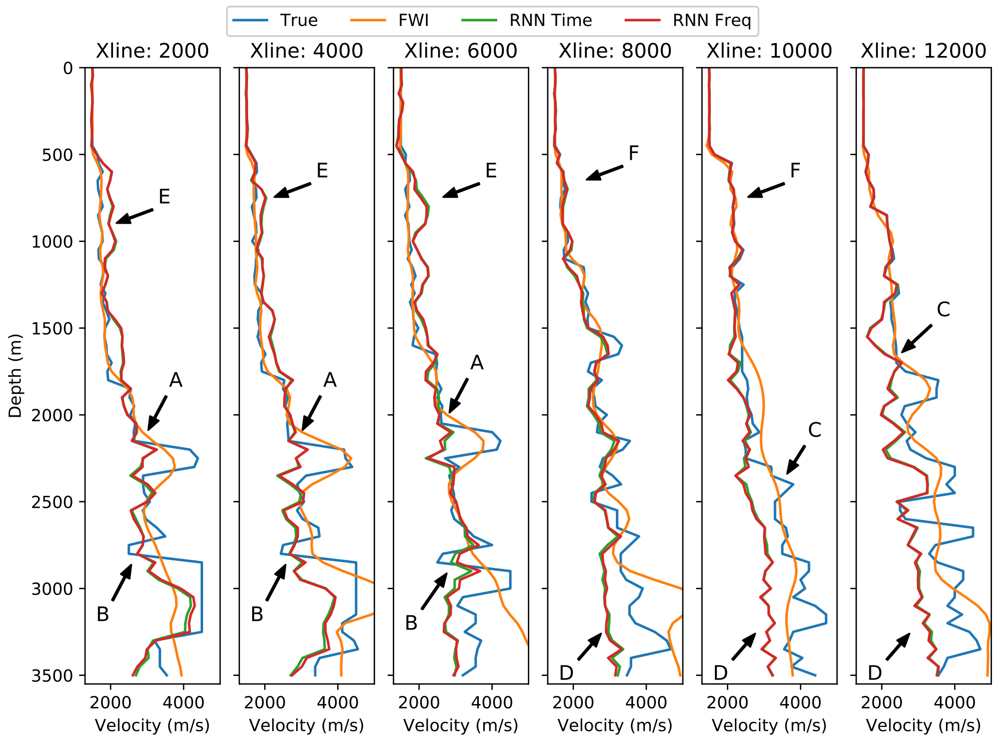


Figure 4.35: Comparison of velocity profiles for RNN and classical FWI. Label **A-B**: RNN is able to identify strata near perfectly, however unable to inverte the amplitudes values correctly. Label **C-D**: Missed layers from RNN approaches. Label **E**: Low frequency artefact for RNN. Label **F**: Near perfect velocity inversion in the middle Xlines, over shallow depth.

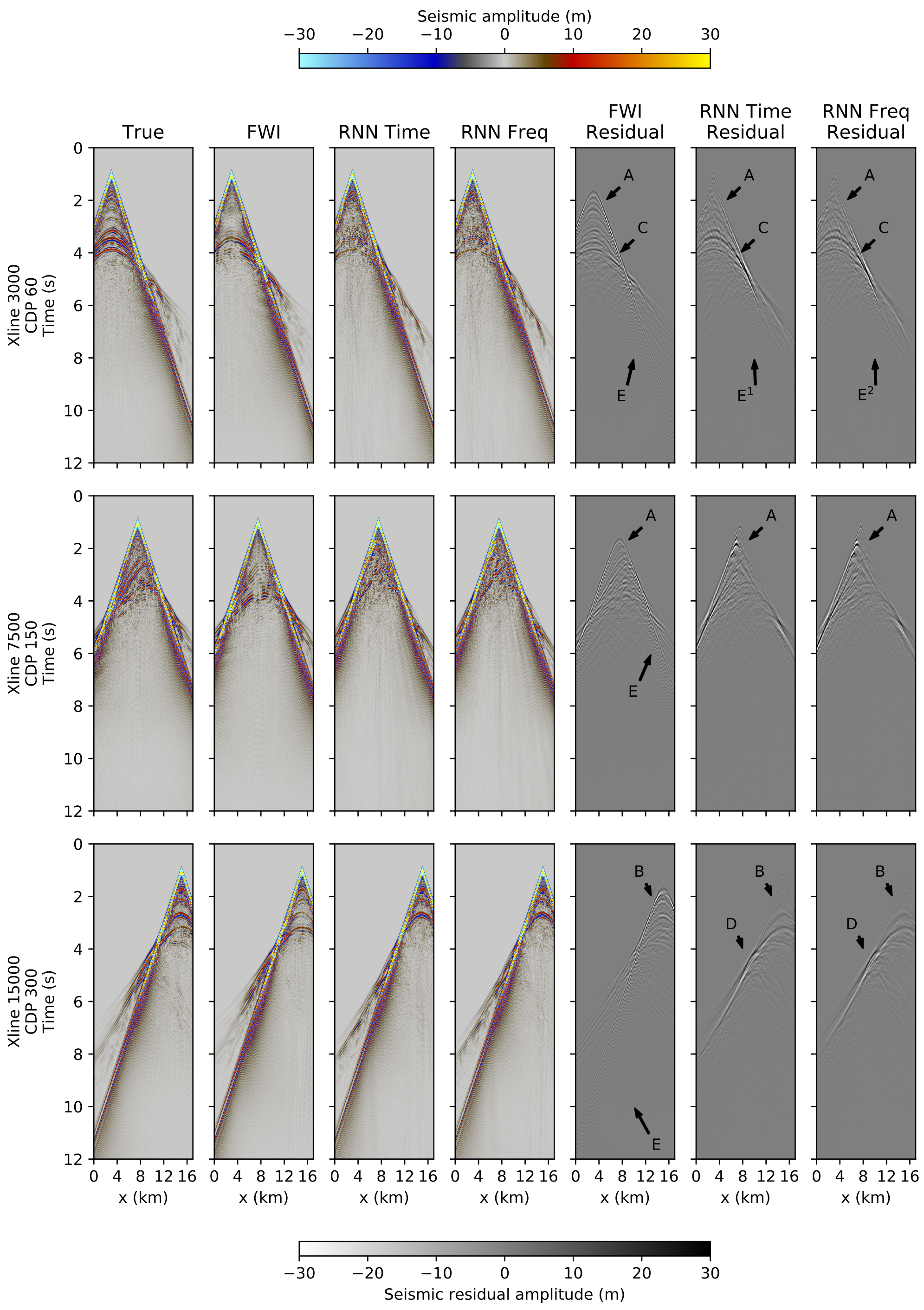


Figure 4.36: Receivers for True, FWI and RNN models at CDP 60, 150 and 300. Label **A** and **B**: Shallow left side is better imaged for FWI. Label **C**: Missing high velocity with incorrect amplitude but positioned correctly. Label **D**: Badly imaged layer. Labels **E**: Better low frequency imaging for either RNN approaches. Labels $\mathbf{E}^1$ and $\mathbf{E}^2$: RNN Freq is able to recover slightly more low frequencies.

4.4 | Data-Driven and Theory-Guided NN Frameworks

In this section, DNN refers to "Data-Driven FWI" and RNN to "Theory-Guided RNN as an Analogue of FWI".

4.4.1 | Data Volume

The volume of data used to train for DNN was 20 orders of magnitude greater. Figure 4.37 shows the normalised loss for the two derived frameworks, with respective count for the number of training and validation shot or shot equivalent. Given that the DNN framework used randomly generated traces with each epoch of 1,000,000 and 100,000 trace for training and validation respectively, these were batched into groups of 340 traces to match the number of receivers in the RNN approach.

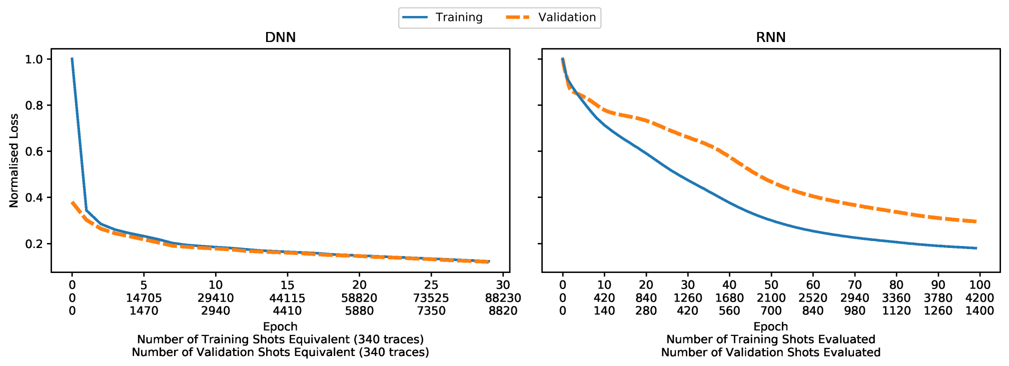


Figure 4.37: Classical FWI and RNN implementation velocity model inversion.

4.4.2 | Data-Driven Uplift

DNN produces more imaging uplift since it is not bound by the deterministic forward-modelling physical constraints from ray-tracing. Figure 4.38 shows a sample of shots through the Marmousi model. Ray-paths below the high velocity at depth 2-2.5km do not arrive at the receivers at the surface for the current model offsets. Acquisition geometry controlling the offset is a hard-limit within FWI. Morgan et al. (2009) considers large-offsets to be fundamental for successful FWI. Moreover, this hinders seismic imaging in and around salt bodies since, by definition, only one-way ray-paths are considered (Jones and Davison, 2014). In the case of the data-generators used for the DNN approach, these are free from offset-constraints and do not influence the inversion experiment employed by classical FWI.

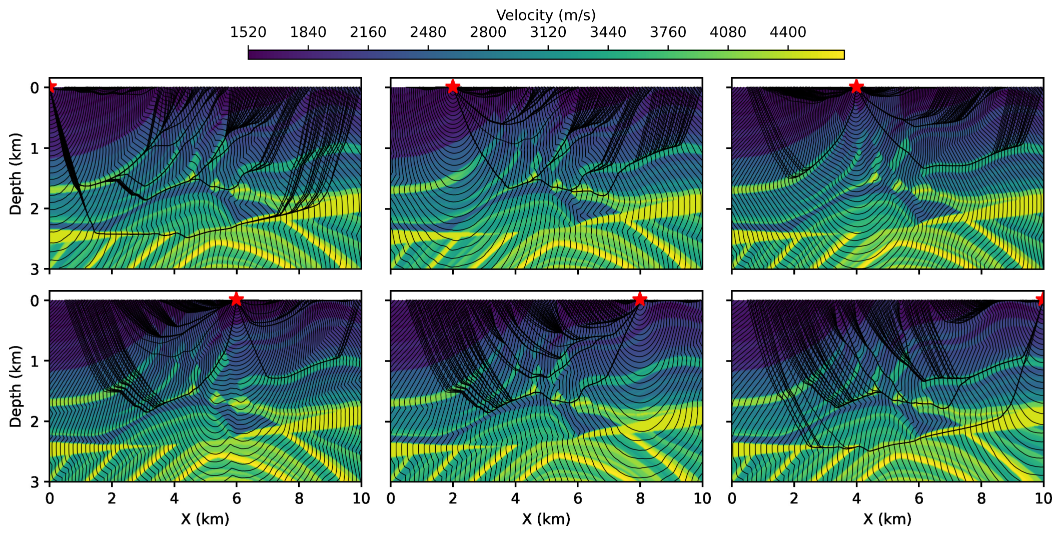


Figure 4.38: Ray-tracing coverage within forward modelling derived from deterministic geophysics.

4.4.3 | Inversions

Figure 4.39 and 4.40 are the full and zoomed-in sections for the inverted models for classical FWI, DNN and RNN respectively. DNN has better layer amplitude continuity as seen by in Zoom 1 and Zoom 3. RNN is better at edge definition than both DNN and FWI as in Zoom 2.

The velocity profiles in Figure 4.41 illustrate with Label A how either approach is good in the middle and shallow sections. Indeed, if the RNN artefact labelled A^1 is excluded in smaller Xlines, this would be valid for either Xline. Label B show how large velocity contrasts within the velocities is well defined for DNN, followed by FWI. In particular, the RNN approach is able to identify the edges, but not able to reconstruct the amplitudes correct. Smaller velocity increases are not an issue as shown by B^1. A combination of FWI and RNN could potentially exploit the benefit of either. However, not in areas with large velocity contrasts such as salt areas and carbonates. With depth, DNN is a better framework as marked with Label C. Comparing RNN and FWI, RNN is more suited at the edges with Depth as shown with Label C^1.

Receivers for pseudo-spectral NN frameworks and true models are shown in Figure 4.42. Either approach is able to recover different shallow arrivals successfully as marked with Label A. DNN has residual and RNN is able to invert more of the shallow, however FWI recovers direct arrival components nearly completely. Considering CDP 150 and 300, RNN is reinforced as being better performant at the edges from the residuals marked with Label A[1]. Excluding the gradients artefact of RNN, theses receivers indicate that RNN is the most suited for shallow sections. Both DNN and RNN have leakage on CDP 60 and CDP 300 as shown with Label B. RNN's residual show evidence of signal, possible symptomatic to cycle-skipping. With depth, RNN in general has less residual as shown with Label C.

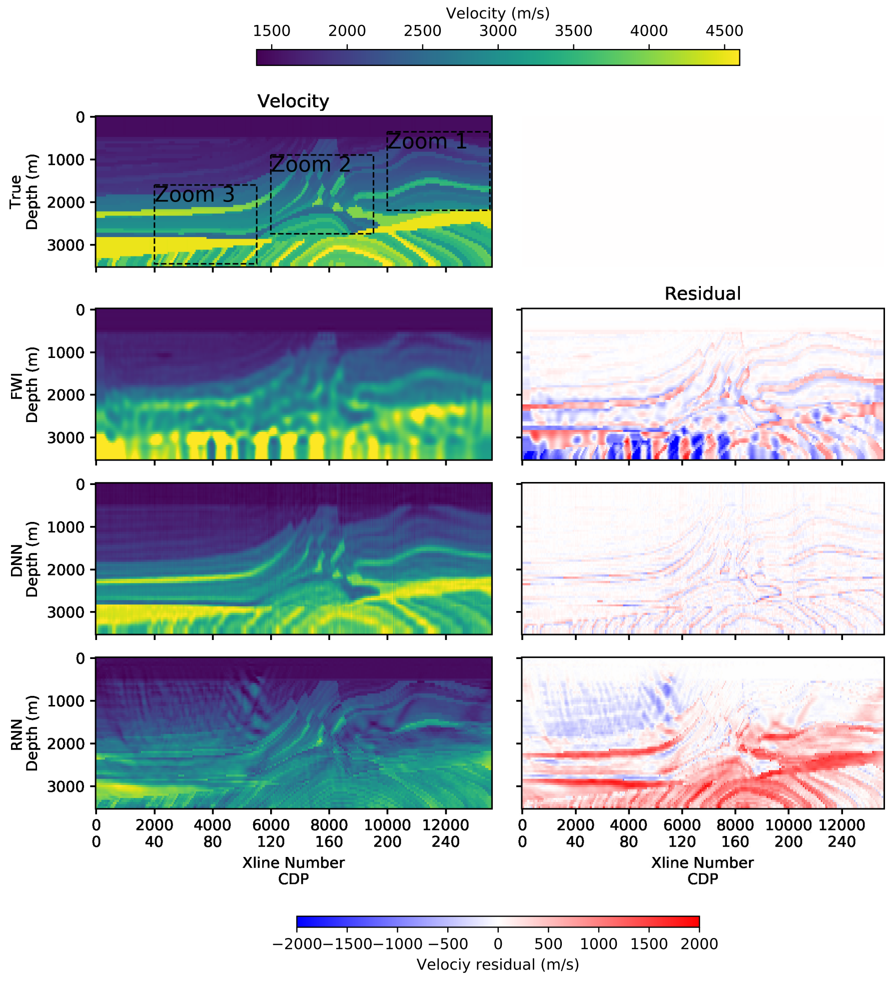


Figure 4.39: Full model showing differences between DNN and RNN velocity inversion as compare to classical FWI, with residual differences.

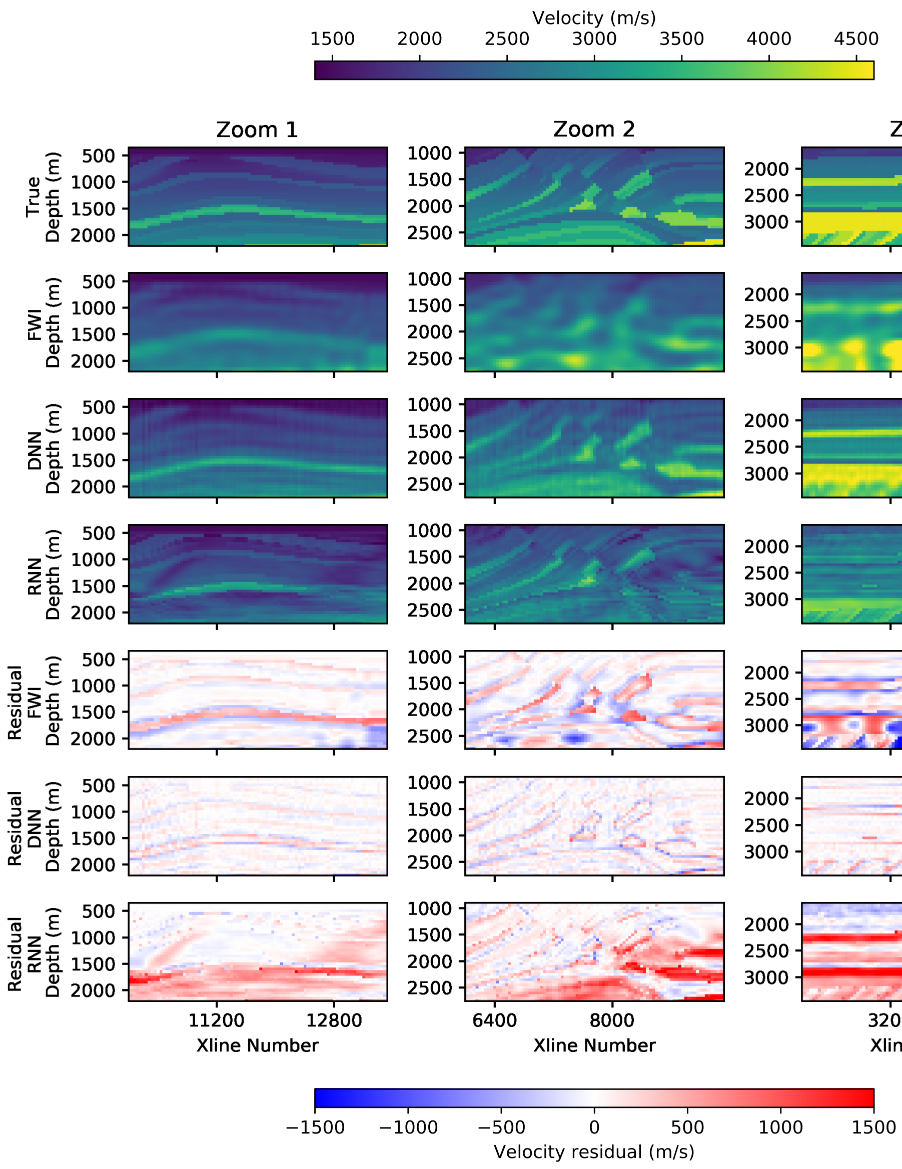


Figure 4.40: Zoomed in sections showing differences between DNN and RNN velocity inversion as compare to classical FWI, with residual differences.

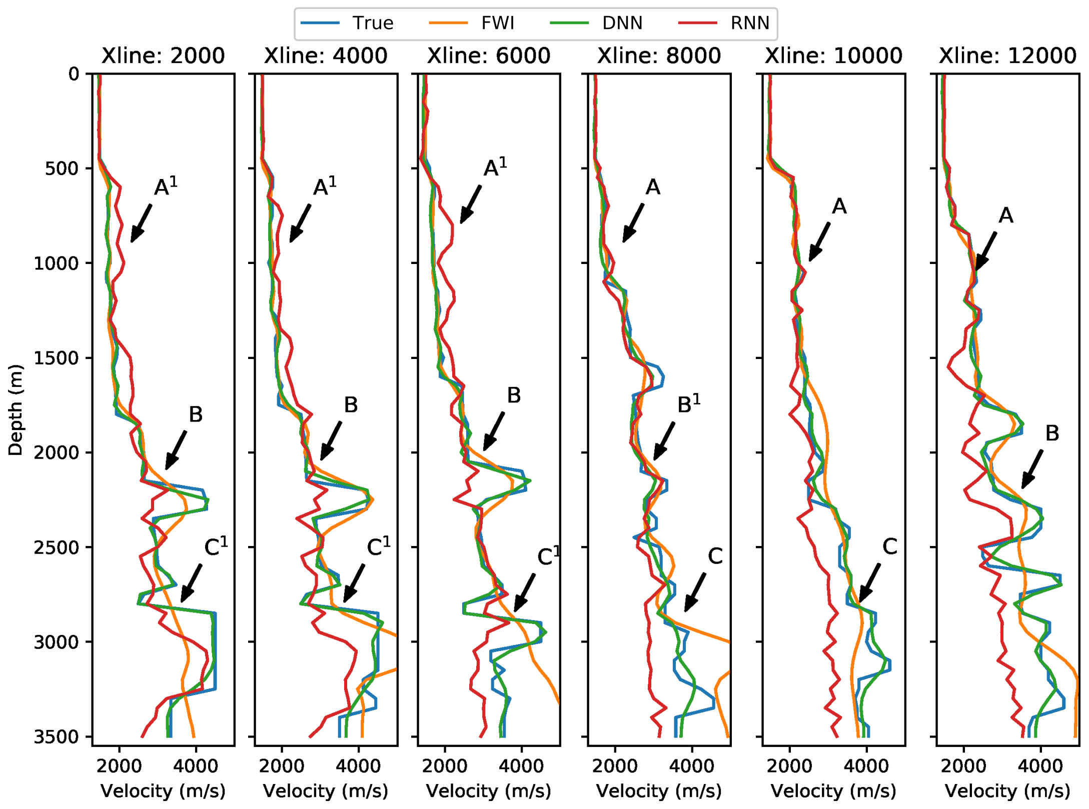


Figure 4.41: Comparison of velocity profiles for DNN, RNN and classical FWI. Label **A**: Either approach is good in the middle and shallow sections, with the exception of the gradient artefact marked $\mathbf{A}^1$. Label **B**: Problematic recover of large velocity contrasts for RNN, whilst smaller velocity contrasts are inverted corrected as marked by $\mathbf{B}^1$. Label **C**: DNN is a better framework with depth for velocity, whilst RNN is more suited at the edges as shown with $\mathbf{C}^1$

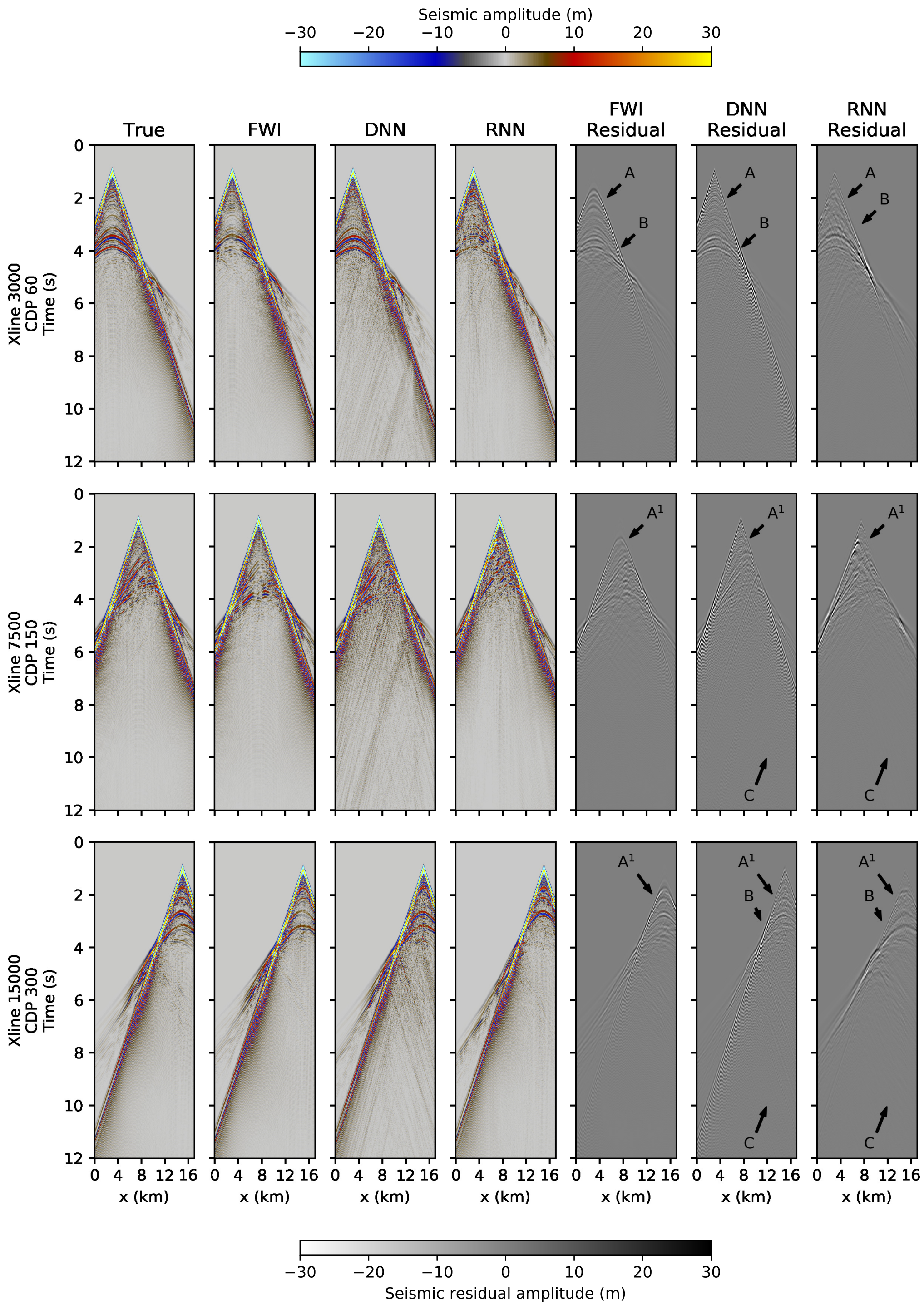


Figure 4.42: Labelled receivers for True, FWI and RNN models at CDP 60, 150 and 300. Label **A**: NN frameworks recover shallow arrivals successfully, with RNN reinforced as being better performant at the edges from the residuals marked with $\mathbf{A}^1$. Label **B**: Some signal leakage on NN frameworks. Label **C**: Less leakage from RNN with depth.

4.5 | Summary

The key outcomes from this chapter can be summarised in the following lists. For "FWI as a Data-Driven DNN":

- Elements within a classical FWI framework can be replaced with a DNN framework successfully. This was practically assessed for multi-strata model using geophysics-generated data.
- Normalization is not a necessary component within the framework.
- The ideal architecture is a 1D convolutional based (Conv1D), with Adadelta loss optimization.
- Kernel and bias distribution plots for the training process confirm that the framework is a learning process.
- Most of the update happens within the first few epochs. Additional epochs refine the inverted velocity.
- Inversion performance in shallow section was equally good for either classical FWI or DNN approach. DNN framework performs better for deeper and overthrust areas since DNNs are not bound by forward-modelling physical constraints from ray-tracing.

For "RNN as an Analogue of FWI":

- Pseudo-spectral RNN frameworks are feasible approaches.
- Based on comparisons of a simple model with an analytical Green's function formulation, RNN Time is able to model the wavefield within a maximum 0.06 error tolerance and 1.74% RPE. RNN Freq is overall more accurate with 0.05 error tolerance and 1.449% RPE.
- Adjoint state and RNN Freq gradients overestimate the Finite Difference, whilst RNN Time under-estimates it with an infinitesimal error. RNN Freq produce a perturbation on the onset of the gradient which is attributed to modelling artefact.
- Based on the model size and compute available, the ideal loss was Adam with a learning rate of 2 and batch size of 1. Model batch size proved to be a limitation for practical implementations.

- RNN is computationally more efficient than the classical FWI presented in this work. RNN freq shows more stable convergence.
- Classical FWI and RNN approaches have merits. RNN frameworks are able to identify faults, but amplitudes are not fully inverted properly. This results in RNN inversions with some missing layers. However, the low frequency content in RNN approaches is better than classical FWI, particularly for RNN Freq.

Outcomes of the comparison between the NN approaches:

- Well performing DNN was achievable through the use of a very large dataset.
- DNN recovers more of the velocity contrast and RNN is better at edge definition.
- RNN is the most suited for the shallow sections (exclude the gradient artefact) and at depth due to the cleaner residuals on the receivers. This is only valid in the middle section of the model with the most coverage from ray-tracing. Indeed, RNN approaches have some leakage on the edges which might be symptomatic to cycle-skipping.
- RNN Freq is able to recover more low frequencies.

5

Discussion and Conclusions

In this chapter, DNN refers to "FWI as a Data-Driven DNN" and RNN as "Theory-Guided RNN as an Analogue of FWI". The first section presents a critique to the work, addresses possible pitfalls, identifies areas of potential improvement and suggests directions for future research. This is followed by a final section which presents concluding remarks for this dissertation.

5.1 | Critique, Limitations and Future Work

5.1.1 | Inversion Paradigm

DNN, RNN and FWI are three different frameworks which fall within different parts of the inversion spectrum. On one end there is the strongly data-driven approach for DNN, on the other there is purely deterministic geophysics with classical FWI, and in between there is theory-guided RNN.

Within the DNN approach, the inversion component is data-oriented and the data generator is based on geophysics. If data-generators are to include information that might not be deterministically available within a seismic survey, the reconstruction process could invert for this information to Lewis and Vigh (2017)'s work. A similar approach has already been applied by Bubba et al. (2019) for brain CT-scans (Figure 5.1). These types of data-driven models could be pre-cursors for deterministic models as has been done in the work of Araya-Polo et al. (2018). Figure 5.2 shows inversion for classical FWI without and with DNN as a priori model. The latter approach produces more layer continuity and better imaging at depth. This comes with relatively no extra overhead cost since the DNN would be a pre-trained network. Furthermore, Grohs et al. (2019) analytically determine the upper bounds for the approximation and generalization power for a NN. This could be implemented and provided as a caveat with data-driven inversion by providing confidence maps for the inversion.

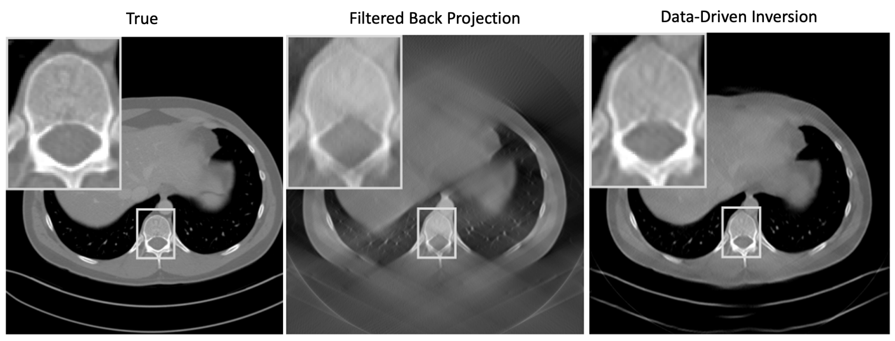


Figure 5.1: Inversion for brain CT-scan, with area of interest. Classical inversion via Filtered Back Projection has areas of poor-illumination due to limited-angle, whilst the data-driven approach offers better inversion. Adapted from Bubba et al. (2019).

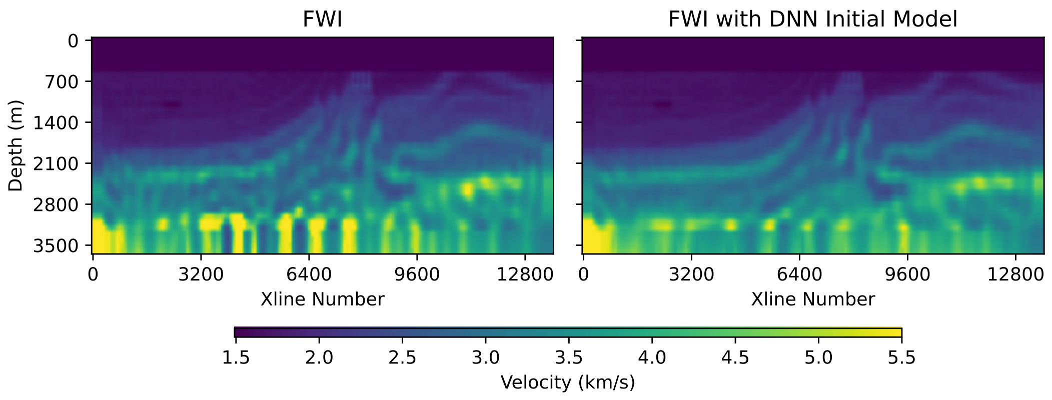


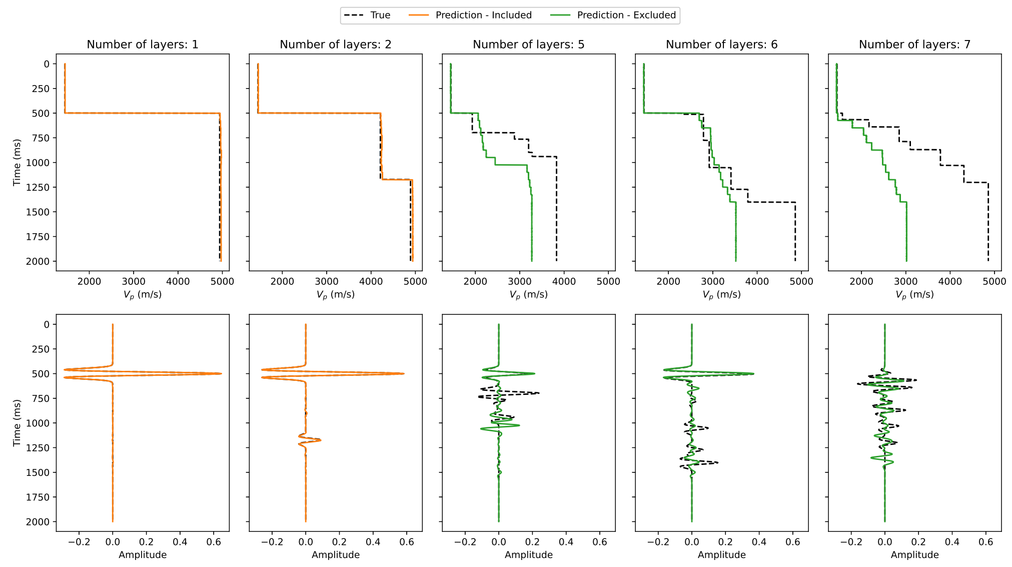
Figure 5.2: Improved FWI result with DNN as initial model.

Theory-guided inversion inherits advantages and problems from either part of the spectrum. It faces challenges of cycle-skipping and local-minima, whilst it benefits from the use of automatic differentiation to calculate the gradient. This reduces development time as it avoids the need to manually implement and verify the adjoint state method. Furthermore, being at the intersection of physics and computer science, it is inherently strengthened by contributions from two communities of researchers. This opens up possibility of considering other deep learning techniques such as dropout or other acyclic-graph architectures such as directed acyclic graphs (Bogaerts et al., 2020).

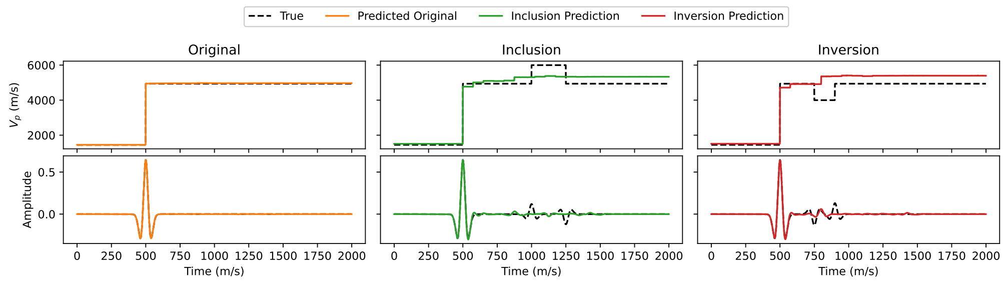
In classical FWI, the wavefield at the final time step is affected by the wavefield during the initial time steps. Back-propagation must occur over the entire sequence of time steps for theory-guided RNN. Application of back-propagation through thousands of layers is not a typical application in deep learning applications and automatic differentiation is not designed to efficiently handle such situations. Strategies common to other FWI frameworks to reduce memory requirements could be translated into the field. Examples would include not saving the wavefield at every time step (Nguyen and Mcmechan, 2015), applying compression to the wavefield time slices (Boehm et al., 2015; Kalita and Alkhalifah, 2017), saving wavefields to disk rather then memory (Shen and Clapp, 2015), and regenerating wavefields during back-propagation rather than storing them (Malcolm and Willemsen, 2016; Yao et al., 2020). Data-driven inversion is gradually proving itself to be a field of its own. An account for some of the main contributions is provided by Arridge et al. (2019). It is up to the end user to decide to which level of confidence they are willing to base their inversion on data.

5.1.2 | Training Datasets for Real Data

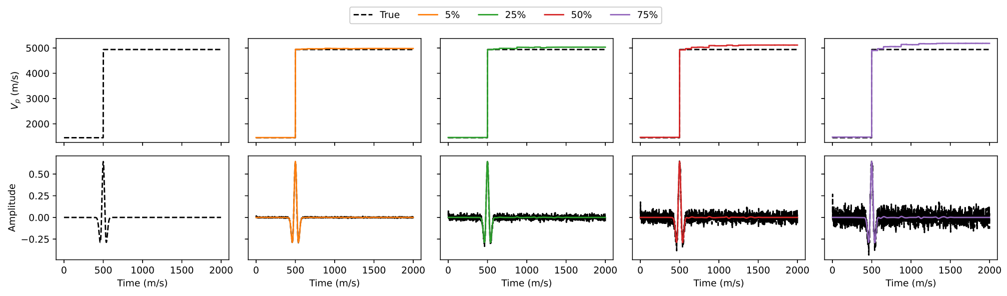
The DNN requires a global model for a real world problem. Consider as example a data generator trained on data that is limited to 3 layers and inversion is carried out for a system made for more than 3 layers. The inversion process will start degradation as shown in Figure 5.3(a). Conv1D-Adadelta DNN was trained for 30 epochs for a maximum 3 layers for velocity ranging from 1450ms^{-1} to 5000ms^{-1}. The first two columns are inversions for velocity profiles within the generator limit for the number of layers. These inversions are of good quality as expected from previous results. For more layers in the velocity profile, the inversion tries to generalize for these profiles but misses some components. The inversion is still able to identify some layers, but not to the same resolution as if trained on those layers. The large velocity contrasts are inverted correctly as shown for the 6 layer velocity profile. Similarly, the geophysics models allowed within the problem need to be considered. Figure 5.3(b) illustrates inversion for a large velocity of 6000ms^{-1} and velocity inversion respectively. Given that these models were not included in the development dataset, the inversion process would never be able to invert for these velocity types correctly. On the other hand, the DNN framework is robust to different noise levels. Figure 5.3(c) showcases the inversion for pink noise contaminated data at different levels. Pink noise contamination was chosen due to its prevalence in the low frequency limit (Randall, 2009), thus making it more suited to assess the effect on the low frequency component of FWI. The inversion remains relatively unaltered before 25% and then starts degradation. This inversion process could be used as a de-noising technique given the correct data-generator.



(a) The first two columns are inversions for velocity profiles within the generator limit for the number of layers. For more layers, the inversion tries to generalize for these profiles but misses some components. The inversion is still able to identify some layers, but not to the same resolution as if trained on those layers.



(b) Large velocity inclusions and Velocity inversion would be missed as well.



(c) Robustness to noise up to 25% pink noise contamination.

Figure 5.3: Conv1D-Adadelta DNN trained for 30 epochs for a maximum 3 layers for velocity ranging from 1450ms^{-1} to 5000ms^{-1}. The quality of the development dataset directly influence the quality of DNN inversion. A: Missing layer. B: Different geophysical models. C: Noise sensitivity.

Modelling techniques and transform spaces could provide an alternative approach for the DNN. Jozinović et al. (2020) use Wigner-Ville distributions for pseudo-spectral representations of the seismograms for prediction of intensity measurements of ground shaking, or other representations such as Recurrence Plots (Kamphorst and Ruelle, 1987), Markov Transition Fields (Wang and Oates, 2015) and Gramian Angular Fields (Wang and Oates, 2015) - See Figure 5.4.

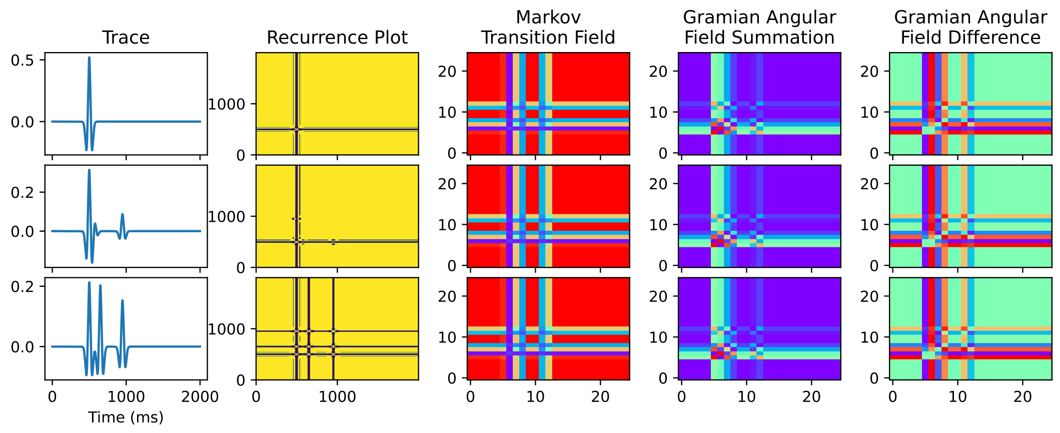


Figure 5.4: Alternative representations for a trace.

For theory-guided RNN, excluding part of the data from training for use as a development dataset is standard practice in deep learning, but not within classical FWI. For a real-world problem, the size of the seismic dataset relative to the model parameters generally has fewer data samples and could potentially prove problematic. Hyperparameter tuning for the optimal parameters for RNN demonstrate that practice can result in convergence to a good model, yet this does not prove a similar result is achievable when using the entire dataset.

5.1.3 | Forward Modelling and Multiples

All shots considered within the forward problem for either FWI and RNN framework were within the water column for the Marmousi model. This implies that receiver data have surface-related multiples, together with all other inter-layer multiple components. Undergoing forward problem solving with and without multiples is a decision that the literature is still unable to resolve.

Multiples travel longer paths and are reflected at small angles in contrast to the primaries and are able to illuminate shadow zones where primary reflections cannot reach (Bergen et al., 2019). Inclusion of these wavefield components can lead to improvement within the inversion process as multiples can contain more subsurface information compared to primary and diving wave (Komatitsch and Tromp, 2002). Bleibinhaus and Rondenay (2009) studied these effects of surface scattering in FWI and concluded that velocity models resulting from neglecting the free surface in the inversion show artifacts and suffered from a loss of resolution. Liu et al. (2020) employ a combination of lower-order multiple as the source and the higher-order multiple to invert, whilst Zhang et al. (2013) transform each hydrophone into a virtual point source with a time history equal to that of the recorded data to help their inversion and are able to produce methods utilizing multiples to improve velocity updates.

Hicks and Pratt (2001) and Operto et al. (2006) demonstrated that traditional FWI would become unstable when it inverts observed data with the free surface related waves. This said, removing of multiples introduce additional processing steps which are subject to error and could lead to removal of signal. The consensus is that the choice of multiple inclusion is per different use case. Indeed, the work presented could be revisited for the sensitivity of multiples within the inversion.

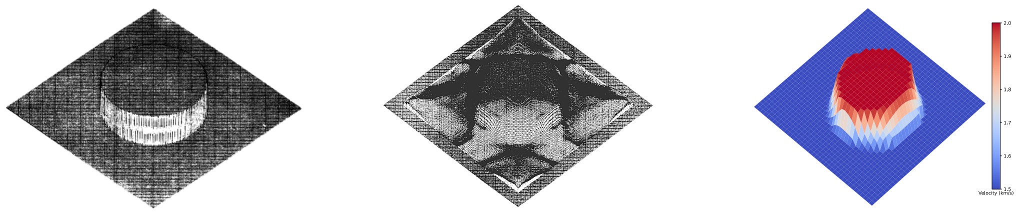
5.1.4 | Implications of Data Volume and Computational Power

More data is directly correlated with better modelling for NN frameworks, and this ability is limited by the resources available. Similar to classical FWI, computational power is a limitation within the frameworks presented. This was already identified within the RNN approach with the limit from the Graphical Processing Unit RAM, constraining the model size and batch processing. A larger batch-size for RNN processing would intuitively imply that the optimization is less likely to get stuck with local minimum and reduce the probability of cycle-skipping. Workaround for this could be multi-Graphical Processing Unit systems, such as NVIDIA's DGX station[1] and Lambda Lab's Vector station[2], or cloud computing such as Amazon Web Services [3] and Google Cloud[4,5]. This cost on memory requirements for NNs is a common issue with solving optimization of large-scale neural networks (Bottou et al., 2018) and efforts have been made into unconventional training methods such as alternating gradient direction methods and Bregman iterations (Boyd et al., 2011; Taylor et al., 2016).

[1]`https://www.nvidia.com/en-us/data-center/dgx-systems/`
[2]`https://lambdalabs.com/gpu-workstations/`
[3]`https://aws.amazon.com/nvidia/`
[4]`https://cloud.google.com/gpu/`
[5]These are just samples of resources which are readily available. There is no affiliation with either of the products mentioned.

5.1.5 | Maturity of the Frameworks

The NN framework results presented in this work should be considered within the context of maturity of the technique. As indicated in the literature review, FWI was originally an academic pursuit and initial inversions results were naive. Figure 5.5 shows results for FWI inversion obtained by Gauthier et al. (1986) for a circular model. Inverting for 8 shots for 5 iterations, with 400 receiver locations, the model is able to recover some structure within the velocity - Figure 5.5(b). Inverting a similar model within a modern FWI framework would produce better global inversion as shown in Figure 5.5(c). The legacy inversion by Gauthier et al. took approximately 1 hour to execute on a CRAY-1S supercomputer (Kolodzey, 1981), whilst modern FWI was run on a personal computer for 2 minutes. This is a major uplift, which is backed by thirty-fives years of advances in hardware, computational modelling, mathematics, geophysics and optimization theory. Taking a similar time consideration for the NN framework, the potential for data-driven pseudo-spectral approach is still in infancy, yet able to produce improvements to a mature technique.



(a) Initial circular model. (b) Legacy FWI from 1986. (c) Modern FWI.

Figure 5.5: Comparison between inversion results for legacy FWI and modern FWI for a circular model. 35 years of advances in hardware, computational modelling, mathematics, geophysics and optimization theory produce major uplift at the fraction of time.

Both frameworks are still to be implemented to higher dimensions. The DNN approach would follow similarly to the work by Liu et al. (2020). 2D receivers are modelled for an arbitrary velocity models and 2D convolutional architectures are trained to invert from the seismic data to the velocity model. Naturally, the next step would be 3D geometry. In the case of theory-guided approaches, addition of this axis is expected to provided better imaging.

5.1.6 | Other Areas of Deep Learning

The intersection of Deep Learning and FWI is just starting to flourish and offers ample opportunity for new research avenues. Alternative architectures are readily available to be implemented. Some examples for DNN are shown in Figure 3.6. Transformers (Vaswani et al., 2017) are currently state-of-the-art sequence learners and Shalova and Oseledets (2020) developed the initial theoretical work for implementation for FWI. Alternatively, Fourier Recurrent Units (Zhang et al., 2018) could be used within theory-guided FWI. These units stabilize training gradients along the temporal dimension with Fourier basis functions and would potentially resolve the gradient anomaly within our work.

The technique of Transfer Learning within Deep Learning has just starting to be considered in geophysics. Indeed, given that DNN generators are stochastic global engines, a pre-trained NN for a survey could easily be employed on another seismic survey with minimal training. Siahkoohi et al. (2019) successfully used the probability distributions of nearby surveys around an area of interest to fine-tune a pre-trained Generative Adversarial Network and be able to map low-cost, low-fidelity solutions to remove surface-related multiples and ghost from shot records and numerical dispersion from receivers.

Yadav et al. (2015) presented the topic of solving differential equations directly using NN architectures and bypassing the finite-difference approach. This framework has been found to be advantageous since (i) the solutions obtained are integrable and differentiable, thus have better interpretability, (ii) solutions are highly generalized and preserves accuracy despite very few points, and (iii) no modification is needed for different kinds of boundary conditions. Indeed, Zhu et al. (2020) developed a neural-network-based full waveform inversion method that integrates deep neural networks with FWI by representing the velocity model with a generative neural network. The velocity model generated by NN is input to conventional FWI partial differential equation solvers. The gradients of both the NN and PDEs are calculated using automatic differentiation, which back-propagates gradients through the acoustic/elastic PDEs and NN layers to update the weights and biases of the generative neural network.

5.2 | Conclusion

Data-driven and theory-guided approaches for FWI have been reviewed with a comprehensive study of literature. The pseudo-spectral approach via deep learning framework was identified to be lacking in any previous work and this proved to be an opportunity for development. FWI was re-cast within a DNN framework for both a data-driven and a theory-guided based formulation. Both approaches were developed theoretically, qualitatively assessed on synthetic data and tested on the Marmousi dataset.

Elements within a classical FWI framework were shown to be substitutable with DNN components. The base architecture for the network was set to be an hour-glass neuron design. This is representative of multi-scale FWI and modern DNN approaches. Two data-generators were used to validate the framework on multi-layer models. This was tested for normalization and concluded not applicable. Fully connected layers, 1D and 2D convolutions, VGG and ResNet type architectures for Adagrad, Adadelta, RMSprop and Adam optimizer were quantitatively evaluated for computational hours, DNN training performance, validation and learning rate performance, and inversion RMSE. The best performing architecture-loss combination was identified as Conv1D-Adadelta. Conv1D architecture ranked the highest in all tests, whilst the differences in optimizer were superficial. The choice of architecture was the most important aspect as choosing a different loss optimizer would not result in deterioration of the result.

The Conv1D-Adadelta network was trained for inversion of the Marmousi model by using a 20-50 layer generator, with velocities ranging from 1450ms^{-1} to 5000 ms^{-1}. It was trained for 30 epochs, at 1,000,000 and 100,000 trace per epoch per training and testing dataset respectively. Analysis of the network kernel and bias distribution for the training and velocity and trace update per epoch confirmed that the framework is a learning process. Most of the update happen within the first few epochs, whilst additional epochs refined the inverted velocity. Multi-scale 3.5Hz classical FWI with Sobolev space norm regularization was compared to this DNN inversion. Inversion performance in shallow sections was equally good for either classical FWI or DNN approach. DNN framework performs better for deeper and over-thrust areas since DNNs are not bound by forward-modelling physical constraints from ray-tracing.

Theory-guided RNN as an analogue of FWI was implemented for 2D experiments and different wavefield components compared to an analytical 2D Green's function and time implementation. Based on these results, RNN Time is able to model the wavefield within a maximum 0.06 error tolerance and 1.74% RPE. RNN Freq is overall more accurate with 0.05 error tolerance and 1.449% RPE. Assessment on the gradients indicates how the adjoint state and RNN Freq gradients in general overestimate finite difference calculation, whilst RNN Time under-estimates it with an infinitesimal error. RNN Freq produced a perturbation on the onset of the gradient which was attributed to modelling artefact and could be mitigated in future versions of this approach. Based on the model size and compute available, the ideal loss was Adam with a learning rate of 2 and batch size of 1. Model batch size proved to be a limitation for practical implementations, yet RNN is computationally more efficient than the classical FWI presented in this work. RNN freq provides more stable convergence and is better performant. Overall, RNN frameworks are able to identify faults, but amplitudes are not fully inverted properly.

From the comparative analysis of the NN approaches, DNN was better performing. DNN networks recovered more of the velocity contrast, whilst RNN was better at edge definition. RNN was more suited for the shallow and depth sections due to the cleaner receiver residuals. This was only valid in the middle section of the model with the most coverage from ray-tracing. Indeed, RNN approaches had some leakage on the edges.

A critique addressing benefits and limitations of these two approaches was also included. The impact of the shift in the inversion paradigm was reviewed for both approaches. Data-driven should be considered within global approximation approaches and has potential to be used as *a priori* to deterministic FWI. Research which can be easily translated was addressed in the form of probabilistic maps for the inversions and analytical accuracy upper bounds per iteration. The RNN approach benefits from the wider community of active researchers. The reduction in development time is a direct integration from Computer Science to geophysics. Vice-versa, Deep Learning frameworks can adopt strategies common to FWI.

Data-driven inversion is still at the very early stages of development, and more research opportunity is still available. However, to truly assess the applicability and relevance of these frameworks, these approaches will have to be applied to real data in the future. The DNN model generators were shown to work within the boundaries of their parameters. Extra layers, velocity inversions and inclusions could be missed altogether if the network is not prepared correctly. Application of pre-trained networks is relatively easy and thus different geophysical model hypothesis could be assessed quickly. In either case, the DNN is robust to noise and future work would involve implementation as a de-noising techniques. Alternative modelling methods and transform spaces were also provided as alternative approaches for DNN.

The forward modelling approach used through this work was critiqued for the use of multiples. Whether to use or not to use multiples within forward modelling is model dependent and should be evaluated for RNN Freq. Similar to classical FWI, computational power was identifiable as a limitation within these DNN frameworks. Although this is currently a limitation, it will not be in the near future due to the relative quick development of GPUs. A corollary to the whole approach was addressed in the form of the maturity of the approach. 35 years of advances applied to these frameworks would be expected to yield very good results. Finally, other areas of DNN that can be applied to FWI were presented. Alternative architectures such as Transformers and use of Fourier Recurrent Units are readily available. Potential of transfer learning and solving differential equations using NN were presented as future directions of research for these frameworks.

A

Additional Theoretical Tools

A.1 | Difference between Time and Frequency FWI

To illustrate the differences within numerical implementations between time and frequency FWI, let us consider the 1D finite difference formulation of the acoustic wave equation given by:

$$\frac{1}{c(x)^2}\frac{\partial^2 p(x,t)}{\partial t^2} - \nabla^2 p(x,t) = s(x,t). \tag{A.1}$$

Following from Igel (2017), Equation A.1 is re-arranged and re-written in dense notation as:

$$\partial_t^2 p = c^2 \partial_x^2 p + s. \tag{A.2}$$

Discretize space and time with constant increments dx and dt such that

$$x_j = jdx, \qquad j = [0, l_{max}] \tag{A.3}$$

$$t_n = ndt, \qquad t = [0, n_{max}]\,. \tag{A.4}$$

Implementing a FD stencil as shown in Figure A.1, the derivatives approximated via central finite differences and Equation A.2 can be written as:

$$\frac{p_j^{n+1} - 2p_j^n + p_j^{n-1}}{dt^2} = c_j^2 \left[\frac{p_j^{n+1} - 2p_j^n + p_j^{n-1}}{dx^2}\right] + s_j^n, \tag{A.5}$$

where the upper index corresponds to time discretization and the lower index corresponds to spatial discretization. Pressure at location $n+1$ based on current location n and previous $n-1$ location is given by:

$$p_j^{n+1} = c_j^2 \left[p_j^{n+1} - 2p_j^n + p_j^{n-1}\right] + 2p_j^n - p_j^{n-1} + dt^2 s_j^n. \tag{A.6}$$

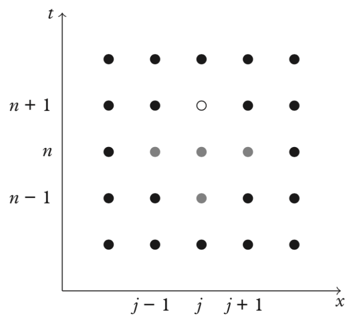


Figure A.1: Space-Time FD discretization stencil for the 1D acoustic wave equation. The x-axis corresponds to space and the y-axis to time. The open circle denotes the point $p(x_j, t_{n+1})$ to which the state of the pressure field is extrapolated. From Igel (2017).

Given a source wavelet s_j^n at location j and time n, an initial wavefield p_j^0 and initial conditions such that everything is at rest at time $t = 0$, all components on the right-hand side of Equation A.6 are known and can be used to propagate particle motion through an acoustic medium.

Considering the Fourier pseudo-spectral implementation (Gazdag, 1981), Fourier theory seeks to approximate a given function by a finite sum over some N orthogonal basis functions Φ_i, namely:

$$f(x) \approx \sum_{i=0}^{N} a_i \Phi_i(x), \tag{A.7}$$

where a_i are the Fourier coefficients. The main benefit of using this approximation is that given the discrete Fourier transform of functions defined on a regular grid, exact machine precision derivatives up to the Nyquist wavenumber $k_N = \frac{\pi}{dx}$ can be calculated. In particular, it can be shown that the Fourier transform operator $\mathcal{F}$ can obtain the exact n^{th} derivate as:

$$f^n(x) = \mathcal{F}^{-1}\left[(ik)^n \mathcal{F}\left[f(x)\right]\right]. \tag{A.8}$$

Substituting Equation A.8 in the 2^{nd} order spatial derivative of the 1D acoustic wave equation in Equation A.2 gives:

$$\partial_x^2 p_j^n = \mathcal{F}^{-1}\left[(ik)^2 P_\nu^n\right] = \mathcal{F}^{-1}\left[-k^2 P_\nu^n\right], \tag{A.9}$$

where P_ν^n is the discrete complex wavenumber spectrum at time n. As such, the main overall error in the numerical solution comes from only the time integration scheme. Substituting Equation A.9 in Equation A.2, discretize on a FD stencil and rearranging yields:

$$p_j^{n+1} = dt^2\left[c_j^2 \mathcal{F}^{-1}\left[k^2 P_\nu^n\right] + s_j^n\right] + 2p_j^n - p_j^{n-1}. \tag{A.10}$$

A.2 | Activation Functions

Below are popular activations functions based on Goodfellow et al. (2016). These are represented graphically in Figure A.2.

- **Binary Step**: $f(x) = \begin{cases} 1, & \forall x \geq 0 \\ 0, & \forall x < 0 \end{cases}$. This a simple switch on-off type function ideal for a binary classification. It is more theoretical than practical as data classifications tasks usually classify into multiple classes. Furthermore, a binary step function has a vanishing gradient, thus making it not usable with back-propagation.

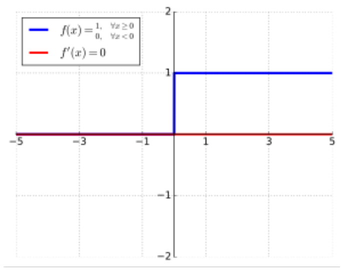
- **Linear**: $f(x) = \alpha x, \forall x \in \mathbb{R}$. This is proportional to the input x, based on the scalar α. It can be applied to various neurons and multiple neurons can be activated at the same time. However, the gradient would be constant at α, resulting in a linear transformation.

- **Sigmoid**: $f(x) = \frac{1}{1+e^{-x}}$. This is widely used function since it is a non-linear smooth continuously differentiable function. The non-linearity enables neurons to approximate non-linear functions. The gradient is always positive, however it still has a vanishing gradient problem as $f^{'}(x)$ approaches zero for larger x. Furthermore, the sigmoid function is non-symmetric and connectivity between neurons is not necessarily always positive. Mitigation is achieved via scaling of the sigmoid, resulting in the Tanh function.

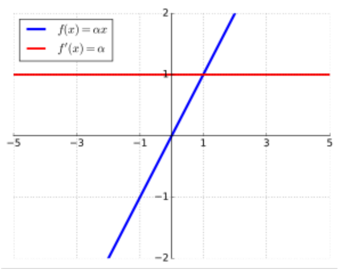
- **Tanh**: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. This function caries the same properties of non-linearity, smoothness and continuous differentiability of the sigmoid function, however it has the added benefit of being symmetric over the origin. This mitigates the problem of having all values of the same sign. Nonetheless, Tanh still retains the vanishing gradient problem as the function is flat with small gradients for large x.

- **Softmax**: $f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$. This function is the sigmoid function extended to multiple classes and provides the probability of the input being in a particular class as an output. The softmax function is ideal when trying to attain the probabilities to define the class of each input.

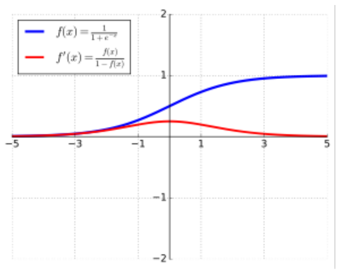
- **ReLU**: $f(x) = max(0, x)$. The Rectified Linear Unit is the most widely used activation function. It is non-linear, allowing easy back-propagation of errors. When employed on a network of neurons, the negative component of the function is converted to zero and the neuron is deactivated, thus introducing sparsity with the network and making it efficient and easy for computation.



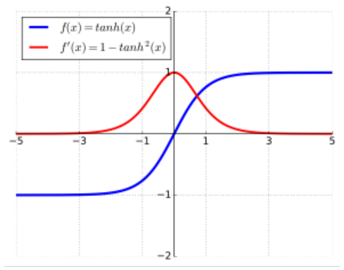
(a) Binary Step



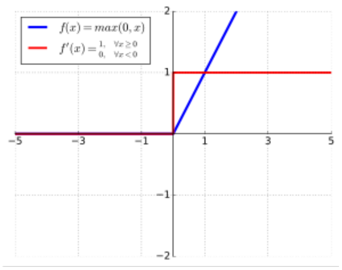
(b) Linear



(c) Sigmoid



(d) Tanh



(e) ReLU

Figure A.2: Activation functions $f(x)$ and their derivative $f^{'}(x)$.

A.3 | Loss Optimizers

To simplify notation in this section, consider objective function $\mathcal{J}(\Theta)$ parameterized by model parameters $\Theta \in \mathbb{R}^d$ to optimize gradient $\nabla_\Theta \mathcal{J}(\Theta)$ with respect to the model parameters for a learning rate η. This can be recast as:

$$\Theta \equiv \Theta - \eta \nabla_\Theta \mathcal{J}(\Theta). \tag{A.11}$$

A.3.1 | Adagrad

The Adaptive Gradient or Adagrad algorithm (Duchi et al., 2011) performs learning rate updates with training based on the frequency of distribution in the data. It performs smaller updates (smaller learning rates) on parameters associated with frequently occurring features and larger updates vice-versa.

As Adagrad uses a different learning rate for every parameter Θ_i at time step t, the gradient of the objective function is given by

$$g_t = \sum_i \nabla_{\Theta_t} \mathcal{J}(\Theta_{t,i}). \tag{A.12}$$

Model updates at time $t + 1$ become:

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t, \tag{A.13}$$

where G_t is the sum of squares of the gradients up to time step t and ϵ is a stabilizing term to avoid division by zero. This optimizer eliminates the need to manually tune the learning rate. However, the accumulation of the squared gradients in the denominator keeps growing with every update and will eventually cause the learning rate to shrink up to the point the algorithm is no longer able to acquire additional knowledge (Ruder, 2016).

A.3.2 | Adadelta

Adadelta (Zeiler, 2012) extends Adagrad and aims to avoid the continual decay of the learning rates by using a decaying average of past gradients to some fixed size w. Equation A.13 becomes

$$\Theta_{t+1} = \Theta_t - \frac{\text{RMS}\left[\Delta\Theta\right]_{t-1}}{\text{RMS}\left[g\right]_t} g_t, \tag{A.14}$$

where $RMS\left[g\right]_t$ is the Root Mean Squared for the gradient, $RMS\left[\Delta\Theta\right]_{t-1} = \sqrt{E\left[\Delta\Theta^2\right]_{t-1} + \epsilon}$ is the Root Mean Squared of parameter updates. Since $RMS\left[\Delta\Theta\right]_t$ is unknown, it is approximated until the previous time step $t - 1$. This algorithm removes the need for a default learning rate.

A.3.3 | RMSprop

Similar to Adadelta, RMSprop (Hinton et al., 2012) attempts to resolve Adagrad's vanishing learning rates by maintaining a moving average of the square of gradients and divides the gradient by the root of this average. Equation A.13 becomes:

$$E\left[g^2\right]_t = 0.9E\left[g^2\right]_{t-1} + 0.1g_t^2 \tag{A.15}$$

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t. \tag{A.16}$$

A.3.4 | Adam

Adaptive Moment Estimation or Adam (Kingma and Ba, 2014) computes adaptive learning rates for each parameter based on estimations of first-order and second-order moments. The algorithm updates exponential moving averages of the gradient $(\hat{m}_t)$ and the squared gradient ($\hat{v}_t$) by

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{A.17}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{A.18}$$

where the hyper-parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages. Equation A.13 for Adam becomes:

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \tag{A.19}$$

A.4 | Regularization

DNN regularization is available via two strategies: (i) functional-based, and (ii) NN architecture-based.

A.4.1 | Functional-based

These type of schemes update the cost function (Equation 3.11) with a regularization term, namely:

$$\hat{\mathcal{J}} = \mathcal{J} + \alpha\Omega, \tag{A.20}$$

where $\mathcal{J}$ is the original cost function, Ω is the regularization term or norm penalty and $\alpha \in [0, \inf)$ is a hyperparameter that weights the relative contribution of $\mathcal{J}$ to Ω. The most commonly used norm penalty is $L^2 - norm$ defined as:

$$\Omega_{L^2} = \frac{1}{2} ||w||_2^2, \tag{A.21}$$

where w are the weights throughout the DNN. This lends itself from the broader subject of regularization theory and was previously identified for FWI in § 3.1.2. Alternative names for this style of regularization are Ridge Regression or Tikhonov regularization.

A.4.2 | Architecture-based

DNN can be regularized via alterations to the NN architecture through (i) Dropout, (ii) Data augmentation, and (iii) Early Stopping.

A.4.2.1 | Dropout

Dropout is a very simple ensemble-based technique (Srivastava et al., 2014). It is the process of allow only some parts of the network to be updated. Figure A.3 illustrates all sub-networks that can be formed from a simple NN that allow weight update within an epoch. Either of these sub-networks is randomly considered and trained at a given epoch and inherently remove nodal dependencies (Srivastava et al., 2014).

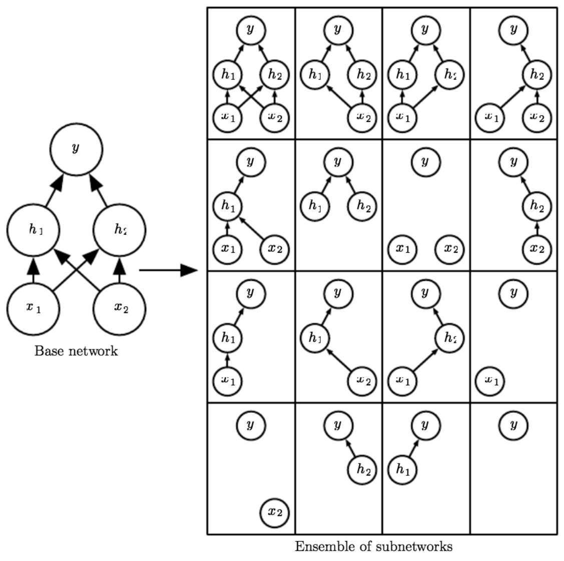


Figure A.3: All 16 possible sub-networks are produced from a simple base network. A large proportion of these do not have sufficient input-to-output connections and are ignored. However, for wide and deeper DNNs, these types of sub-networks become insignificant since the probability of dropping all possible paths from inputs to outputs becomes negligible. If such an occurrence does happen, it would only effect a single epoch and this would be mitigated with long enough training. From Goodfellow et al. (2016).

A.4.2.2 | Data Augmentation

The best way to make a DNN model generalize better for "unseen" data is to train on more data (Goodfellow et al., 2016). The amount of data available in real-world application is limited and it is reasonably straight forward to create new "fake" data. This is achieved via transformations of the original x input and mapping to the correct y (Lee and Moloney, 2018). Keivan Ekbatani et al. (2017) and Le et al. (2017) show successful training of a DNN using synthetic created input-output pairs in practice.

A.4.2.3 | Early Stopping

When DNN model complexity is sufficient to represent over-fitting, it is likely that the training error decreases steadily over time, but validation set error begins to rise again and makes the model worse off with any additional iteration (Yao et al., 2007). Early stopping is the technique which terminates training at a given epoch $\mathcal{E}^n$ if the errors on the validation sets at $\mathcal{E}^{n+1}$ are greater than those of $\mathcal{E}^n$. Early stopping regularization is the most widely used regularization technique as it is unobtrusive to the learning dynamics for DNN (Caruana et al., 2000).

A.5 | CNN Building Blocks

This section reviews the elements for CNNs referenced through-out the dissertation.

A.5.1 | Convolutional Layer

A convolutional layer differs from a normal fully connected layer through the introduction of convolutions across the neurons. The window across which convolutions are applied is referred to as a kernel or filter. Mathematically, the convolutional operation on an input image tensor I_c for the l^{th} layer is given by:

$$f_l^k(p,q) = \sum_c \sum_{x,y} i_c(x,y) e_l^k(u,v), \tag{A.22}$$

where $i_c(x,y)$ is an element of the input image tensor, $e_l^k(u,v)$ is the index of the k^{th} convolutional kernel k_l, and the output or feature-map of the k^{th} convolutional kernel is given by $F_l^k = \left[f_l^k(1,1), \cdots, f_l^k(P,Q)\right]$ with P and Q being the total number of rows and columns in I_C. A graphical representation of the convolutional operation is given in Figure A.4.

A.5.2 | Pooling Layer

Once a feature-map is extracted from a convolutional layer, the exact location of one element becomes less important as long as its relative position to others is preserved (Khan et al., 2020). Sub-regions of interest can thus be inferred via reduction of dimensionality (Lee et al., 2016). There are two main types mechanisms, max and min pooling. Max pooling picks the maximum value and min pooling picks the minimum value from the assigned region. Figure A.5 illustrates an example of max pooling.

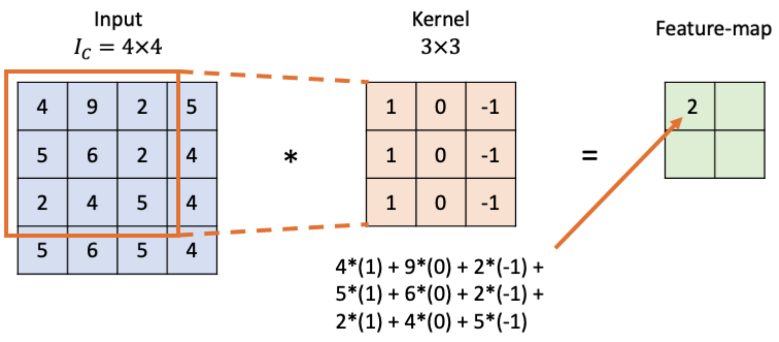


Figure A.4: Example of 2D convolution.

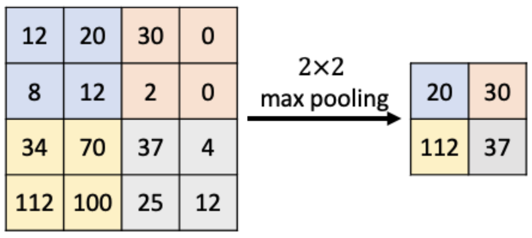


Figure A.5: Example of 2D max-pooling.

Either of these pooling operations support feature extraction which is invariant to translational shifts and small distortions (Ranzato et al., 2007). These invariant features regulate the complexity of the network and reduce over-fitting. Alternative formulations such as average, L2, overlapping, spatial pyramid pooling have also been used for CNNs (Boureau et al., 2010; He et al., 2015; Wang et al., 2012).

A.5.3 | Batch Normalization

Input distributions to the layers in deep network may vary when weights are updated during training. This covariance shift in the distribution of hidden unit values slows down the convergence by forcing learning rate to small values. Batch Normalization is used to counteract this internal covariance shift by standardising the input to zero mean and unit variance for each batch (Ioffe and Szegedy, 2015). It smoothens the flow of gradient and acts as a regulating factor (Santurkar et al., 2018). Batch normalization for a transformed feature-map F_l^k is given by

$$\mathcal{N}_l^k = \frac{\mathcal{F}_l^k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{A.23}$$

where $\mathcal{N}_l^k$ is the normalized feature map, $\mathcal{F}_l^k$ is the input feature map, μ_B is the mean, σ_B^2 is the variance for a batch and ϵ is added for numerical stability (Ioffe and Szegedy, 2015).

A.6 | Common CNN Architectures

A.6.1 | AlexNet

AlexNet (Krizhevsky et al., 2012) is considered as the first deep neural network with 8 layers, compared to its predecessor LeNet with 5 layers (LeCun et al., 1990). In particular, it has large filters $(11 \times 11, 5 \times 5)$ at the initial layers, dropout during training to enforce the model to learn more robust features, ReLU to improve convergence rates and overlapping sub-sampling and local response normalization to improve generalization. The basic architectural blueprint is given in Figure A.6(a).

A.6.2 | VGG

VGG (Simonyan et al., 2014) is a rather simplistic, homogenous network. However, it is 19 layers deep. It stacks (3×3) filters to reduce the computational complexity, uses max pooling after the convolutional layers and padding is applied to maintain the spatial resolution. The only problem associated with VGG network are the 138 million trainable parameters, which make it computationally expensive for network training and deploying on system with low resources. The architectural blueprint is given in Figure A.6(b).

A.6.3 | ResNet

ResNet (He et al., 2016b) is a 152-layers deep network shown in Figure A.6(c). ResNet is 20 times deeper than AlexNet and 8 times deeper VGG. Pivotal to this network is the concept of a residual block, shown in Figure A.6(d). This introduces the "identity map" or "skip connection" that skips one or more layers. This propagates the error deep through the network layers and allows it to learn how to minimize the residual.

A.7 | RNN Graphs and Unfolding

RNNs are capable of learning sequentially representing dynamic temporal behaviour (Rumelhart et al., 1986). These can be categorized into two broad classes: (i) finite impulse with directed acyclic graphs, and (ii) infinite impulse with directed cyclic graphs (Sherstinsky, 2020). A finite impulse RNN is a directed acyclic graph that can be unrolled and replaced with a strictly feed-forward NN, whilst an infinite impulse RNN is a directed acyclic graph which cannot be unrolled (Sherstinsky, 2020). Given the directionality of time in wave propagation (i.e. the wavefield at the time t is not affected by future wavefield values $t+1$ and time t is only affect by the previous step $t-1$), a finite impulse directed acyclic graph is suitable for wave simulation. “Unfolding” of a RNN is a useful technique to visualize directed acyclic graphs. Figure A.7 shows part of an unfolded RNN.

A.8 | LSTM Components

A.8.1 | Forget gate

The forget gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

A.8.2 | Input gate

The input gate i_t accepts the previous hidden state and current input into a sigmoid function. This activation function decides which values will be updated by transforming the values to be between 0 and 1, with 0 being not important and 1 being more important. The current input and previous hidden state are passed into the tanh function to squeeze values between -1 and 1 and get a potential new candidate.

A.8.3 | Cell state

The cell state C_t acts as a mechanism to transfers information all the way down the sequence and can carry relevant information throughout the processing of the sequence. This enables information from earlier time steps to be available at later time steps, thus reducing the effects of short-term memory. The preservation over time of gradient information by LSTM is illustrated in Figure A.8. As the cell state goes through the chain structure of the LSTM sequence, information from the input and forget gates are multiplied and only updates the cell states to values that the neural network learns to be relevant.

A.8.4 | Output gate

The output gate decides what the next hidden state should be and what is to be used for predictions. It accepts the previous hidden state and current cell input into a sigmoid function. The modified cell state is passed into a tanh function which will learn to decide which part of the data should be pushed forward through the sequence. The output is the hidden state. The new cell state and the new hidden are carried over to the next time step.

A.9 | Automatic Differentiation and Adjoint State

Following from Richardson (2018), consider the 1D scalar wave equation consisting of only one shot with one receiver. Using cost function in Equation 3.11, the gradient with respect to one wavefield time step is

$$\frac{\partial J}{\partial \mathbf{u}_t} = \frac{\partial J}{\partial \mathbf{u}_t} + \frac{\partial J}{\partial \mathbf{u}_{t+1}} \frac{\partial \mathbf{u}_{t+1}}{\partial \mathbf{u}_t} + \frac{\partial J}{\partial \mathbf{u}_{t+2}} \frac{\partial \mathbf{u}_{t+2}}{\partial \mathbf{u}_t}, \tag{A.24}$$

where $\frac{\partial \mathcal{J}}{\partial \mathbf{u}_t}$ indicates the row vector of partial derivatives of $\mathcal{J}$ with respect to elements of $\mathbf{u}_t$, while the wave speed $\mathbf{c}$ and the wavefields from other time steps $\mathbf{u_{t' \neq t}}$ are held constant. $\frac{\partial \mathcal{J}}{\partial \mathbf{u}_{t+1}}$ includes all changes to $\mathcal{J}$ caused by changes of the wavefield at a previous $\mathbf{u}_{t+1}$ and a later time step $\mathbf{u}_{t+1}$.

Considering Automatic differentiation, the gradient of the cost function with respect to the wave speed is obtained by chaining different derivates at different time steps via the chain rule. Namely,

$$\frac{\partial \mathcal{J}}{\partial \mathbf{c}} = \sum_{t=1}^{N_t} \frac{\partial \mathcal{J}}{\partial \mathbf{u}_t} \frac{\partial \mathbf{u}_t}{\partial \mathbf{c}}. \tag{A.25}$$

Consider $N_t = 4$ as an example. The steps to calculate the gradient of $\mathcal{J}$ would be

$$\frac{\partial \mathcal{J}}{\partial \mathbf{u}_4} = \frac{\partial \mathcal{J}}{\partial \mathbf{u}_4}, \tag{A.26}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{u}_3} = \frac{\partial \mathcal{J}}{\partial \mathbf{u}_3} + \frac{\partial \mathcal{J}}{\partial \mathbf{u}_4} \frac{\partial \mathbf{u}_4}{\partial \mathbf{u}_3}, \tag{A.27}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{u}_2} = \frac{\partial \mathcal{J}}{\partial \mathbf{u}_2} + \frac{\partial \mathcal{J}}{\partial \mathbf{u}_3}\frac{\partial \mathbf{u}_3}{\partial \mathbf{u}_2} + \frac{\partial \mathcal{J}}{\partial \mathbf{u}_4}\frac{\partial \mathbf{u}_4}{\partial \mathbf{u}_2}, \tag{A.28}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{u}_1} = \frac{\partial \mathcal{J}}{\partial \mathbf{u}_1} + \frac{\partial \mathcal{J}}{\partial \mathbf{u}_2}\frac{\partial \mathbf{u}_2}{\partial \mathbf{u}_1} + \frac{\partial \mathcal{J}}{\partial \mathbf{u}_3}\frac{\partial \mathbf{u}_3}{\partial \mathbf{u}_1}. \tag{A.29}$$

The required partial derivatives are given by

$$\frac{\partial \mathcal{J}}{\partial \mathbf{u}_t} = 2(\delta_{x_r}^T \mathbf{u}_t - d_t)\delta_{x_r}^T, \tag{A.30}$$

$$\frac{\partial \mathbf{u}_{t+1}}{\partial \mathbf{u}_t} = \mathbf{c}^2\Delta_t^2 \mathbf{D_x}^2 + 2, \tag{A.31}$$

$$\frac{\partial \mathbf{u}_{t+2}}{\partial \mathbf{u}_t} = -1, \tag{A.32}$$

$$\frac{\partial \mathbf{u}_t}{\partial \mathbf{c}} = 2\mathbf{c}\Delta_t^2 \left(\mathbf{D_x}^2 \mathbf{u}_{t-1} - \mathbf{f}_{t-1}\right). \tag{A.33}$$

Substituting into Equation A.25 yields

$$\frac{\partial \mathcal{J}}{\partial \mathbf{c}} = \sum_{t=1}^{N_t} \left(2(\delta_{x_r}^T \mathbf{u}_t - d_t)\delta_{x_r}^T + \frac{\partial \mathcal{J}}{\partial \mathbf{u}_{t+1}}(\mathbf{c}^2\Delta_t^2\mathbf{D_x}^2 + 2) - \frac{\partial \mathcal{J}}{\partial \mathbf{u}_{t+2}} \right) \tag{A.34}$$

$$\times 2\mathbf{c}\Delta_t^2 \left(\mathbf{D_x}^2 \mathbf{u}_{t-1} - \mathbf{f}_{t-1}\right)$$

$$= \sum_{t=1}^{N_t} \left(\mathbf{c}^2\Delta_t^2 \left(\mathbf{D_x}^2 \frac{\partial \mathcal{J}}{\partial \mathbf{u}_{t+1}} + \frac{2}{\mathbf{c}^2\Delta_t^2}(\delta_{x_r}^T \mathbf{u}_t - d_t)\delta_{x_r}^T \right) \right. \tag{A.35}$$

$$\left. +2\frac{\partial \mathcal{J}}{\partial \mathbf{u}_{t+1}} - \frac{\partial \mathcal{J}}{\partial \mathbf{u}_{t+2}} \right) 2\mathbf{c}\Delta_t^2 \left(\mathbf{D_x}^2 \mathbf{u}_{t-1} - \mathbf{f}_{t-1}\right) \tag{A.36}$$

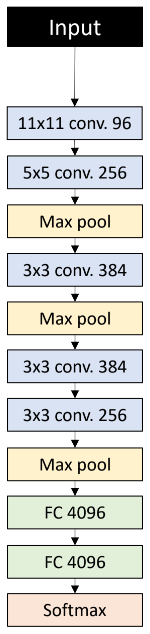
$$= \sum_{t=1}^{N_t} -F^{-1}\left(\frac{2}{\mathbf{c}^2\Delta_t^2}(\delta_{x_r}^T \mathbf{u}_t - d_t)\delta_{x_r}^T \right) 2\mathbf{c}\Delta_t^2 \left(\mathbf{D_x}^2 \mathbf{u}_{t-1} - \mathbf{f}_{t-1}\right), \tag{A.37}$$

where $F^{-1}(\mathbf{g}_t)$ is wave propagation backward in time of the source amplitude $\mathbf{g}_t$. Recognize that the wave equation can be used to express the factor on the right as the second time derivative of the forward propagated source wavefield,

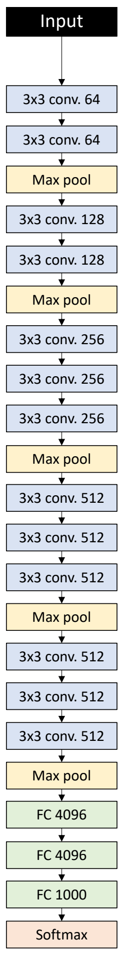
$$\frac{\partial \mathcal{J}}{\partial \mathbf{c}} = \sum_{t=1}^{N_t} -F^{-1}\left(\frac{2}{\mathbf{c}^2\Delta_t^2}(\delta_{x_r}^T \mathbf{u}_t - d_t)\delta_{x_r}^T \right) \frac{2\Delta_t^2}{\mathbf{c}}\frac{\partial^2 \mathbf{u}_{t-1}}{\partial t^2} \tag{A.38}$$

$$= \sum_{t=1}^{N_t} -F^{-1}\left(2(\delta_{x_r}^T \mathbf{u}_t - d_t)\delta_{x_r}^T \right) \frac{2}{\mathbf{c}^3}\frac{\partial^2 \mathbf{u}_{t-1}}{\partial t^2}, \tag{A.39}$$

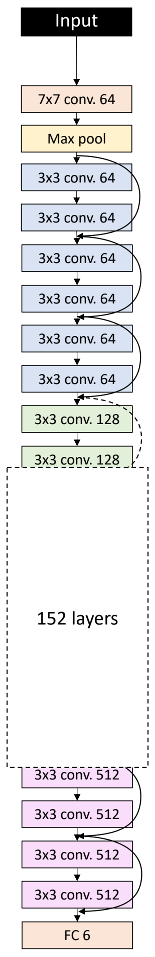
where the linearity of wave propagation is used to take factors out of the source amplitude of the left term. The result is the same equation that is used in the adjoint state method.



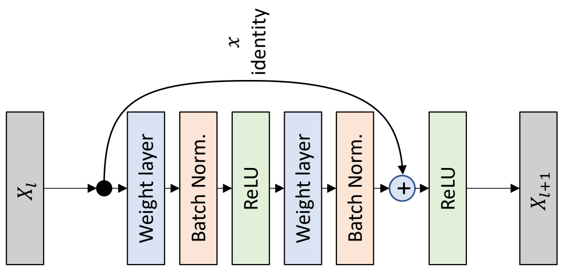
(a) AlexNet: Five convolutional and three fully connected layers.



(b) VGG: Five stacked conv. layers with max pooling and 3 fully connected layers.



(c) ResNet: 152 layer deep network with residual blocks.



(d) Residual block introduces an "identity map" or "skip connection" which skips one or more layers.

Figure A.6: Architectural blueprints for (a) AlexNet, (b) VGG, (c) ResNet and (d) Residual Block for ResNet. The numbers in each layer in (a)-(c) indicate the number of trainable parameters. Adapted from Krizhevsky et al. (2012), Simonyan et al. (2014) and He et al. (2016b).

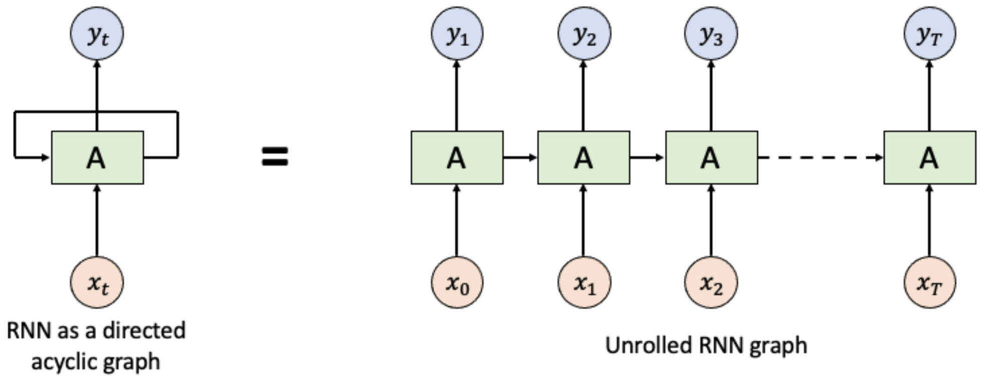


Figure A.7: RNNs can be represented as directed acyclic graphs. Chuck of the NN A looks at some input x_t and outputs a value y_t. A loop allows information to be passed from one step of the network to the next. Bias weights are omitted for clarity. Adapted from Olah (2015).

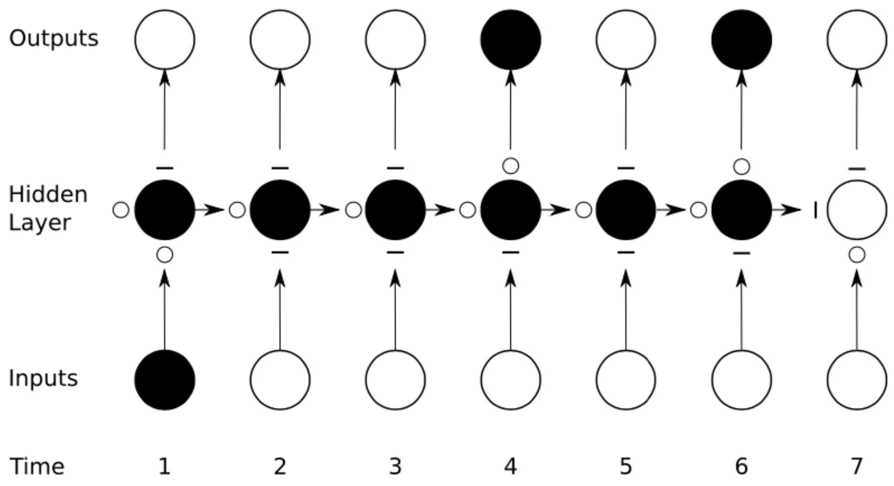


Figure A.8: As in Figure 3.9, the shading of the nodes indicates their sensitivity to the inputs at a particular point in time. In this case, the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. In this example, all gates are either entirely open ('O') or closed ('—'). From Graves (2012).

B

Code Repository and Additional Results

B.1 | Code Repository

A git repository with all code used in this dissertation is available at `https://gitfront.io/r/zerafachris/52df30fb666ba880749c8e951a3d056ce628a6cd/PhD/`. In the following chapter, any reference made to code will refer to this repository.

B.2 | Marmousi-2 Model

The original Marmousi-2 model has been made available by Martin et al. (2006) under a Creative Commons Attribution 4.0 International License. This was modified with a 150m median filter. Figure B.1 shows the impact of the 150m median filter on the vertical resolution of the model. The code used for this modification is available at `./code/appendix/Marmousi_2_generator.ipynb`.

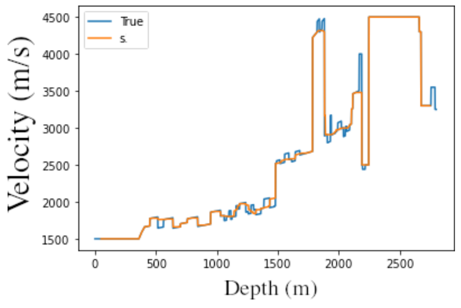


Figure B.1: Sample velocity through original and modified Marmousi-2 model. s is the 150m median filtered modified Marmousi-2.

B.3 | Classical FWI

B.3.1 | Inversion

FWI with Sobolev space norm regularization was used as the deterministic version of FWI within this work. The maximum frequency of the inversion process was set to be 3.5Hz. The iterative update process started from frequency 1Hz and iteratively updated by a factor of 1.2 until reaching a maximum frequency of 3.45Hz. The optimization algorithm was L-BFGS-B, with 50 iterations per frequency. Figure B.2 is the loss update for L-BFGS-B and Stochastic Gradient Descent. Figure B.3 shows the progression of the frequency updates. Code for this implementation is available at `./classical_FWI/marmousi`.

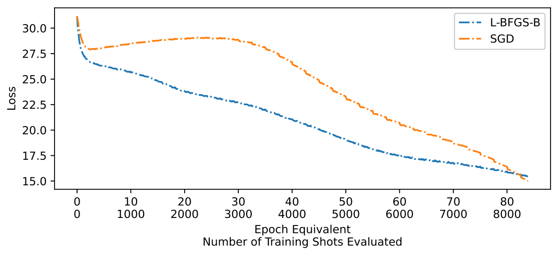


Figure B.2: Classical FWI loss update for L-BFGS-B and Stochastic Gradient Descent. L-BFGS-B was a better loss optimizer than Stochastic Gradient Descent due to the monotonically decreasing loss. Stochastic Gradient Descent training should have been stopped at an earlier epoch due to the increase at 30 when compared to earlier epoches.

B.3.2 | Ray-Tracing

Pre-cursor to FWI is ray-tracing modelling to assess areas of update from standard FWI formulation. Open source version of **fteikpy** Python library provided by Noble et al. (2014) was adapted and utilized on the Marmousi-2 in § B.2. This implementation computes accurate first arrival travel-times in 2D heterogeneous isotropic velocity models. The algorithm solves a hybrid Eikonal solver that combines a spherical approximation when close to the source and a plane wave approximation when far away. This reproduces properly the spherical behaviour of wave fronts in the vicinity of the source. Figure B.4 shows a sample of ray-paths for a source at 0km and depth 0km and ray coverage for the Marmousi model. The adapted code is available within `./ray_tracing/marmousi/marmousi_ray_tracing.ipynb`.

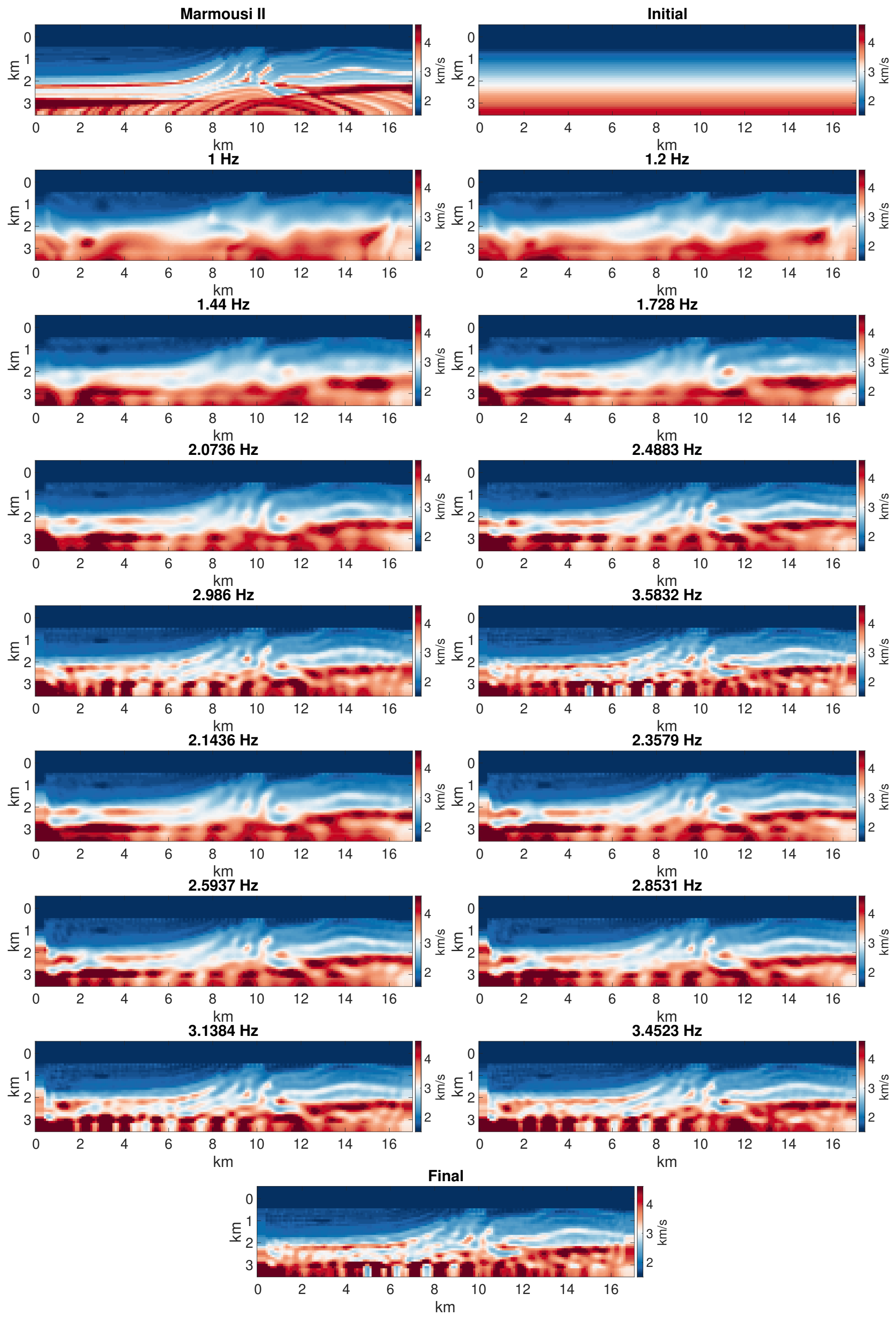
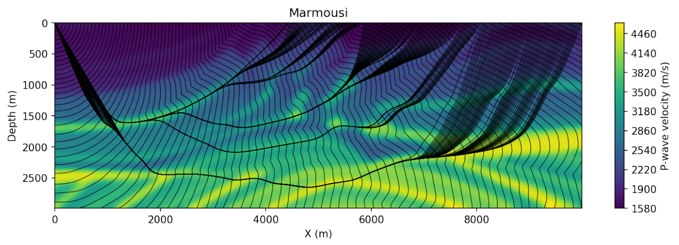
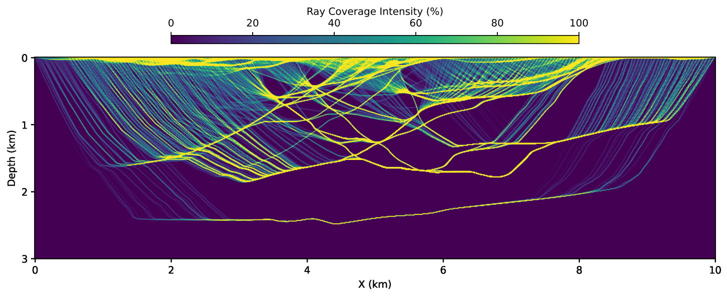


Figure B.3: Classical FWI frequency updates. Starting from 1Hz, model update frequency was increased by a factor of 1.2 until a maximum frequency of 3.45Hz.The optimization algorithm was L-BFGS-B, with 50 iterations per step.



(a) Sample of ray-paths through Marmousi



(b) Area of coverage intensity from ray-tracing.

Figure B.4: Ray-tracing using **fteikpy**.

B.4 | Data-Driven FWI

B.4.1 | DNN Architectures

Table B.1 lists DNN architectures used throughout Section 4.2.

Architecture	Code Repository
Time to Pseudo-Spectral 1D	`./appendix/DNN_arch_time_pseudo_1D.txt`
Time to Pseudo-Spectral 2D	`./appendix/DNN_arch_time_pseudo_2D.txt`
Conv1D	`./appendix/DNN_arch_conv1d.txt`
Conv2D	`./appendix/DNN_arch_conv2d.txt`
VGG	`./appendix/DNN_arch_vgg.txt`
ResNet	`./appendix/DNN_arch_resnet.txt`
Marmousi - Time to Pseudo-Spectral	`./appendix/DNN_arch_marm_time_pseudo.txt`
Marmousi - Pseudo-Spectral to Velocity	`./appendix/DNN_arch_marm_pseudo_velocity.txt`

Table B.1: Repositories defining different architectures used in Section 4.2.

B.4.2 | Architecture and Loss Tuning

Tables B.2-B.5 show results for different architecture and loss optimizer combinations.

Architecture	Loss Optimizer	Duration (Hours)	Rank
MLP	Adagrad	45	18
MLP	Adadelta	54	16
MLP	RMSprop	31	20
MLP	Adam	36	19
Conv1D	Adagrad	49	17
Conv1D	Adadelta	59	14
Conv1D	RMSprop	59	14
Conv1D	Adam	55	15
Conv2D	Adagrad	66	12
Conv2D	Adadelta	66	12
Conv2D	RMSprop	67	10
Conv2D	Adam	67	10
VGG	Adagrad	75	8
VGG	Adadelta	75	8
VGG	RMSprop	177	1
VGG	Adam	75	8
ResNet	Adagrad	108	4
ResNet	Adadelta	108	4
ResNet	RMSprop	144	2
ResNet	Adam	107	5

Table B.2: Architecture and Loss comparison - Duration. The shortest duration is better result.

B.4.3 | Network Training Process

For each of the Architecture-Loss combinations available, these networks were trained on an Intel i7-7800x X-series CPU workstation provided by the Department of Physics at the University of Malta. The script for this is available at `./code/appendix/ArchitectureComparison.py`. The initial maximum epoch was set to 20, with 5 epoch early stopping and 2 epoch learning rate reduction of 0.2 when reaching a plateau. The number of traces in the training and testing epoch generators were 1,000,000 and 100,000 respectively. The batch size was set to 100 for MLP and Conv1D, whilst 20 for the other networks due to ram size of the workstation.

Architecture	Loss Optimizer	Train MSE	Rank
MLP	Adagrad	6352.22549	4
MLP	Adadelta	86460.7877	13
MLP	RMSprop	8098.47514	7
MLP	Adam	1369163.37	19
Conv1D	Adagrad	5180.91202	2
Conv1D	Adadelta	9913.78826	8
Conv1D	RMSprop	14578.6939	9
Conv1D	Adam	20305.0825	10
Conv2D	Adagrad	6808.32294	5
Conv2D	Adadelta	111152.159	15
Conv2D	RMSprop	1618.25641	1
Conv2D	Adam	5145821.2	20
VGG	Adagrad	6139.20768	3
VGG	Adadelta	93423.5512	14
VGG	RMSprop	59200.3044	11
VGG	Adam	78038.1544	12
ResNet	Adagrad	7280.4593	6
ResNet	Adadelta	131353.653	17
ResNet	RMSprop	123707.716	16
ResNet	Adam	232489.037	18

Table B.3: Architecture and Loss comparison - Training MSE. The lowest MSE is the better result.

Architecture	Loss Optimizer	Rank
MLP	Adagrad	20
MLP	Adadelta	19
MLP	RMSprop	9
MLP	Adam	9
Conv1D	Adagrad	15
Conv1D	Adadelta	19
Conv1D	RMSprop	9
Conv1D	Adam	17
Conv2D	Adagrad	15
Conv2D	Adadelta	15
Conv2D	RMSprop	9
Conv2D	Adam	3
VGG	Adagrad	15
VGG	Adadelta	15
VGG	RMSprop	9
VGG	Adam	4
ResNet	Adagrad	17
ResNet	Adadelta	15
ResNet	RMSprop	3
ResNet	Adam	3

Table B.4: Architecture and Loss comparison - Qualitative assessment of under-fitting/over-fitting and learning rate performance.

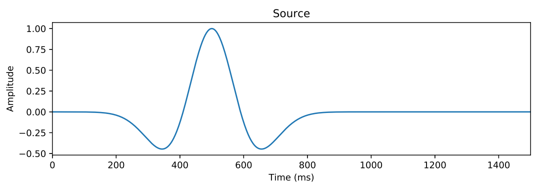
Architecture	Loss Optimizer	Inversion RMSE (Hours)	Rank
MLP	Adagrad	102.008808	11
MLP	Adadelta	144.749442	8
MLP	RMSprop	41.112304	15
MLP	Adam	161.546795	7
Conv1D	Adagrad	19.0347424	18
Conv1D	Adadelta	15.4491243	19
Conv1D	RMSprop	15.4070707	20
Conv1D	Adam	20.1883355	17
Conv2D	Adagrad	84.3773592	12
Conv2D	Adadelta	252.899762	4
Conv2D	RMSprop	45.4919989	14
Conv2D	Adam	117.230985	10
VGG	Adagrad	61.7665919	13
VGG	Adadelta	135.254508	9
VGG	RMSprop	23.2408697	16
VGG	Adam	220.758795	5
ResNet	Adagrad	358.686894	2
ResNet	Adadelta	264.61373	3
ResNet	RMSprop	162.649915	6
ResNet	Adam	406.824565	1

Table B.5: Architecture and Loss comparison - Inversion RMSE. RMSE for 100,000 validation velocity profile are compared to the true velocity.

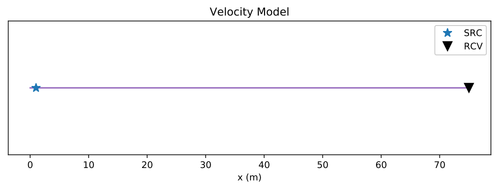
B.5 | Theory-Guided FWI

B.5.1 | 1D Results

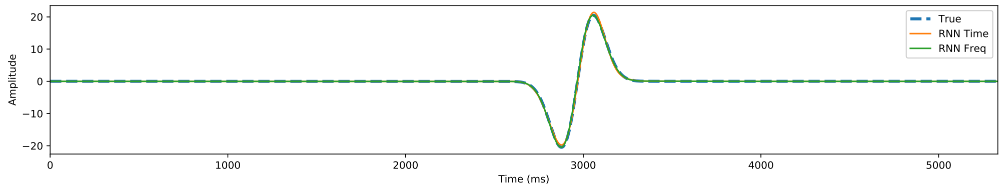
Replacing the forward modelling component with RNN directly implies that the RNN should be able to retrieve all wavefield components. This was first tested by considering a 1D direct wave as shown in Figure B.5. A 10Hz Ricker wavelet (Figure B.5(a)) was injected into a 1D 1500 ms^{-1} constant velocity model (Figure B.5(b)) with a single source and single receiver. The wave was forward propagated for 5333 time-steps at 1ms, with a 10m grid spacing. The resulting direct wave is illustrated in Figure B.5(c), with True being the analytical solution calculated using a 1D Green's function, RNN Time and RNN Freq are the RNN implementation for forward modelling using Time and Fourier spatial derivatives respectively. Qualitatively, there is no visible difference between either approach. Quantitative analysis shown in Table B.6 indicates that the pseudo-spectral approach is producing slightly better results with an improved error tolerance of 0.2 in amplitude, resulting in a 0.9% improvement in the RPE.



(a) Source for 1D experiments.



(b) 1D constant 1500 m/s velocity model.



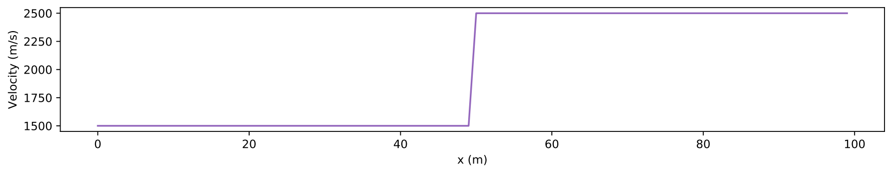
(c) 1D direct wave trace forward modelling comparison to analytical 1D Green's function wavefield, RNN Time and RNN Freq showing no discrepancies.

Figure B.5: 1D Direct wave forward modelling comparison.

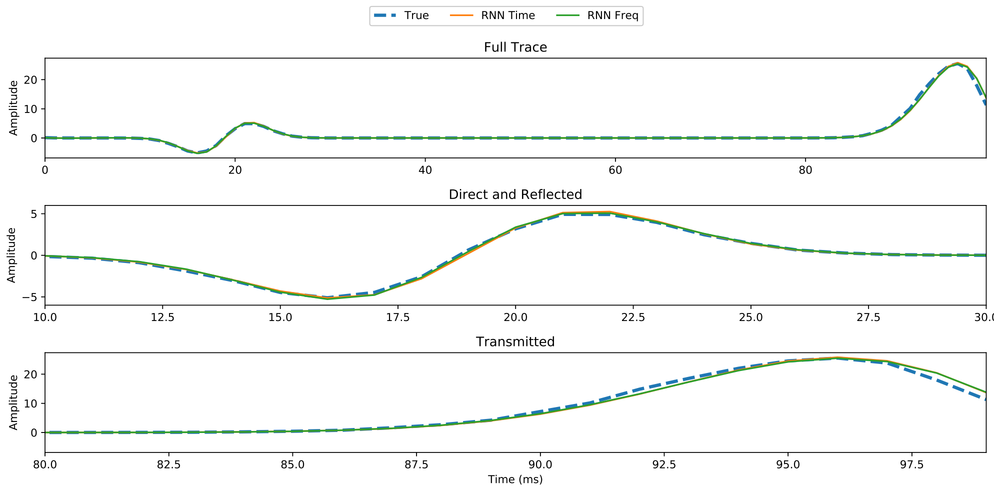
Modelling	Error Tolerance	RPE (%)
RNN Time	0.980	4.786
RNN Freq	0.780	3.809

Table B.6: Empirical comparison of 1D Direct wave modelling to 1D Green's function.

An identical source was used to test for Reflected and Transmitted arrivals. Figure B.6(b) is the 1D step velocity model ranging from 1500 ms^{-1} to 2500 ms^{-1} used to produce the forward propagated wavefields in Figure B.6(b). The top shows the full wavefield, with Direct and Reflected arrivals at about 20 ms and Transmitted wave with peak at 96 ms. Middle and bottom sections of Figure B.6(b) show zoomed sections of the wavefield respectively. Either the Reflected and Transmitted component match near identically to the Green's function formulation. Quantitatively, the RNN Time implementation was found to be slightly improved, with an improvement of 0.1 in error tolerance and 0.2% RPE (Table B.7).



(a) 1D step velocity model ranging from 1500 ms^{-1} to 2500 ms^{-1}.



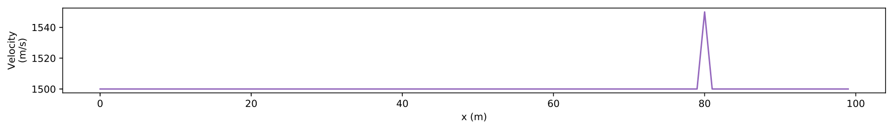
(b) Top: Full wavefield showing Direct and Reflected waves at about 20ms and Transmitted wave with peak at 96 ms. Middle and Bottom: Zoomed in sections from Full trace showing different components of the wavefield and near perfect reconstruction.

Figure B.6: 1D direct, reflected and transmitted wave forward modelling comparison.

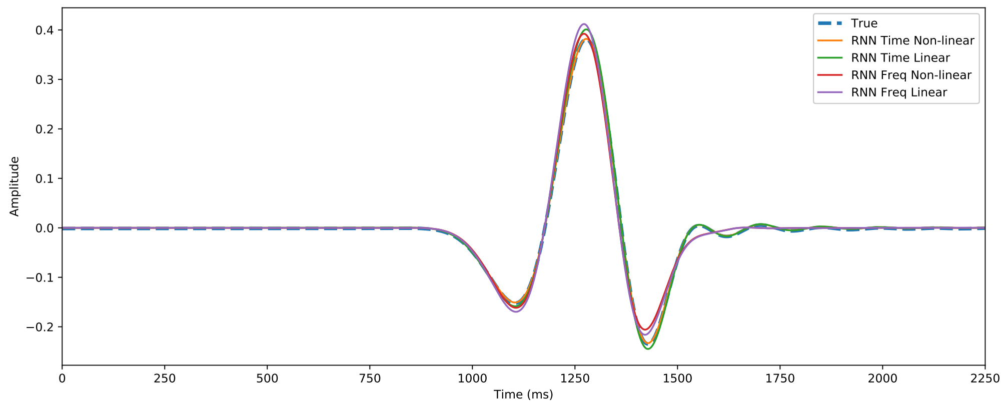
Modelling	Error Tolerance	RPE (%)
RNN Time	2.460	9.637
RNN Freq	2.520	9.872

Table B.7: Empirical comparison of 1D direct, reflected and transmitted wave modelling.

The remaining wavefield component to be modelled are Scattering waves. A constant velocity model with one-point scatterer was created with velocity ranging from 1500 ms^{-1} to 1550 ms^{-1} at the point scatterer (Figure B.7(a)). The same source in the previous two wavefields was used. RNN implementations were modelled for the waveform at a point to be depended non-linearly on the scattering amplitude and then approximately linearised. The resulting wavefields are given in Figure B.7(b). The Direct wave was not included in the Scattered wavefield reconstruction. From the error tolerance and RPE comparison (Table B.8), RNN Time produces marginally better results than RNN Freq. This is due to the lack of ringing effect introduced due to discretisation beyond 1500ms which gets absorbed within the complex component of the pseudo-spectral approach. The RNN Frequency approach is thus a superior modelling approach in 1D. In the 2D case, RNN Time does not suffer from this effect.



(a) 1D scattering velocity model ranging from 1500 ms^{-1} to 1550 ms^{-1}.



(b) Scattering wavefield modelling. Direct wavefield was excluded in the modelling.

Figure B.7: 1D scattering wave forward modelling comparison.

Modelling	Error Tolerance	RPE (%)
RNN Time – Non-linear	0.009	0.031
RNN Time – Linear	0.010	2.493
RNN Freq – Non-linear	0.030	7.649
RNN Freq – Linear	0.040	9.711

Table B.8: Empirical comparison of 1D scattering wave modelling.

B.5.2 | RNN Hyper-Parameter Tuning

Similarly to the approach shown in Sun et al. (2019), a benchmark 1D 4-layer synthetic profile, with velocities [2, 3, 4, 5] kms^{-1}, was used to identify the ideal parameters for the RNN architecture. Classical 1D second-order Finite Difference (FD) modelling was used to generate the required true receiver data. Batch size is used as a discriminator throughout Figure B.9. The results indicate that the larger the batch size used, the better the inversion as more data is being used. However, given fore-sight that this hyper-parameter tuning will be used a large dataset that might not fit in Graphical Processing Unit RAM, this was fixed at batch size one.

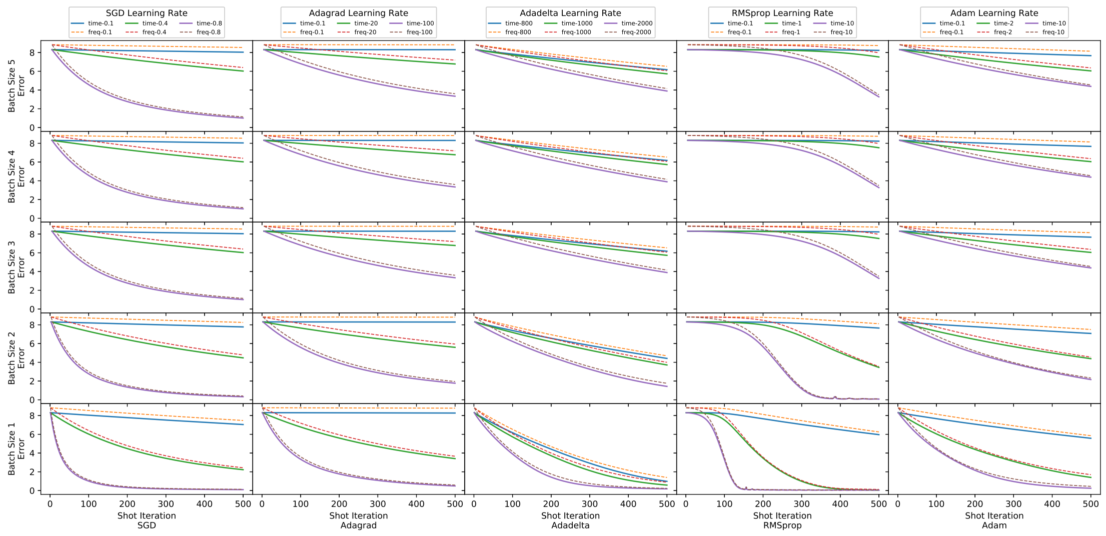
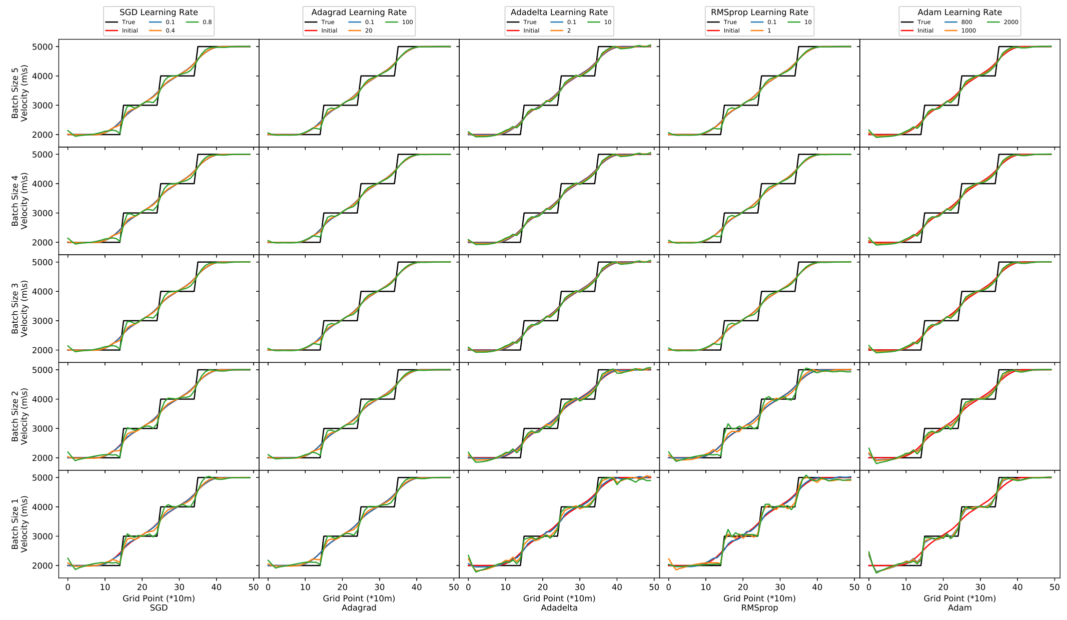
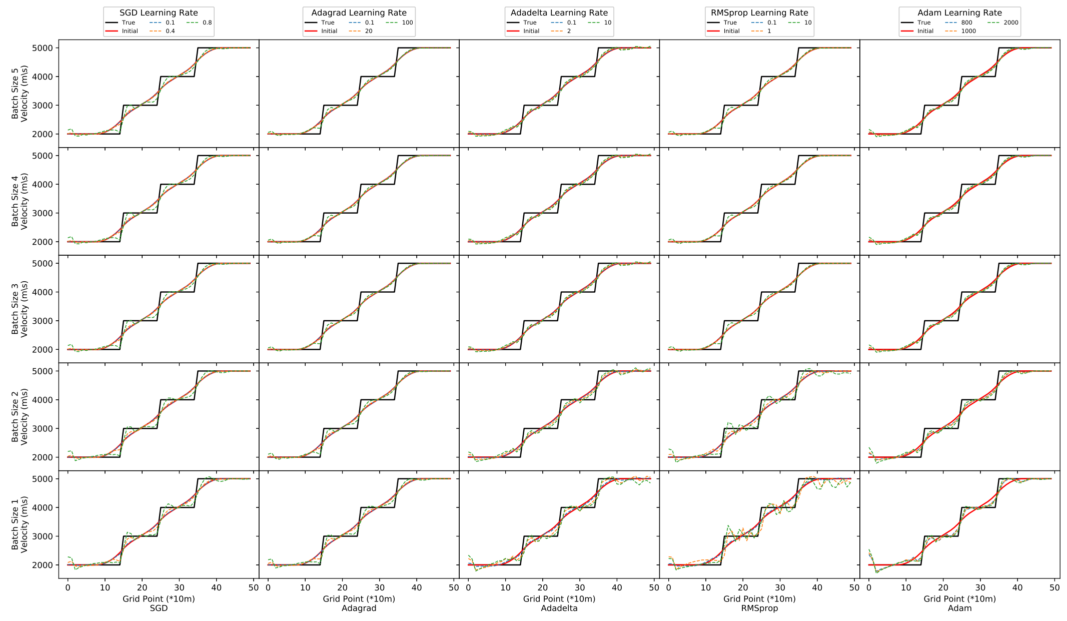


Figure B.8: Losses for different loss optimizer learning rate hyper-parameter tuning.



(a) RNN Time



(b) RNN Freq

Figure B.9: Loss optimizer learning rate hyper-parameter tuning results.

B.5.3 | RNN Inversion Update Progress

Complementary to inverted Marmousi models in § 4.3.5.3, Figure B.10 gives the update progress at epoch 10, 25, 40, 55, 70, 85 and 100 for RNN Time and RNN Freq, together with residual. Furthermore, classical FWI progress is included at different update frequency scales. In addition, receivers are provided in Figure B.11.

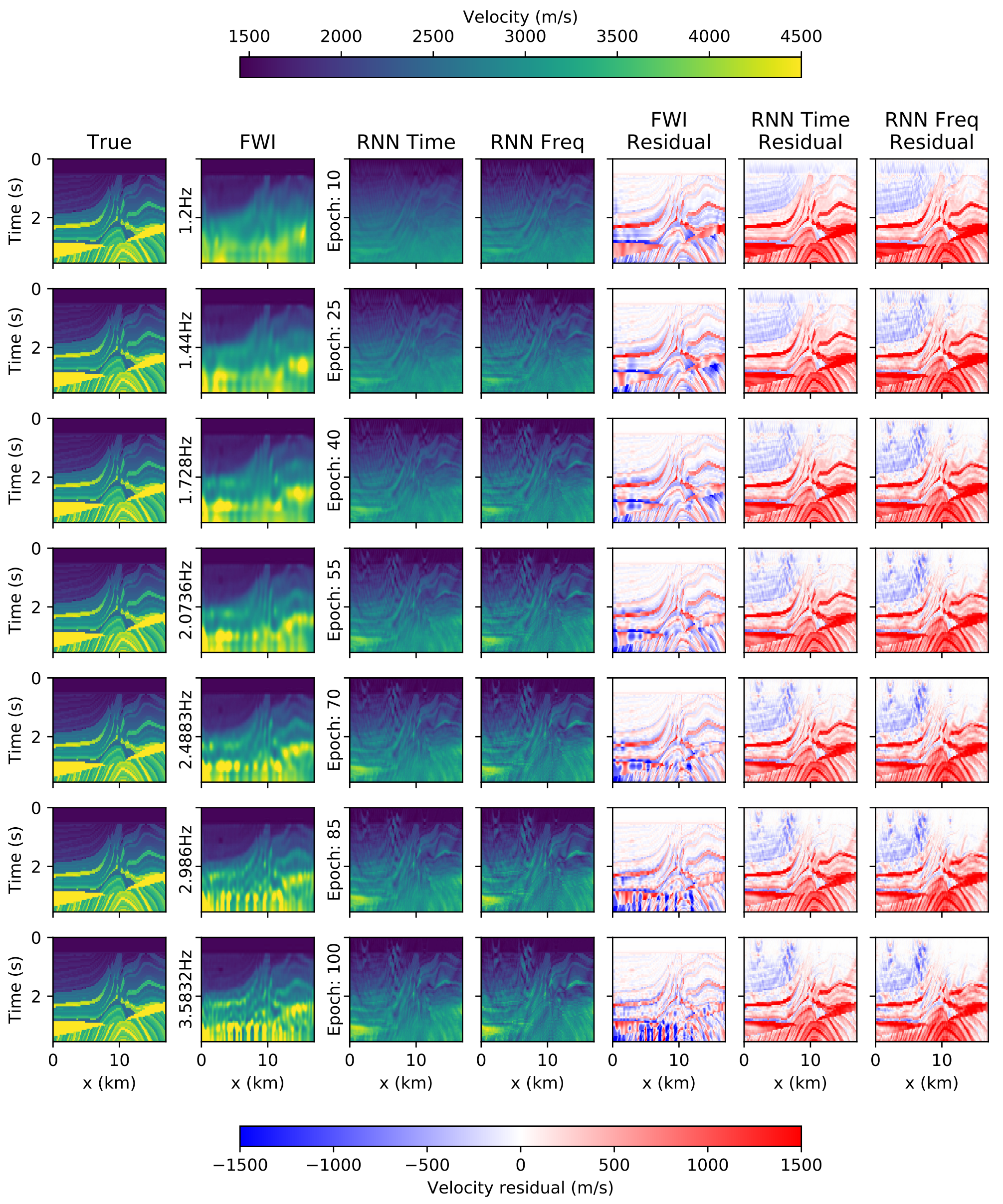


Figure B.10: Velocity model inversion update progress for classical FWI, RNN Time and Freq, with residuals.

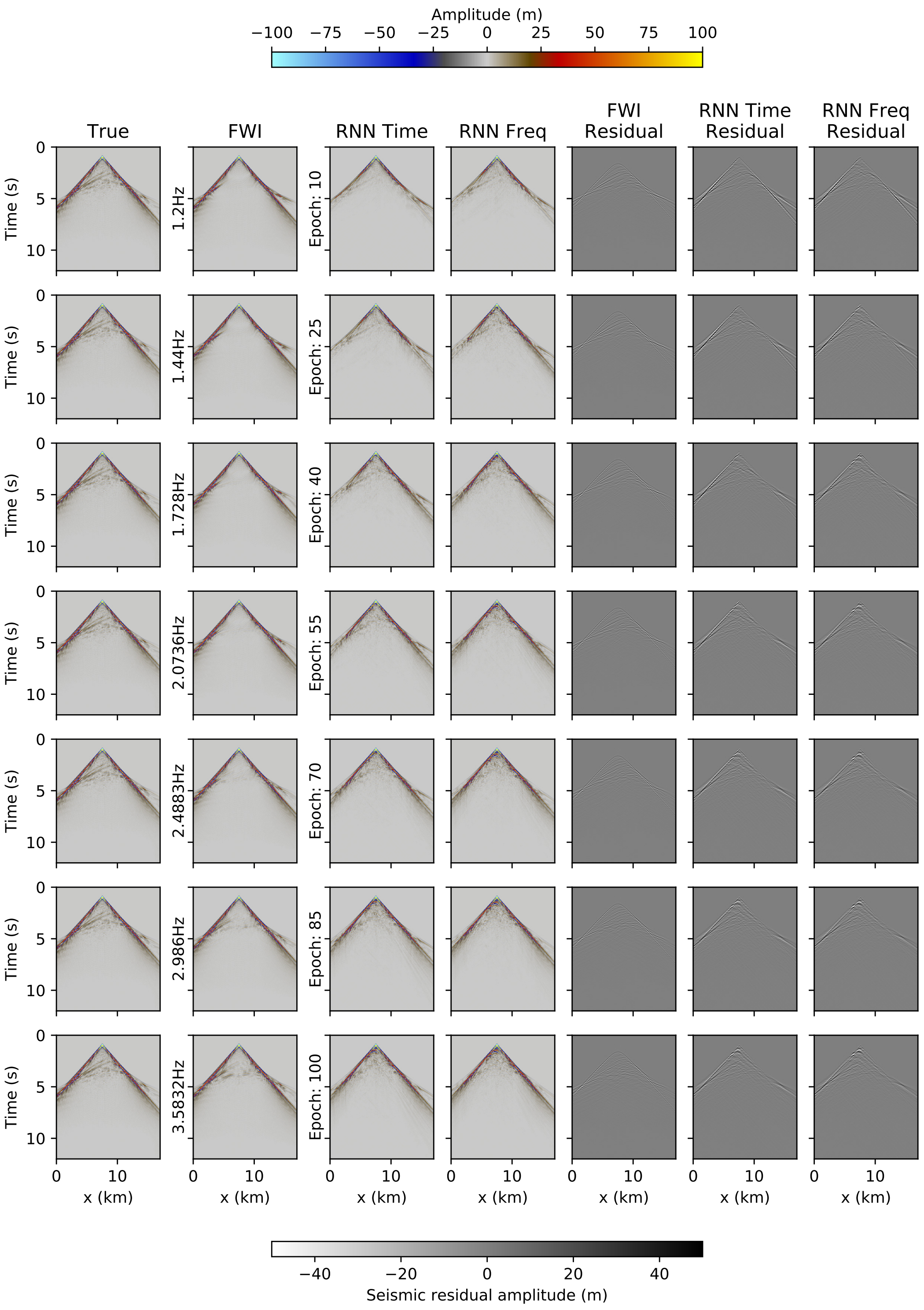


Figure B.11: Receiver progress through model updates for classical FWI, RNN Time and Freq, with residuals.

C

Publications and Collaborations

C.1 | Publications

Zerafa, C., Galea, P. & Sebu, C. (2019). "Learning to Invert Pseudo-Spectral Data for Seismic Waveforms". *Xjenza Online* **7**(1),3-17.

Zerafa, C. (2018) "DNN application of pseudo-spectral FWI". *First EAGE/PESGB Workshop on Machine Learning: European Edition.*

TBD *Parts of this work are planned for publishing in SEG Geophysics and OUP Geophysical Journal International.*

C.2 | Conferences & Poster Presentations

- **Jul 2020** Zerafa, C. "Overview of Machine Learning Applications in Geophysics and Seismology". *Department of Geosciences Summer Seminar, University of Malta.*
- **Nov 2019**. Zerafa, C., Galea, P. & Sebu, C. "Learning to Invert Pseudo-Spectral Data for Seismic Waveforms". *OptML: Optimization and Machine Learning, University of Southampton, UK.*
- **Mar 2018**. Zerafa, C. "An Introduction to High Resolution Seismic Imaging". *Scubed Annual Scientific Conference.*

C.3 | Appointments

- **Jan 2022** Post-Doc at Istituto Nazionale di Geofisica e Vulcanologia, Pisa.
  Will be joining Istituto Nazionale di Geofisica e Vulcanologia for 2-year Post-Doc at Istituto Nazionale di Geofisica e Vulcanologia for project titled *SOME Seismological Oriented Machine lEarning*. I will be involved in two main working groups:

  – *WP2*: Earthquake detection and characterization using large seismic datasets from tectonic/volcanic processes and hydrocarbon/geothermal exploitation. In this WP, we plan to extensively test published or newly developed supervised and unsupervised Machine Learning algorithms on the wealth of already available seismic datasets. We will use seismic datasets from regional (e.g., INSTANCE, AlpArray, Amatrice/Norcia) and local scale (Mugello, Irpinia fault system, Val d'Agri, Amiata, Larderello, Campi Flegrei, Etna).

  – *WP4*: Automatic extraction of phase and group velocity surface dispersion curves. We plan to build up on the recent work by Zhang et al. (2020), applying the method to a various range of datasets already available at Istituto Nazionale di Geofisica e Vulcanologia (Molinari et al., 2020, 2015) or available in the near future as results of Department Projects at local and regional scales and in a wide frequency range.

- **Apr 2020** - CA17137 WG2 - Co-Leader - `www.g2net.eu/wgs/wg2`.
  I joined a leadership position within the working group together with Dr. Ilec. My involvement includes overseeing and organising the research initiatives within the group, introduced a common code repository to encourage collaboration and hold weekly meetings to steer the working group forward.

- **Sep 2018** - CA17137 University of Malta Representative - `https://www.g2net.eu/`.
  Active member within CA17137 - g2net - A network for Gravitational Waves, Geophysics and Machine Learning, with particular focus to Working Group 2 - Machine Learning for low-frequency seismic measurement. Research deals with acquisition, processing and interpretation of seismic data, with the goal of combating the seismic influences at Gravitational Wave detector site, using multi-disciplinary research focussing on advanced techniques available from state of the art machine learning algorithms.

C.4 | Collaborations

- **Apr 2020** Short Term Scientific Mission at Istituto Nazionale di Geofisica e Vulcanologia, Pisa.
  In collaboration with Giunchi, C., De Matteo, G., Gaviano, S., developed two CNN approaches able to classify local earthquakes into 13 classes with different epicentre and magnitude characterizations.

C.5 | Organisation of Events and Guest Lecturing

- **Sep 2020** Kaggle Competition - Classification of Gravitational Wave Glitches. Cuoco, E., Zerafa, C., Messenger, C., Williams, M.
  Collaborating with Kaggle and other g2net members to host an international competition using gravitational wave data which could lead to novel was of automatic detection of gravitational wave detection. In turn, this could directly influence the next generation of gravitational wave research and better understanding of the known and unknown universe.

- **Mar 2020** CA17137 – g2net – 2nd Training School.
  Hosted an international training school for CA17137 within the University of Malta. Was mainly responsible within the Scientific Organizing Committee, Local Organizing Committee and one of the lecturers. Classes though included a course in Machine Learning and hosted a hackathon for all participants. Testimonials, lecture notes, video recordings and code can be found at `https://github.com/zerafachris/g2net_2nd_training_school_malta_mar_2020`.

- **Sep 2019** PyMalta: First Steps towards Machine Learning
  Hosted a two-part training introductory series on Machine Learning using Python. All code can be found in the repo `https://github.com/PyMalta/Introduction_to_ML_CZ`.

C.6 | Relevant Training & Certification

- **Nov 2019** "Oberwolfach Graduate Seminar: Mathematics of Deep Learning", *MFO, Oberwalfach, Germany*. Tutorial based course focussing on state-of-the-art mathematical analysis of deep learning algorithms. Training focussed on (i) approximation theory, (ii) expressivity, (iii) generalization, and (iv) interpretability. See `MFO`.

- **Mar 2019** "ML and Statistical Analysis", Distinction, *The Data Incubator*. Tutorial based course on the use of ML on real world data set, with a heavy emphasis on creative use of different data science techniques to solve problems from multiple perspectives. Projects included NLP, clustering, Time series analysis and anomaly detection. See `DataIncubator`.
- **Sep 2018** "Full Waveform Inversion: Maths and Geophysics", *KIT, Karlsruhe, Germany*. Tutorials for high performance computing with specific focus on computational mathematics. See `KIT2018`.
- **Jul 2018** "International Summer School on Deep Learning", *Gdansk University of Technology, Poland*. In-depth training and mini-projects on a range of learning methods, with particular focus on DNNs. See `ISSDL`.
- **Jun 2018** "Scientific Computing and Python for Data Science", Distinction, *The Data Incubator*. Tutorial based course covering linear regression and gradient descent techniques. Particular emphasis included cost function analysis, dimensionality reduction, regularization and feature engineering. See `DataIncubator`.
- **Apr 2018** "Graphical Processing Unit-based analytics and data science, *Vicomtech Research Center, Spain*. Intensive training on use of GPUs using PyCuda for big data analytics as applied to machine learning for image processing, segmentation, de-noising, filtering, interpolation and reconstruction. See `BigSkyEarth`.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *Software available from tensorflow.org*.

Adler, J. and Öktem, O. (2017). Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):1–24.

Adler, J., Ringh, A., Öktem, O., and Karlsson, J. (2017). Learning to solve inverse problems using Wasserstein loss. *Iclr 2018*, pages 1–13.

Ali, H. B. H., Operto, S., Virieux, J., and Sourbier, F. (2007). 3D acoustic frequency-domain full-waveform inversion. In *SEG Technical Program Expanded Abstracts 2007*, pages 1730–1734. Society of Exploration Geophysicists.

Anderson, G. J. and Lucas, D. D. (2018). Machine Learning Predictions of a Multiresolution Climate Model Ensemble. *Geophysical Research Letters*, 45(9):4273–4280.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems 29 (NIPS 2016)*.

Araya-Polo, M., Jennings, J., Adler, A., and Dahlke, T. (2018). Deep-learning tomography. *The Leading Edge*, 37(1):58–66.

Arridge, S., Maass, P., Öktem, O., and Schönlieb, C.-B. (2019). Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174.

Asnaashari, A., Brossier, R., Garambois, S., Audebert, F. F., Thore, P., and Virieux, J. (2013). Regularized seismic full waveform inversion with prior model information. *Geophysics*, 78(2):R25–R36.

Ben-Hadj-Ali, H., Operto, S., and Virieux, J. (2008). Velocity model building by 3D frequency-domain, full-waveform inversion of wide-aperture seismic data. *Geophysics*, 73(5):VE101–VE117.

Ben Hadj Ali, H., Operto, S., Virieux, J., and Sourbier, F. (2007). 3D acoustic frequency-domain full-waveform inversion. *Extended Abstracts*, pages 1730–1734.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Bergen, K. J., Johnson, P. A., Maarten, V., and Beroza, G. C. (2019). Machine learning for data-driven discovery in solid Earth geoscience. *Science*, 363(6433).

Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. (2018). Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Biswas, R. and Sen, M. K. (2017). 2D Full Waveform Inversion and Uncertainty Estimation using the Reversible Jump Hamiltonian Monte Carlo. *SEG Technical Program Expanded Abstracts 2017*, pages 1280–1285.

Biswas, R., Sen, M. K., Das, V., and Mukerji, T. (2019). Pre-stack inversion using a physics-guided convolutional neural network. In *SEG Technical Program Expanded Abstracts 2019*, pages 4967–4971. Society of Exploration Geophysicists.

Blanch, J. O., Robertsson, J. O. A., and Symes, W. W. (1995). Modeling of a constant Q: Methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique. *Geophysics*, 60(1):176.

Bleibinhaus, F. and Rondenay, S. (2009). Effects of surface scattering in full-waveform inversion. *GEOPHYSICS*, 74(6):WCC69–WCC77.

Boehm, C., Fichtner, A., de la Puente, J., and Hanzich, M. (2015). Lossy Wavefield Compression for Full-Waveform Inversion. In *AGU Fall Meeting Abstracts*, volume 2015, pages S23C–2714.

Bogaerts, T., Masegosa, A. D., Angarita-Zapata, J. S., Onieva, E., and Hellinckx, P. (2020). A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data. *Transportation Research Part C: Emerging Technologies*, 112:62–77.

Born, M. and Wolf, E. (1980). Principles of optics. *Pergamon Press*, 6:188–189.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

Boyd, S., Parikh, N., and Chu, E. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc.

Bruna, J., Sprechmann, P., and LeCun, Y. (2015). Super-Resolution with Deep Convolutional Sufficient Statistics.

Bryson, A. E. (1961). A gradient method for optimizing multi-stage allocation processes. In *Proc. Harvard Univ. Symposium on digital computers and their applications*, volume 72.

Bubba, T. A., Kutyniok, G., Lassas, M., Maerz, M., Samek, W., Siltanen, S., and Srinivasan, V. (2019). Learning the invisible: a hybrid deep learning-shearlet framework for limited angle computed tomography. *Inverse Problems*, 35(6):64002.

Bunks, C., Saleck, F. M., Zaleski, S., and Chavent, G. (1995). Multiscale seismic waveform inversion. *Geophysics*, 60(5):1457–1473.

Cai, X.-H., Liu, Y., Ren, Z.-M., Wang, J.-M., Chen, Z.-D., Chen, K.-Y., and Wang, C. (2015). Three-dimensional acoustic wave equation modeling based on the optimal finite-difference scheme. *Applied Geophysics*, 12(3):409–420.

Calde On-Macías, C., Sen, M. K., and Stoffa, P. L. (1997). Hopfield neural networks, and mean field annealing for seismic deconvolution and multiple attenuation. *GEOPHYSICS*, 62(3):992–1002.

Caruana, R., Lawrence, S., and Giles, L. (2000). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *the 13th International Conference on Neural Information Processing Systems*, pages 402–408.

Chang, J. H., Li, C.-L., Póczos, B., Kumar, B. V. K., and Sankaranarayanan, A. C. (2017). One Network to Solve Them All—Solving Linear Inverse Problems using Deep Projection Models. *IEEE International Conference on Computer Vision (ICCV)*.

Chentouf, R. (1997). Construction de reseaux de neurones multicouches pour l'approximation. *Ph.D. thesis, Institut National Polytechnique, Grenoble*.

Chollet, F. (2015). Keras. *https://github.com/fchollet/keras*.

Ciresan, D. C., Meier, U., and Masci, J. (2011). A Committee of Neural Networks for Traffic Sign Classification. *International Joint Conference on Neural Networks (IJCNN-2011, San Francisco)*, 1(1).

Ciresan, D. C., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-Column Deep Neural Network for Traffic Sign Classification. *Neural Networks*, pages 333–338.

Claerbout, J. F. (1971). Toward a unified theory of reflector mapping. *Geophysics*, 36(3):467–481.

Claerbout, J. F. (1976). *Fundamentals of geophysical data processing: McGraw-Hili Book Co.* Inc.

Cochrane, G. and Cooper, A. K. (1991). Sonobuoy seismic studies at ODP drill sites in Prydz Bay, Antarctica. pages 27–43.

Dadvand, P., Lopez, R., and Onate, E. (2006). Artificial Neural Networks for the Solution of Inverse Problems A Variational Formulation for the Multilayer Perceptron. *Proceedings of the International Conference on Design Optimisation Methods and Applications ERCOFTAC*, 2006:1–10.

Dai, H. and MacBeth, C. (1994). Split shear-wave analysis using an artificial neural network. *First Break*, 12(12):605–613.

De los Reyes, J. C., Schönlieb, C.-B., and Valkonen, T. (2017). Bilevel parameter learning for higher-order total variation regularisation models. *Journal of Mathematical Imaging and Vision*, 57(1):1–25.

Deng, L. and Yu, D. (2013). Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*.

Dokmanić, I., Bruna, J., Mallat, S., and de Hoop, M. (2016). Inverse Problems with Invariant Multiscale Statistics. *CoRR*.

Dolenko, S., Efitorov, A., Burikov, S., Dolenko, T., Laptinskiy, K., and Persiantsev, I. (2015). Neural Network Approaches to Solution of the Inverse Problem of Identification and Determination of the Ionic Composition of Multi-component Water Solutions. *CCIS*, 517:109–118.

Dowla, F. U. and Rogers, L. L. (1996). *Solving Problems in Environmental Engineering and Geosciences with Artificial Neural Networks*. MIT Press.

Downton, J. E. and Hampson, D. P. (2018). Deep neural networks to predict reservoir properties from seismic. *Presented at 6th GeoConvention 2018*.

Dreyfus, S. (1973). The computational solution of optimal control problems with time lag. *IEEE Transactions on Automatic Control*, 18(4):383–385.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Elshafiey, I. M. (1991). Neural network approach for solving inverse problems.

Fahlman, S. E. (1991). The recurrent cascade-correlation architecture. In *Advances in neural information processing systems*, pages 190–196.

Falsaperla, S., Graziani, S., Nunnari, G., and Spampinato, S. (1996). Automatic classification of volcanic earthquakes by using multi-layered neural networks. *Natural Hazards*, 13(3):205–228.

Fletcher, R. (1987). Practical Methods of Optimization, Second Edition. pages 3–6.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

Galliani, S., Lanaras, C., Marmanis, D., Baltsavias, E., and Schindler, K. (2017). Learned Spectral Super-Resolution.

Gardner, G. H. F., Gardner, L. W., and Gregory, A. R. (1974). Formation Velocity and Density - The diagnostic basics for Stratigraphic Traps. *GEOPHYSICS*, 39(6):770–780.

Gauthier, O., Virieux, J., and Tarantola, A. (1986). Two-dimensional nonlinear inversion of seismic waveforms: Numerical results. *Geophysics*, 51(7):1387–1403.

Gazdag, J. (1981). Modeling of the acoustic wave equation with transform methods. *Geophysics*, 46(6):854.

Gerdova, I. V., Churina, I. V., Dolenko, S. A., Dolenko, T. A., Fadeev, V. V., and Persiantsev, I. G. (2002). New opportunities in solution of inverse problems in laser spectroscopy due to application of artificial neural networks. *SPIE Proceedings*, 4749(8):157–166.

Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

Gerstoft, P. (1994). Inversion of seismoacoustic data using genetic algorithms and a posteriori probability distributions. *The Journal of the Acoustical Society of America*, 95(2):770–782.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge MA.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. *CoRR*.

Graves, A. (2012). Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer.

Grohs, P., Perekrestenko, D., Elbrächter, D., and Bölcskei, H. (2019). Deep Neural Network Approximation Theory.

Hadamard, J. (1907). Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées. *Mem. Sav. Etrang.*, 33:515–641.

Hallström, E. (2016). Backpropagation from the beginning – Erik Hallström. *Medium*.

Halpert, A. D. (2018). Deep learning-enabled seismic image enhancement. In *SEG Technical Program Expanded Abstracts 2018*, pages 2081–2085. Society of Exploration Geophysicists.

Hammer, B. (2000). On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1-4):107–123.

Hamshaw, S. D., Dewoolkar, M. M., Schroth, A. W., Wemple, B. C., and Rizzo, D. M. (2018). A New Machine-Learning Approach for Classifying Hysteresis in Suspended-Sediment Discharge Relationships Using High-Frequency Monitoring Data. *Water Resources Research*, 54(6):4040–4058.

Haykin, S. S. (2009). *Neural Networks and Learning Machines*. Number 3rd Edition. PEARSON, Prentice Hall.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. *Proceedings of the IEEE*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Hebb, D. O. (1949). The organization of behavior: A neuropsychological theory. *Brain Theory*.

Hecht-Nielsen, R. (1989). Theory of the Backpropagation Neural Network. *Proceedings Of The International Joint Conference On Neural Networks*, 1:593–605.

Hecht-Nielsen, R. (1990). *Neurocomputing*. Addison-Wesley Pub. Co.

Hicks, G. J. and Pratt, R. G. (2001). Reflection waveform inversion using local descent methods: Estimating attenuation and velocity over a gas-sand deposit. *Geophysics*, 66(2):598–612.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., and Sainath, T. N. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

Hochreiter, J. (1991). Untersuchungen zu dynamischen neuronalen Netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen*, pages 1–71.

Hochreiter, S. and Schmidhuber, J. (1996). *Bridging long time lags by weight guessing and "Long Short-Term Memory"*. IOP Press.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, 9:1735–1780.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximator. *Neural Networks*, 2:359–366.

Hu, J., Shen, L., and Sun, G. (2017). Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 7.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106.

Hughes, T. W., Williamson, I. A. D., Minkov, M., and Fan, S. (2019). Wave physics as an analog recurrent neural network. *Science advances*, 5(12):eaay6946.

Igel, H. (2017). *Computational seismology: a practical introduction*. Oxford University Press.

Innanen, K. (2014). Quantifying the incompleteness of the physics model in seismic inversion. *CREWES Research Report*, 26.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2017-Octob.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, pages 1097–1105.

Jones, I. F. and Davison, I. (2014). Seismic imaging in and around salt bodies. *Interpretation*, 2(4):SL1–SL20.

Jozinović, D., Lomax, A., Štajduhar, I., and Michelini, A. (2020). Rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network. *Geophysical Journal International*, 222(2):1379–1389.

Kalita, M. and Alkhalifah, T. (2017). Efficient full waveform inversion using the excitation representation of the source wavefield. *Geophysical Journal International*, 210(3):1581–1594.

Kamphorst, J.-P. E. S. O. and Ruelle, D. (1987). Recurrence Plots of Dynamical Systems. *Europhysics Letters*, 4(9):17.

Kazei, V. and Ovcharenko, O. (2019). Simple frequency domain full-waveform inversion (FWI) regularized by Sobolev space norm.

Keivan Ekbatani, H., Pujol, O., and Segui, S. (2017). Synthetic Data Generation for Deep Learning in Counting Pedestrians. In *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*, volume 2017-Janua, pages 318–323. SCITEPRESS - Science and Technology Publications.

Kelley, H. J. (1960). Gradient Theory of Optimal Flight Paths. *ARS Journal*, 30(10):947–954.

Kelly, B., Matthews, T. P., and Anastasio, M. A. (2017). Deep Learning-Guided Image Reconstruction from Incomplete Data. *arXiv preprint arXiv:1709.00584*.

Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, pages 1–62.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Köhler, A., Ohrnberger, M., and Scherbaum, F. (2010). Unsupervised pattern recognition in continuous seismic wavefield records using Self-Organizing Maps. *Geophysical Journal International*, 182(3):1619–1630.

Kolmogoro, A. N. and Tikhomirov, V. M. (1956). On the representation of continuous functions of several variables as superpositions of functions of smaller number of variables. In *Soviet. Math. Dokl*, volume 108, pages 179–182.

Kolodzey, J. (1981). CRAY-1 Computer Technology. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, 4(2):181–186.

Komatitsch, D. and Tromp, J. (2002). Spectral-element simulations of global seismic wave propagation - I. Validation. *Geophysical Journal International*, 149(2):390–412.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Weinberger, F. P., Burges, C. J. C., Bottou, L., and Q., K., editors, *Advances in neural information processing systems*, volume 25, pages 1097–1105. Curran Associates, Inc.

Krizhevsky, A., Sutskever, I., Hinton, G. E., Tasci, T., and Kim, K. (2015). ImageNet Classification with Deep Convolutional Neural Networks. *Stanford cs231b*.

Kumar, J., Ramrez, A., and Butt, S. (2012a). Preparing Data for Full Waveform Inversion: A Workflow for Free-surface Multiple Attenuation. *74th EAGE Conference and Exhibition-Workshops*.

Kumar, M., Manral, D. S., Banerjee, M. K., Karmakar, K., Das, A., Reddy, B. J., Dasgupta, R., and Singh, &. S. N. (2012b). High Performance Computing in Geosciences: Promises & Challenges. *9th Biennial International Confrence & Exposition on Petroleum Geophysics*.

Lailly, P. and Bednar, J. (1983). The seismic inverse problem as a sequence of before stack migrations. *Conference on inverse scattering: theory and application*.

Lang, K. J. and Hinton, G. E. (1988). The development of the time-delay neural network architecture for speech recognition. *Technical Report CMU-CS-88-152*.

Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1):23–43.

Langer, H., Nunnari, G., and Occhipinti, L. (1996). Estimation of seismic waveform governing parameters with neural networks. *Journal Of Geophysical Research-Solid Earth*, 101(B9):20109.

Larsson, G., Maire, M., and Shakhnarovich, G. (2016). Learning representations for automatic colorization. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9908 LNCS, pages 577–593.

Le, T. A., Baydin, A. G., Zinkov, R., Wood, F., Atılım, G., Zinkov, R., and Wood, F. (2017). Using Synthetic Data to Train Neural Networks is Model-Based Reasoning. *ArXiv e-prints*, pages 3514–3521.

LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in perspective*, 19:143–155.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE.

Lee, C.-Y., Gallagher, P. W., and Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pages 464–472.

Lee, K. and Moloney, D. (2018). Evaluation of synthetic data for deep learning stereo depth algorithms on embedded platforms. In *2017 4th International Conference on Systems and Informatics, ICSAI 2017*, volume 2018-Janua, pages 170–176. IEEE.

Lee, M. W., Hutchinson, D. R., Collett, T. S., and Dillon, W. P. (1996). Seismic velocities for hydrate-bearing sediments using weighted equation. *Journal of Geophysical Research: Solid Earth*, 101(B9):20347–20358.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer Feed Forward Networks With A Nonpolynomial Activation Function Can Approximate Any Function. *Neural Networks*, 6:861–867.

Levin, L. (1973). Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116.

Lewis, W. and Vigh, D. (2017). Deep learning prior models from seismic images for full-waveform inversion. *SEG Technical Program Expanded Abstracts 2017*, pages 1512–1517.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018a). Visualizing the Loss Landscape of Neural Nets. In *Neural Information Processing Systems*.

Li, H., Yang, W., and Yong, X. (2018b). Deep learning for ground-roll noise attenuation. In *SEG Technical Program Expanded Abstracts 2018*, pages 1981–1985. Society of Exploration Geophysicists.

Li, S., Liu, B., Ren, Y., Chen, Y., Yang, S., Wang, Y., and Jiang, P. (2019). Deep-learning inversion of seismic data. *arXiv preprint arXiv:1901.07733*.

Lin, T., Horne, B. G., Tino, P., and Giles, C. L. (1996). Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338.

Lines, L. (2014). FWI and the "Noise" Quandary. *CREWES Research Report*.

Linnainmaa, S. (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7.

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT*, 16(2):146–160.

Lippmann, R. P. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, 4(April):4–22.

Liu, Y., He, B., and Zheng, Y. (2020). Controlled-order multiple waveform inversion. *Geophysics*, 85(3):R243–R250.

Liu, Y. and Sen, M. K. (2011). 3D acoustic wave modelling with time-space domain dispersion-relation-based finite-difference schemes and hybrid absorbing boundary conditions. *Exploration Geophysics*, 42(3):176–189.

Lucas, A., Iliadis, M., Molina, R., and Katsaggelos, A. K. (2018). Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods. *IEEE Signal Processing Magazine*, 35(1):20–36.

Malcolm, A. and Willemsen, B. (2016). Rapid 4D FWI using a local wave solver. *The Leading Edge*, 35(12):1053–1059.

Mangalathu, S., Jang, H., Hwang, S.-H., and Jeon, J.-S. (2020). Data-driven machine-learning-based seismic failure mode identification of reinforced concrete shear walls. *Engineering Structures*, 208:110331.

Martens, J. (2010). Deep learning via Hessian-free optimization. *27th International Conference on Machine Learning*, 951:735–742.

Martin, G. S., Marfurt, K. J., and Larsen, S. (2002). Marmousi-2: An updated model for the investigation of AVO in structurally complex areas. In *SEG Technical Program Expanded Abstracts 2002*, pages 1979–1982. Society of Exploration Geophysicists.

Martin, G. S., Wiley, R., and Marfurt, K. J. (2006). Marmousi2: An elastic upgrade for Marmousi. *The Leading Edge*, 25(2):156–166.

McCormack, M. D., Zaucha, D. E., and Dushek, D. W. (1993). First-break refraction event picking and seismic data trace editing using neural networks. *Geophysics*, 58(1):67–78.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Meinhardt, T., Moeller, M., Hazirbas, C., and Cremers, D. (2017). Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 1799–1808.

Menke, W. (1989). Geophysical Data Analysis: Discrete Inverse Theory.

Michaels, P. and Smith, R. B. R. B. (1992). Recurrent Neural Network representation of the Inelastic Wave Equation and Full-Waveform inversion without local minima. *62nd Ann. Internat. Mtg*, pages 22–25.

Minsky, M. and Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

Mispel, J., Furre, A., Sollid, A., and Maaø, F. A. (2019). High Frequency 3D FWI at Sleipner: A Closer Look at the CO2 Plume. In *81st EAGE Conference and Exhibition 2019*.

Molinari, I., Obermann, A., Kissling, E., Hetényi, G., and Boschi, L. (2020). 3D crustal structure of the Eastern Alpine region from ambient noise tomography. *Results in Geophysical Sciences*, 1-4:100006.

Molinari, I., Verbeke, J., Boschi, L., Kissling, E., and Morelli, A. (2015). Italian and A lpine three-dimensional crustal structure imaged by ambient-noise surface-wave dispersion. *Geochemistry, Geophysics, Geosystems*, 16(12):4405–4421.

Møller, M. (1993). Exact Calculation of the Product of the Hessian Matrix of Feed-Forward Network Error Functions and a Vector in 0 (N) Time. *DAIMI Report Series*, page 14.

Montúfar, G., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the Number of Linear Regions of Deep Neural Networks. *arXiv preprint arXiv:1402.1869*.

Morgan, J., Warner, M., Arnoux, G., Hooft, E., Toomey, D., VanderBeek, B., and Wilcock, W. (2016). Next-generation seismic experiments - II: Wide-angle, multi-azimuth, 3-D, full-waveform inversion of sparse field data. *Geophysical Journal International*, 204(2):1342–1363.

Morgan, J., Warner, M., Bell, R., Ashley, J., Barnes, D., Little, R., Roele, K., and Jones, C. (2013). Next-generation seismic experiments: wide-angle, multi-azimuth, three-dimensional, full-waveform inversion. *Geophysical Journal International*, 195(3):1657–1678.

Morgan, J. V., Christeson, G. L., and Warner, M. (2009). Using swath bathymetry as an a priori constraint in a 3D full wavefield tomographic inversion of seismic data across oceanic crust. In *AGU Fall Meeting Abstracts*, volume 1, page 7.

Mothi, S., Schwarz, K., and Zhu, H. (2013). Impact of full-azimuth and long-offset acquisition on Full Waveform Inversion in deep water Gulf of Mexico. *SEG Houston 2013 Annual Meeting*, (June 2013):924–928.

Mozer, M. C. (1992). Induction of multiscale temporal structure. In *Advances in neural information processing systems*, pages 275–282.

Murat, M. E. and Rudman, A. J. (1992). Automated First Arrival Picking: a Neural Network Approach. *Geophysical Prospecting*, 40(6):587–604.

Nath, S. K., Chakraborty, S., Singh, S. K., and Ganguly, N. (1999). Velocity inversion in cross-hole seismic tomography by counter-propagation neural network, genetic algorithm and evolutionary programming techniques. *Geophysical Journal International*, 138(1):108–124.

Nazari Siahsar, M. A., Gholtashi, S., Kahoo, A. R., Chen, W., and Chen, Y. (2017). Data-driven multitask sparse dictionary learning for noise attenuation of 3D seismic data. *Geophysics*, 82(6):V385–V396.

Newton, I. (1687). Philosophiae Naturalis Principia Mathematica. *Pan*, page 510.

Nguyen, B. and Mcmechan, G. (2015). Five ways to avoid storing source wavefield snapshots in 2D elastic prestack reverse time migration. *GEOPHYSICS*, 80:S1–S18.

Nicholson, A. and Gibson, A. (2016). Deeplearning4j: Open-source distributed deep learning for the jvm. *Apache Software Foundation License*, 2.

Noble, M., Gesret, A., and Belayouni, N. (2014). Accurate 3-D finite difference computation of traveltimes in strongly heterogeneous media. *Geophysical Journal International*, 199(3):1572–1585.

Öktem, O. and Adler, J. (2018). Mathematics of Deep Learning with an Emphasis on Inverse Problems.

Olah, C. (2015). Understanding lstm networks.

Operto, S., Miniussi, A., Brossier, R., Combe, L., Haller, N., Kjos, E., and Metivier, L. (2015). Efficient 3D Frequency-domain Full-waveform Inversion of Ocean-bottom Cable Data - Application to Valhall in the Visco-ac. In *EAGE Extended Abstracts*.

Operto, S., Ravaut, C., Improta, L., Virieux, J., Herrero, A., and Dell'Aversana, P. (2004). Quantitative imaging of complex structures from dense wide-aperture seismic data by multiscale traveltime and waveform inversions: a case study. *Geophysical Prospecting*, 52(6):625–651.

Operto, S., Virieux, J., Amestoy, P., L'Excellent, J.-Y., Giraud, L., and Ali, H. B. H. (2007). 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study. *Geophysics*, 72(5):SM195–SM211.

Operto, S., Virieux, J., Dessa, J., and Pascal, G. (2006). Crustal seismic imaging from multifold ocean bottom seismometer data by frequency domain full waveform tomography: Application to the eastern Nankai trough. *Journal of Geophysical Research: Solid Earth*, 111(B9).

Parker, P. B. (1999). *Genetic algorithms and their use in geophysical problems*. PhD thesis, University of California, Berkeley.

Paszke, A., Chanan, G., Lin, Z., Gross, S., Yang, E., Antiga, L., and Devito, Z. (2017). Automatic differentiation in PyTorch. *Advances in Neural Information Processing Systems*, 30(Nips):1–4.

Patterson, J. and Gibson, A. (2017). *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.".

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269.

Peng, C., Wang, M., Chazalnoel, N., and Gomes, A. (2018). Subsalt imaging improvement possibilities through a combination of FWI and reflection FWI. *The Leading Edge*, 37(1):52–57.

Petersen, P. C., Bölcskei, H., Grohs, P., and Kutyniok, G. (2017). Optimal approximation with sparse deep neural networks.

Plate, T. A. (1993). Holographic recurrent networks. In *Advances in neural information processing systems*, pages 34–41.

Plessix, R. (2009). Three-dimensional frequency-domain full-waveform inversion with an iterative solver. *Geophysics*, 74(6):WCC149–WCC157.

Plessix, R.-E. and Perkins, C. (2010). Thematic Set: Full waveform inversion of a deep water ocean bottom seismometer dataset. *First Break*, 28(4):71–78.

Plessix, R.-É., Stopin, A., Milcik, P., and Matson, K. (2014). Acoustic and anisotropic multi-parameter seismic full waveform inversion case studies. In *SEG Technical Program Expanded Abstracts 2014*, pages 1056–1060. Society of Exploration Geophysicists.

Poulton, M. M., Sternberg, B. K., and Glass, C. E. (1992). Location of subsurface targets in geophysical data using neural networks. *Geophysics*, 57(12):1534–1544.

Pratt, R. G. (1990). Inverse theory applied to multi-source cross-hole tomography - Part 2: Elastoc Wave-Equation Method. *Geophysical Prospecting*, 38(3):311–329.

Pratt, R. G. (1999). Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale model. *GEOPHYSICS*, 64(3):888–901.

Pratt, R. G. and Goulty, N. R. (1991). Combining wave-equation imaging with traveltime tomography to form high-resolution images from crosshole data. *Geophysics*, 56(2):208–224.

Pratt, R. G., Song, Z.-M., Williamson, P., and Warner, M. (1996). Two-dimensional velocity models from wide-angle seismic data by wavefield inversion. *Geophysical Journal International*, 124(2):323–340.

Pratt, R. G. and Worthington, M. H. (1990). Inverse theory applied to multi-source cross-hole tomography - Part 1: Acoustic Wave-Equation Method. *Geophysical prospecting*, 38(March 1989):287–310.

Press, F. (1968). Earth Models Obtained by Monte Carlo Inversion. *J. Geophys. Res.*, 73(16):5223–5234.

Pullammanappallil, S. K. and Louie, J. N. (1994). A generalized simulated-annealing optimization for inversion of first-arrival times. *Bulletin of the Seismological Society of America*, 84(5):1397–1409.

Puskorius, G. V. and Feldkamp, L. A. (1994). Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. *IEEE Transactions on neural networks*, 5(2):279–297.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Randall, P. (2009). Pink Color of Seismic noise in the low frequency limit. [Accessed 10/8/2014].

Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

Raschka, S. and Mirjalili, V. (2017). *Python machine learning*. Packt Publishing Ltd.

Reading, A. M., Cracknell, M. J., Bombardieri, D. J., and Chalke, T. (2015). Combining Machine Learning and Geophysical Inversion for Applied Geophysics. In *ASEG-PESA 2015 24th International Geophysical Conference and Exhibition*, volume 2015, page 1.

Richardson, A. (2018). Seismic Full-Waveform Inversion Using Deep Learning Tools and Techniques. *arXiv preprint arXiv:1801.07232*.

Robinson, A. J. and Fallside, F. (1987). *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering Cambridge, MA.

Robinson, E. A. (1957). PREDICTIVE DECOMPOSITION OF SEISMIC TRACES. *GEOPHYSICS*, 22(4):767–778.

Robinson, E. A. (1967). Predictive decomposition of time series with application to seismic exploration. *Geophysics*, 32(3):418–484.

Romano, Y., Elad, M., and Milanfar, P. (2016). The Little Engine that Could: Regularization by Denoising (RED). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844.

Romeo, G. (1994). Seismic signals detection and classification using artiricial neural networks. *Annals Of Geophysics*, 37(3).

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Röth, G. and Tarantola, A. (1994). Neural networks and inversion of seismic data. *Journal of Geophysical Research*, 99(B4):6753–6768.

Rothman, D. H. (1985). *Large near-surface anomalies, seismic reflection data, and simulated annealing*. PhD thesis, Stanford University.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., Vinet, L., and Zhedanov, A. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Russell, S. and Norvig, P. (2008). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Rusu, C. and Thompson, J. (2017). Learning fast sparsifying transforms. *IEEE Transactions on Signal Processing,*, 65(16):4376–4378.

Ryan, H. (1994). Ricker, Ormsby, Klauder, Butterworth - A choice of wavelets. *CSEG Recorder*, pages 8–9.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493.

Schakel, M. D. and Mesdag, P. R. (2014). Fully data-driven quantitative reservoir characterization by broadband seismic. In *2014 SEG Annual Meeting*. OnePetro.

Schmidhuber, J. (1992a). A fixed size storage O (n 3) time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation*, 4(2):243–248.

Schmidhuber, J. (1992b). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

Schmidhuber, J. (2015). Deep learing in neural networks: an overview. *Neural Networks*, 61.

Schraudolph, N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738.

Schultz, P. S., Ronen, S., Hattori, M., and Corbett, C. (1994). Seismic-guided estimation of log properties (Part 1: A data-driven interpretation methodology). *The Leading Edge*, 13(5):305–310.

Sen, M. K. and Stoffa, P. L. (1995). *Global optimization methods in geophysical inversion*. Elsevier.

Sever, A. (2015). An inverse problem approach to pattern recognition in industry. *Applied Computing and Informatics*, 11(1):1–12.

Shahnas, M. H., Yuen, D. A., and Pysklywec, R. N. (2018). Inverse Problems in Geodynamics Using Machine Learning Algorithms. *Journal of Geophysical Research: Solid Earth*, 123(1):296–310.

Shalova, A. and Oseledets, I. (2020). Tensorized Transformer for Dynamical Systems Modeling. *arXiv preprint arXiv:2006.03445*.

Sharma, M., Pachori, R. B., and Rajendra Acharya, U. (2017). Adam: a Method for Stochastic Optimization. *Pattern Recognition Letters*, 94:172–179.

Shen, X. and Clapp, R. (2015). Random boundary condition for memory-efficient waveform inversion gradient computation. *GEOPHYSICS*, 80:R351–R359.

Sheriff, R. E. and Geldart, L. P. (1985). *Exploration seismology. Volume 2*. Cambridge University Press, New York, NY, United States.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306.

Shimshoni, Y. and Intrator, N. (1998). Classification of seismic signals by integrating ensembles of neural networks. *IEEE Transactions on Signal Processing*, 46(5):1194–1201.

Shin, C., Koo, N.-H. H., Cha, Y. H., and Park, K.-P. P. (2010). Sequentially ordered single-frequency 2-D acoustic waveform inversion in the Laplace-Fourier domain. *Geophysical Journal International*, 181(2):935–950.

Siahkoohi, A., Louboutin, M., and Herrmann, F. J. (2019). The importance of transfer learning in seismic modeling and imaging. *GEOPHYSICS*, 84(6):A47–A52.

Simonyan, K., Others, and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, pages 1–14.

Singh, R. V. (2015). ImageNet Winning CNN Architectures – A Review. pages 3–8.

Sirgue, L., Barkved, O. I., Dellinger, J., Etgen, J., Albertin, U., and Kommedal, J. H. (2010). Thematic set: Full waveform inversion: The next leap forward in imaging at Valhall. *First Break*, 28(4):65–70.

Sirgue, L., Barkved, O. I., Van Gestel, J. P., Askim, O. J., and Kommedal, J. H. (2009). 3D waveform inversion on Valhall wide-azimuth OBC. In *71st EAGE Conference and Exhibition incorporating SPE EUROPEC 2009*.

Sirgue, L., Etgen, J., Albertin, U., and America, B. P. (2007). 3D full-waveform inversion: Wide-versus narrow-azimuth acquisitions. *SEG Technical Program Expanded Abstracts 2007*, pages 1760–1764.

Sirgue, L. and Pratt, R. G. (2004). Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies. *Geophysics*, 69(1):231–248.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Stephens, A. D. (2006). Measurable functions. *M3/M4S3 - Statistical Theory II Lecture Notes*.

Sun, J., Niu, Z., Innanen, K. A., Li, J., and Trad, D. O. (2019). A theory-guided deep learning formulation of seismic waveform inversion. In *SEG Technical Program Expanded Abstracts 2019*, pages 2343–2347. Society of Exploration Geophysicists.

Sun, Y., Xia, Z., and Kamilov, U. S. (2018). Efficient and accurate inversion of multiple scattering with deep learning. *Optics Express*, 26(11):14678.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., Hill, C., and Arbor, A. (2014). Going Deeper with Convolutions. *cv-foundation.org*, pages 1–9.

Tanaka, M. M. (2003). *Inverse problems in engineering mechanics IV : International Symposium on Inverse Problems in Engneering Mechanics 2003 (ISIP 2003), Nagano, Japan*. Elsevier.

Tarantola, A. (1984a). Inversion of seismic reflection data in the acoustic approximation. *GEOPHYSICS*, 49(8):1259–1266.

Tarantola, A. (1984b). Linearized inversion of seismic reflection data. *Geophysical prospecting*, 32(6):998–1015.

Tarantola, A. (1987). Inverse problem theory: Methods for data fitting and parameter estimation.

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*.

Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., and Goldstein, T. (2016). Training neural networks without gradients: A scalable admm approach. In *International conference on machine learning*, pages 2722–2731. PMLR.

Thulasiraman, K. and Swamy, M. N. S. (2011). *Graphs: theory and algorithms*. John Wiley & Sons.

Tikhonov, A. N. (1963). On the Solution of Incorrectly Stated Problems and a Method of Regularization. *Dokl. Acad. Nauk SSSR*, 151:501–504.

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of ill-posed problems*. Winston, New York.

Törn, A. and Žilinskas, A. (1989). *Global optimization*, volume 350. Springer.

Tran, K. T. and Hiltunen, D. R. (2011). Two-dimensional inversion of full waveforms using simulated annealing. *Journal of Geotechnical and Geoenvironmental Engineering*, 138(9):1075–1090.

Tran, K. T. and Hiltunen, D. R. (2012). One-Dimensional Inversion of Full Waveforms using a Genetic Algorithm. *Journal of Environmental & Engineering Geophysics*, 17(4):197–213.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Vigh, D., Starr, B., Kapoor, J., and Li, H. (2010). 3D Full waveform inversion on a Gulf of Mexico WAZ data set. *SEG Technical Program Expanded Abstracts 2010*, pages 957–961.

Vigh, D. and Starr, E. W. (2008). 3D prestack plane-wave, full-waveform inversion. *GEOPHYSICS*, 73(5):VE135–VE144.

Virieux, J. and Operto, S. (2009). An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26.

Vito, E. D., Rosasco, L., Caponnetto, A., Giovannini, U. D., and Odone, F. (2005). Learning from Examples as an Inverse Problem. *Journal of Machine Learning Research*, 6(May):883–904.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

Wang, H. and Tsvankin, I. (2016). Feasibility of waveform inversion in acoustic orthorhombic media. *SEG Technical Program Expanded Abstracts 2016*, pages 311–316.

Wang, L. and Mendel, J. M. (1992). Adaptive minimum prediction-error deconvolution and source wavelet estimation using Hopfield neural networks. *Geophysics*, 57(5):670–679.

Wang, T., Wu, D. J., Coates, A., and Ng, A. Y. (2012). End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3304–3308. IEEE.

Wang, Y. (2015). Frequencies of the Ricker wavelet. *Geophysics*, 80(2):A31—-A37.

Wang, Z. and Oates, T. (2015). Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, volume 1.

Warner, M., Ratcliffe, A., Nangoo, T., Morgan, J., Umpleby, A., Shah, N., Vinje, V., Štekl, I., Guasch, L., Win, C., Conroy, G., and Bertrand, A. (2013). Anisotropic 3D full-waveform inversion. *GEOPHYSICS*, 78(2):R59–R80.

Warner, M., Stekl, I., and Umpleby, A. (2007). Full Wavefield Seismic Tomography–Iterative Forward Modelling in 3D. In *69th EAGE Conference and Exhibition incorporating SPE EUROPEC 2007*.

Warner, M., Stekl, I., and Umpleby, A. (2008). 3D wavefield tomography : synthetic and field data examples. *2008 SEG Annual Meeting*, pages 3330–3334.

Webster, G. M. (1978). *Deconvolution*. Society of Exploration Geophysicists, Tulsa, OK, geophysics edition.

Wei, Q., Fai, K., and Carin, L. (2017). An Inner-loop Free Solution to Inverse Problems using Deep Neural Networks. *Advances in Neural Information Processing Systems*, pages 2370–2380.

Werbos, P. J. (1981). Applications of Advances in Nonlinear Sensitivity Analysis. In *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*, pages 762–770.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Wu, G., Fomel, S., and Chen, Y. (2018a). Data-driven time–frequency analysis of seismic data using non-stationary Prony method. *Geophysical Prospecting*, 66(1):85–97.

Wu, Y., Lin, Y., and Zhou, Z. (2018b). InversionNet: Accurate and efficient seismic waveform inversion with convolutional neural networks. In *SEG Technical Program Expanded Abstracts 2018*, pages 2096–2100. Society of Exploration Geophysicists.

Xie, Y., Zhou, B., Zhou, J., Hu, J., Xu, L., Wu, X., Lin, N., Loh, F. C., Liu, L., and Wang, Z. (2017). Orthorhombic full-waveform inversion for imaging the Luda field using wide-azimuth ocean-bottom-cable data. *The Leading Edge*, 36(1):75–80.

Xu, Q., Yu, H., Mou, X., Zhang, L., Hsieh, J., and Wang, G. (2012). Low-dose X-ray CT reconstruction via dictionary learning. *IEEE transactions on medical imaging*, 31(9):1682–1697.

Yadav, N., Yadav, A., and Kumar, M. (2015). *An Introduction to Neural Network Methods for Differential Equations*.

Yao, G., Wu, D., and Wang, S.-X. (2020). A review on reflection-waveform inversion. *Petroleum Science*, 17(2):334–351.

Yao, Y., Rosasco, L., and Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315.

Yu, S. and Principe, J. C. (2019). Understanding autoencoders with information theoretic concepts. *Neural Networks*, 117:104–123.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zhang, D. L., Dai, W., Ge, Z., and Schuster, G. (2013). Multiples waveform inversion. In *75th EAGE Conference & Exhibition incorporating SPE EUROPEC 2013*, pages cp–348. European Association of Geoscientists & Engineers.

Zhang, J., Lin, Y., Song, Z., and Dhillon, I. S. (2018). Learning Long Term Dependencies via Fourier Recurrent Units.

Zhang, X., Jia, Z., Ross, Z. E., and Clayton, R. W. (2020). Extracting dispersion curves from ambient noise correlations using deep learning. *IEEE Transactions on Geoscience and Remote Sensing*, 58(12):8932–8939.

Zhang, Y. and Paulson, K. V. (1997). Magnetotelluric inversion using regularized Hopfield neural networks. *Geophys. Prosp.*, 45(05):725–743.

Zhou, Y.-T. and Chellappa, R. (1988). Computation of optical flow using a neural network. In *ICNN*, pages 71–78.

Zhu, C. Y., Byrd, R. H., Lu, P. H., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *Acm Transactions on Mathematical Software*, 23(4):550–560.

Zhu, W., Xu, K., Darve, E., Biondi, B., and Beroza, G. C. (2020). Integrating Deep Neural Networks with Full-waveform Inversion: Reparametrization, Regularization, and Uncertainty Quantification. *arXiv preprint arXiv:2012.11149*.

Zimmermann, H. G., Grothmann, R., Schaefer, A. M., and Tietz, C. (2006). Identification and Forecasting of Large Dynamical Systems by Dynamical Consistent Neural Networks. *S. Haykin, J. Principe, T. Sejnowski, and J. McWhirter, editors, New Directions in Statistical Signal Processing: From Systems to Brain*, pages 203–242.